Value-as-Property Semantics for Propagation-Oriented Multi-Level Modeling

Hermann Bense
hb@bense.com
[bense.com] Verlagsgesellschaft für digitales Publizieren mbH
Dortmund, Germany

Abstract

Multi-level modelling organises types, instances, potency, and declarations across abstraction layers. When taxonomies and product families evolve, cross-layer feature commitments and literal rematerialisation create recurring maintenance pressure.

A central architectural reading is that much of this pressure tracks how literal commitments are treated as repeatedly layer-local materialisations rather than as propagation-governed semantic carriers along admissible unary refinements, wherever policies and compatibility make that reading legitimate.

We foreground a *property-centric* angle: *Value-as-Property* (VAP) treats selected commitments as Δ-properties—transparent carriers whose literals may remain visible along those refinements when propagation policies and layer compatibility allow. A progression from plain, meta-level, and uniform schema data properties toward transparent data properties sketches the design space alongside, not instead of, established instantiation-centric accounts. The paper contributes a propagation-policy vocabulary, qualitative contrasts on materialisation pressure, and an F6–F8-oriented narrative; Section 6 sharpens an exploratory layer-versus-level normalization hypothesis in the same spirit.
Scope. Workshop exposition without empirical benchmark; vocabulary and contrasts are not offered as a closed calculus; entailment-centric OWL/DL classification on every individual is out of scope for the Δ-policy story told here.

Keywords

Multi-Level Modeling, Property-centric MLM, Value-as-Property, Semantic Propagation, O4Top, Deriver, Abstraction Layers

1 Introduction

Research on MLM supplies clabjects, potency, and instantiation chains [2, 13]. A recurring friction is how literal commitments *travel* between layers without combinatorial re-declaration [6, 24]. Hierarchy evolution and reclassification increase the cost of restating cross-layer values [13, 25]. Layering fixes *where* symbols live and potency constrains depth [3]; serialisations still differ in how cleanly unary refinements carry literals [6].

The wider landscape spans multilevel metamodelling and rationales [2, 3, 13], comparative dimensions F1–F8 [12], MLT-style instantiation theory [1, 10], expressive MLM serialisations toward RDF/OWL and DL classification [4, 9, 26], surveys that relate deep instantiation to ConceptBase/Telos-style kernels and tool chains [12, 20, 22], and abstraction studies on hierarchy evolution and materialisation pressure [24, 25]. This note foregrounds *propagation* of literal visibility along admissible unary refinements as distinct from, yet coordinated with, potency-centric *instantiation* bookkeeping; Section 2 develops citations and positioning in full.

The contribution is a *property-centric propagation layer*: F6–F8 propagation along PDP → MDP → UDP → TDP/VAP [12, 25], with `.ΔName` for transparency policies [6, 8]. O4Top and Deriver supply compact carrier syntax and unary $A(B)$ traces [5–7]. Figures illustrate Δ-examples and a barometer chain [13].
Positioning. The note targets policy-governed visibility of literals along derivations; it sits beside MLT-style instantiation and OWL/DL classification [4, 10, 26], not as a substitute foundation or global subclass-entailment story.

The stance is intentionally *engineering-oriented* rather than foundational: propagation readability and maintenance behaviour under hierarchy evolution matter here more than a reset of potency or deep-instantiation theory [3, 12]. VAP complements instantiation-centric MLM semantics; it does not replace them [10].
MULTI relevance. Propagation-oriented feature semantics align with workshop themes on layers and reuse [2].
Outline. Related work and F6–F8 anchoring; layering and VAP/Δ-properties with a propagation sketch; powertype and meta-level contrasts; examples; *Ontologies Reloaded* (layers vs. levels); discussion and outlook.

2 Background and Related Positioning

Atkinson and Kühne articulate multilevel metamodelling [2, 3]; Frank [13] surveys rationales for multi-level conceptual modelling; Neumayr, Schrefl, and Thalheim survey abstraction techniques [24]. Guizzardi [17] and UFO with Almeida et al. [18] clarify roles and dependence; unary constructors echo frame projections [11].

Fonseca et al. organise comparative MLM along F1–F8 [12]. We use F6–F8 (feature assignments, cross-level feature relations, cross-level domain relations) for propagation semantics; full instantiation theory stays with MLT [1, 10] and expressive MLM for RDF/OWL serialisations [9]. Surveys [12] cite Melanie/Melanee, MetaDepth, DeepTelos, DeepJava-style deep instantiation, ConceptBase-style kernels, and MultiEcore when contrasting instantiation durability with cross-layer property visibility. O4Top stays compact relative to rich foundational programmes [5, 8]. Formal Concept Analysis (FCA) [14] studies attribute implications in intent–extent structure; Δ-policies may *loosely* recall how attribute-like commitments stay organisationally salient, yet they remain derivation- and policy-centric rather than lattice-centric, and no FCA embedding is claimed.

MULTI 2026, co-located with MODELS 2026
2026.

3 Property-Centric Multi-Level Modeling

O4Top provides a small vocabulary for classes, properties, particulars, and relators as graph artefacts [5]. It doubles as a lightweight *carrier vocabulary* for ordinary data, meta-level, and transparent propagation-oriented property shapes when aligning exports to RDF/OWL-style graphs [4, 5]. It complements UFO [18]; unary constructions map onto RDF/OWL-shaped graphs [4, 26]. Optional PNG assets (`o4top.png`) remain illustrative only.

Cross-level object relations (S1–S4). Besides class attributes and value-as-property carriers, IN^3/O4Top-oriented encodings also admit *relations* between particulars and class-shaped nodes, with local data carried on the relator itself—a reading aligned with cross-level abstraction requirements S1–S4 in the sense of Neumayr et al. [23, 24].

```
(>isDesignerOf_LM, .YearOfDesign, 1966)
(»NPS-Marcello_Gandini, >isDesignerOf_LM,
  .^Lamborghini_Miura)
```

Reading. `NPS-Marcello_Gandini` is a particular; `Lamborghini_Miura` a class-shaped type; `isDesignerOf_LM` links them, while `.YearOfDesign = 1966` stays on the relator instead of being scattered across designer, model, class-wide slots, or every instance—supporting S4 (links between objects and specialisations across layers) and S3 (local attachment at the relationship) [23, 24]. Regularity-oriented patterns in MLT-style accounts attach attribute-like values to instance-of-instance shapes (e.g. regularity-scoped readings akin to `instancesScreenSize`) [10, 12]; they do not by themselves disambiguate *class-level* data-property readings from instance-level ones. Without an explicit instantiation facet for class-anchored data properties, merging `MobilePhoneModel` with `MobilePhone` risks spurious propagation, e.g. `launchDate` appearing on every handset [10].

*Layer-crossing relations from **AL0** to **AL1**.* IN^3/O4Top supports explicit layer-crossing object relations between particulars and classes.
In Figure 1, the particular `>NPS_Marcello_Gandini` links via `»pof` to the class `^natPerson`, while simultaneously participating in `»isDesignerOf` targeting `^Lamborghini_Miura`. At the upper abstraction layer, the relation type is declared between `^natPerson` and `^Car`. This illustrates direct semantic layer crossing without auxiliary metaclasses, alongside the usual class hierarchy on **AL1** (`^Lamborghini_Miura »is ^Lamborghini` and `^Lamborghini »is ^Car`). The pattern aligns with requirement S4 (relationships between entities at different abstraction levels) in the sense of Neumayr and Schrefl [23].

```
^Lamborghini_Miura >>is ^Lamborghini
^Lamborghini >>is ^Car
^natPerson <>isDesignerOf ^Car

>NPS_Marcello_Gandini >>pof ^natPerson
>NPS_Marcello_Gandini >>isDesignerOf ^Lamborghini_Miura
```

Write $A(B)$ for unary composition along a chain [6, 7]; surface syntax is illustrative and no completeness is claimed. Genus preservation is captured informally by

$$A(B) \sqsubseteq A. \tag{1}$$

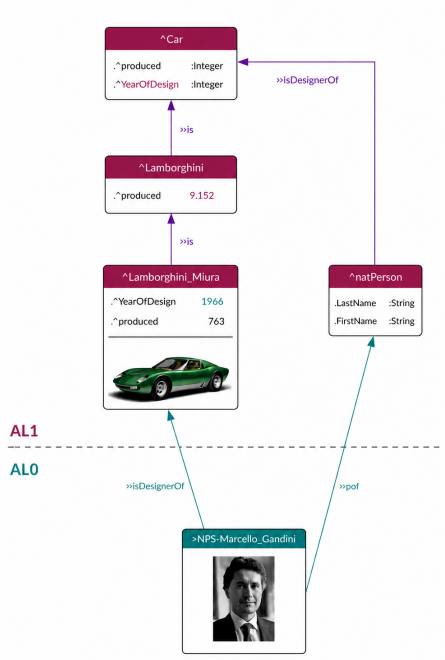


Figure 1: Cross-layer relations (O4Top/IN^3).

For transparent properties p, an engineering convention (not DL inference) is that p may remain visible *along* admissible derivational links unless propagation is blocked or the carrier is incompatible at the target step. Iterated unary steps form derivational clusters [15, 16].

Potency constrains depth [3]; unary narratives explain composed intensions [6]. Clabjects retain their usual readings [13]. Labels **AL0**–**AL3** stress semantic strata and propagating commitments [12], coexistent with MLT order indices and the M0–M3 shorthand.

The diagram in Figure 2 distinguishes instance layer (IL) from schema layer: here IL is **AL0** and schema spans **AL1**–**AL3**. The mapping is orthogonal to M0–M3 and MLT order naming; it only fixes an abstraction-layer vocabulary for cross-layer propagation of

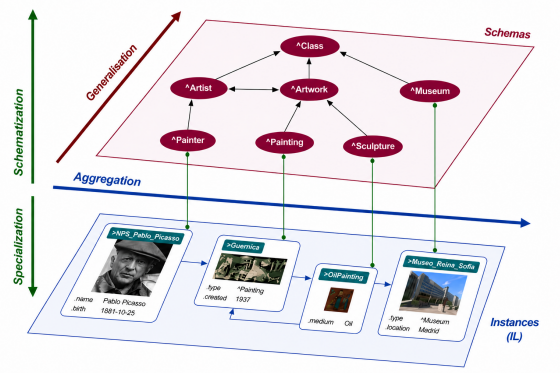


Figure 2: IL as AL0; schema AL1–AL3.

VAP/Δ-properties. VAP/Δ-properties become salient at the boundary between IL/**AL0** and schema levels **AL1**–**AL3**, because they state which literal commitments may remain visible when moving along derivational structures across those layers [6].

We summarise a *property-centric* reading of MLM: abstraction layers **AL0**–**AL3** carry not only types but also *how* literal commitments propagate. Following [12], Δ-properties / VAP sharpen cross-layer feature semantics (F6–F8) while potency remains authoritative for instantiation depth [3].

Plain data properties (PDP) attach literals inside a layer with full materialisation per assertion unless a generic mechanism applies [4]. Meta-level data properties (MDP) describe classifier-shaped artefacts and can duplicate literal shapes across meta-steps [13]. Uniform schema-level declarations (UDP) factor defaults within a layer without crossing layers [8]. Transparent data properties (TDP) / VAP generalise UDP toward cross-layer propagation: literals declared as `.ΔName` may remain visible along admissible unary refinements when policies permit—co-visible literals along the trace instead of re-storing every hop [6]. This differs from OWL subclass inheritance, which emphasises derivational readability and selective literal mobility [4].

> **Definition (Δ-property / VAP).** A Δ-*property* denotes a transparently propagating semantic property whose values may remain visible across admissible unary refinements and abstraction layers **AL0**–**AL3** without requiring explicit re-materialisation at each intermediate construction [6].

VAP reading. Here *VAP* abbreviates a propagation-oriented semantic carrier abstraction: an operational layer for policy-governed literal visibility along admissible derivational structures—not a universal inheritance mechanism [6].

Δ-properties differ from class attributes in purpose: they foreground cross-layer visibility, propagation-oriented reuse, and reduced re-materialisation when policies align [4]. Table 2 lists representative surface forms; rows are illustrative, not exhaustive.

Class-level data properties (CDP) such as `.UnitsInStock` or `.totalSold` attach to class-shaped subjects: readings may be evaluated at the classifier without treating every individual as carrying the same transparent Δ-literal [4].

Transparent Δ-carriers such as `.ΔwarmBlooded` or `.ΔRecommendedSalesPrice` instead record policy-governed literal visibility along admissible unary refinements when compatibility holds [6]. Δ-carriers still depend on semantic scope, layer compatibility, policy, and interpretation; policies interpret carriers rather than silently extending OWL-style inheritance [6].

Aggregation-oriented CDP names may use prefixes such as *total*, *count*, *avg*, *min*, or *max*. For `.totalSold`, the *total* cue signals that the class-level value can be obtained by rolling up corresponding subclass counters. Writing $\mathrm{CDP}_{\mathrm{totalSold}}(C)$ for the formal reading of `.totalSold` at classifier C, a schematic roll-up is

$$\mathrm{CDP}_{\mathrm{totalSold}}(C) \;=\; \sum_{S \in \mathrm{subClasses}(C)} \mathrm{CDP}_{\mathrm{totalSold}}(S)\,, \qquad (2)$$

stated only as illustrative schema mathematics, not as deployed aggregate infrastructure. Upward *aggregation* along specialisation hierarchies is therefore distinct from transparent Δ-propagation: aggregating CDPs *compute* summaries from subclass materialisations, whereas Δ-properties *preserve semantic visibility* of declared literals along derivations [6]. `.Extinct` remains a compact scope reminder—unconstrained Δ-visibility to arbitrary specimens would be a scope error.

MLM feature	VAP / Δ rôle	Mat. pressure
F6 Feature assignments	Candidate carriers	Medium (policy-bound)
F7 Cross-level feature relations	Primary locus	Medium–high w/o propagation
F8 Cross-level domain relations	Cross-layer attachment	High if literals repeat

Table 1: F6–F8 vs. propagation view (qualitative).

Surface property	Kind	Intuition
`.UnitsInStock`	CDP	Class stock; e.g. = 37
`.totalSold`	CDP	Roll-up metric; e.g. = 14820 (*total*)
`.ΔwarmBlooded`	TDP/VAP	Taxonomic flag; e.g. = TRUE
`.ΔRecommendedSalesPrice`	TDP/VAP	List price; e.g. = 195
`.ΔWarrantyPeriod`	TDP/VAP	Shared term (variants)
`.ΔEnergySource`	TDP/VAP	Mobility config. anchor

Table 2: Illustrative CDP vs. Δ surface forms.

Classical materialisation copies associations downward when abstract occurrences surface as concrete facts [25]; Δ-properties foreground *which* values accompany *which* unary refinements, reducing repeated literal assertion when policies align [25]. This paper standardises on `.ΔName` for pdfLaTeX clarity.

Prefix families in illustrations—dot-prefixed attributes (`.Property`), dot–caret shorthands (`.^Type`) for class-shaped layered cues, and `.ΔProperty`—are surface notation, not the full semantics in the lexeme string. Each form expands to declarations with property kind, domain, range/ADT, propagation policy, and layer scope. Controlled notation improves diagram readability without tying semantics to accidental spelling. Feature-centric surveys already separate potency from cross-layer feature behaviour [12]; the Δ-prefix marks the propagation-intensive fragment.

A stylised expansion is `.ΔMass : Float` with `propertyKind` = TransparentDataProperty/VAP, `range` = Float, `propagationPolicy` = transparent-cross-layer, and `scope` = AL0–AL3. Fully explicit models multiply declaration nodes in diagrams; prefixes stay compact because ADTs remain at declaration level.

Toward a Lightweight Propagation Calculus.

Proposition 3.1 (Propagation vocabulary). *The predicate kit below is an operational workshop vocabulary for discussing transparency along unary derivations and tooling alignment [6]. It is not a closed model- or proof-theoretic calculus: no soundness or completeness claims are made.*

The symbols instantiate Proposition 3.1.

Predicates (informal).
derives(A, B); transparent(Δp); compatible$(A, B, \Delta p)$;
blocked$(\Delta p, B)$;
local$(B, \Delta p, v)$; propagates$(\Delta p, A, B)$;
propagated$(A, B, \Delta p, v)$; value$(B, \Delta p) = v$.

Illustrative inference shapes.

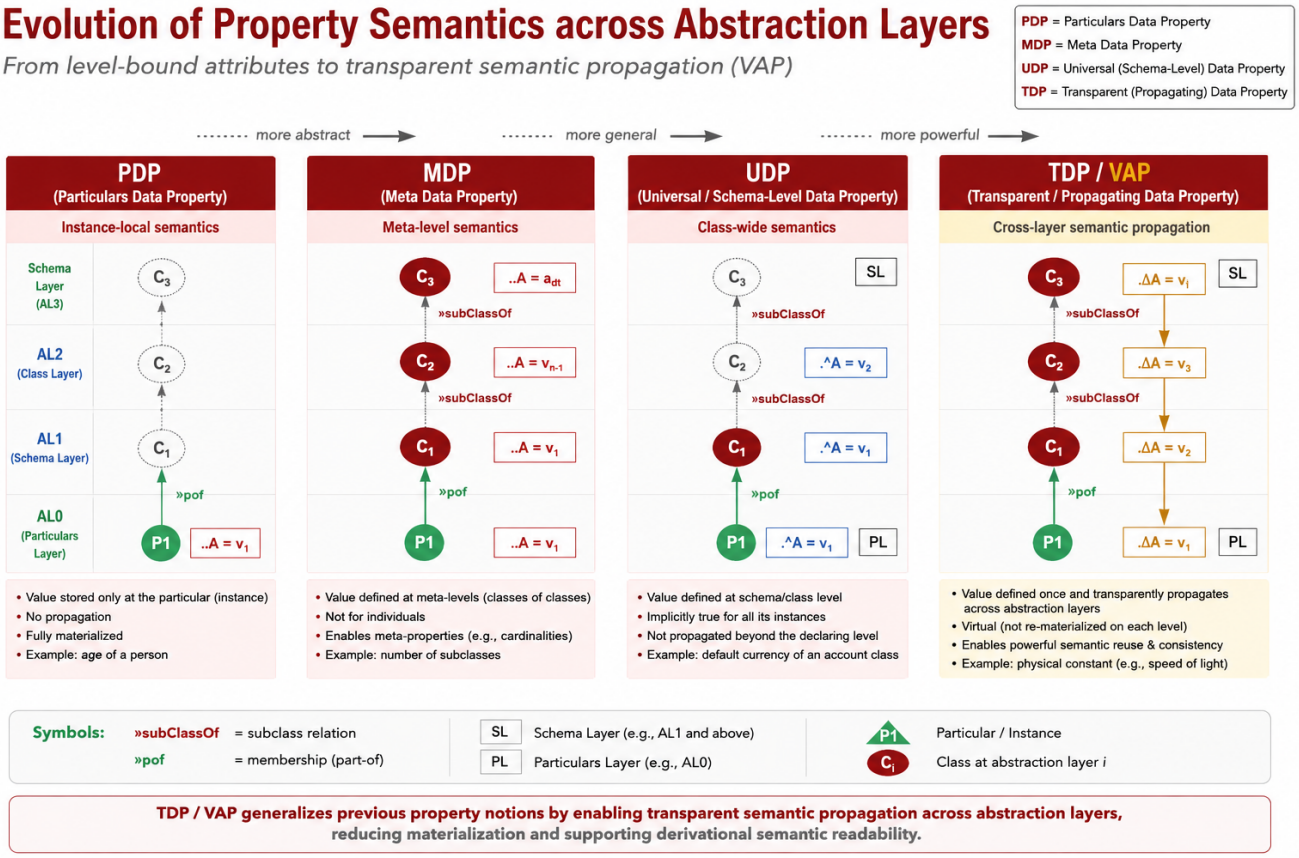


Figure 3: VAP progression PDP–TDP (F6–F8 lens).

$$\textsc{Vis}: \quad \text{transparent}(\Delta p) \wedge \text{derives}(A,B) \wedge \text{compatible}(A,B,\Delta p) \wedge \neg\,\text{blocked}(\Delta p, B) \Rightarrow \text{propagates}(\Delta p, A, B)$$

$$\textsc{Blk}: \quad \text{derives}(A,B) \wedge \text{blocked}(\Delta p, B) \Rightarrow \neg\,\text{propagates}(\Delta p, A, B)$$

$$\textsc{Ovr}: \quad \text{propagated}(A,B,\Delta p, v_1) \wedge \text{local}(B,\Delta p, v_2) \Rightarrow \text{value}(B,\Delta p) = v_2$$

$$\textsc{Scp}: \quad \neg\,\text{compatible}(A,B,\Delta p) \Rightarrow \neg\,\text{propagates}(\Delta p, A, B)$$

The sketch reads like operational typing and modular feature composition, not like a closed MLT-style calculus [1, 10]. Data abstraction [21], overriding, and loose C++-style images of virtual dispatch are *analogies* only; OWL-style subClass materialisation [4] and OO inheritance chiefly move structural or axiomatic commitments within their respective hierarchies, whereas VAP/Δ-properties expose policy-governed visibility of literals along explicit unary derivations across MLM layers [2, 6]. Telos/ConceptBase [20, 22] similarly illustrates deductive meta-modelling without equating transparency with executable code.

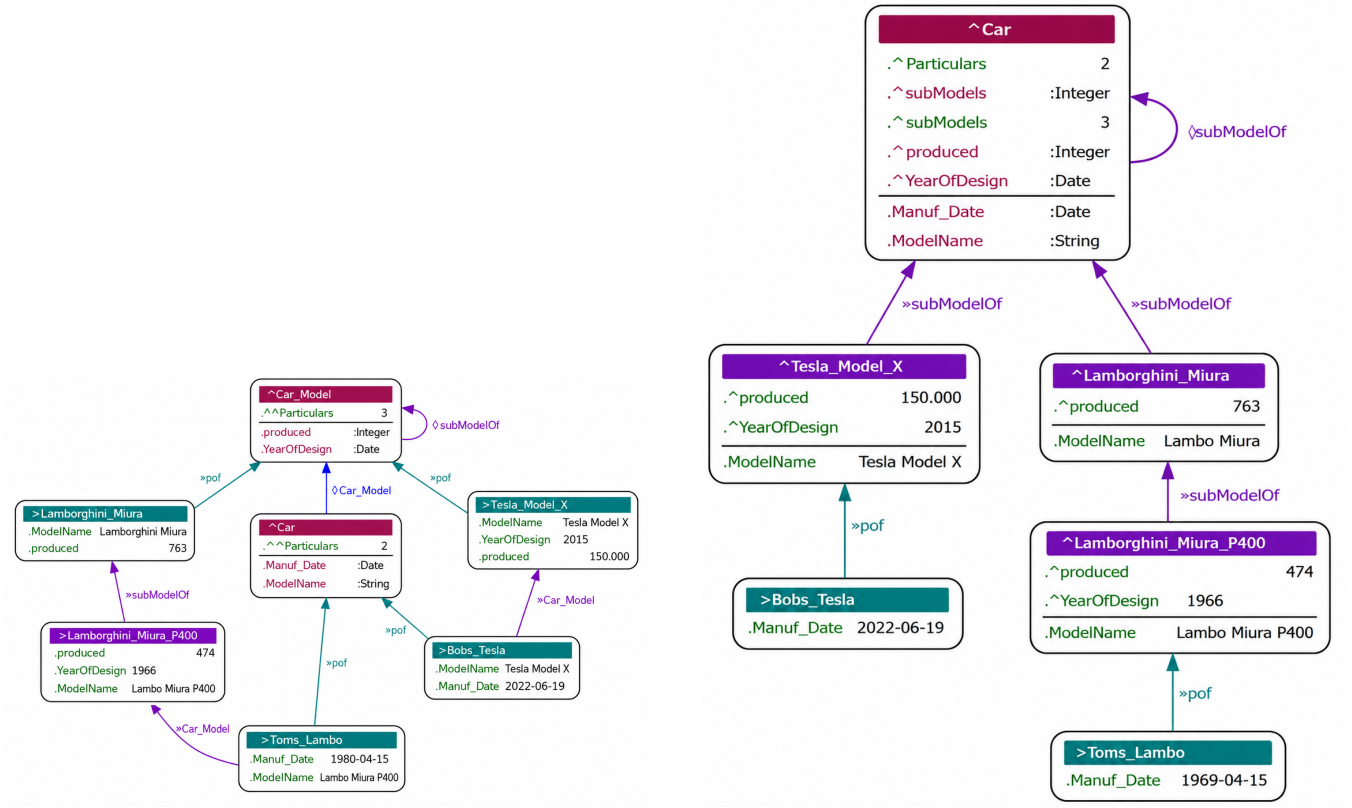
Policy	Meaning (engineering)
`transparent`	Literal visible on admissible derives steps when compatible
`blocked`	Suppress incoming derives at target; local value may still apply
`override`	Local wins for value$(B, \Delta p)$ vs. propagated candidate
`local`	Layer-local unless widened
`scope-limited`	Only within declared **AL0**–**AL3** span

Table 3: Propagation policy keywords.

4 From Powertype Subsumption to VAP Propagation

We write **AL0**–**AL3** for semantic strata and propagating commitments across a model stack, not only tool meta-level counts. A familiar explicit layout keeps model- and meta-level classes plus inter-layer relations visible; similar property clusters may reappear when refinements repeat—materialisation patterns discipline that pressure [25]. Powertype-oriented subsumption consolidates structure and reduces duplicated scaffolding while leaving powertypes intact.

Structural consolidation lowers duplicated containers (Figure 4b). VAP extends that toward *semantic* carriers: `.ΔProperty` (e.g. `.ΔEnergySource`, `.ΔRecommendedSalesPrice`) may follow unary refinements across **AL0**–**AL3** when policies apply—property-centric reuse along derivations. Pirotte et al. motivate disciplined replication [25]; Δ-properties curb repeated *property* replication



(a) Explicit layout. **(b) Powertype layout.**

Figure 4: Powertype consolidation vs. explicit layout.

where traces show literals should stick, extending consolidation while keeping powertype readings intact.

5 Examples and Architectural Perspective

Naming every variant explicitly mirrors multi-level abstraction challenges [13, 24]. VAP keeps selected commitments on the derivational spine instead of restating them per combination. A *PhoneModel* family may declare `.ΔRecommendedSalesPrice` = 195 (currency implicit), `.ΔWarrantyPeriod` = 24 months, and `.ΔDiscontinued` = *true* when retired; variants and catalogue entries reuse literals along specialisations until overridden.

A coarse chain

$$FloatingVehicle \longrightarrow Boat \longrightarrow FloatingResearchPlatform$$

illustrates capability continuity: `.ΔFloating` = *true* may remain visible where buoyancy semantics are preserved—visibility along derivations, not universal OWL entailment on every asset. For mobility,

$$electric\_vehicle = vehicle(electric),$$
$$goods\_electric\_vehicle = electric\_vehicle(goods).$$

Unary discipline forbids $vehicle(goods, electric)$, and `.ΔEnergySource` with value *electric* sketches configuration continuity without restating literals on every hop [8]. Compatibility and installed-component idioms [9, 12] remain appropriate when constraints are first-class cross-level facts; Section 7 contrasts relation-centric scaffolding with derivational propagation. Cross-layer reuse [13] stays subject to potency, typing, and propagation scope [2].

One may write

$$air\_pressure = pressure(air), \quad gauge = instrument(measurement),$$
$$barometer = gauge(air\_pressure) = gauge(pressure(air)).$$

The shortcut $instrument(pressure(air))$ stays coarse because *instrument* is polysemous [11]. Read through Deriver-style unary composition, the point is not merely to classify a name but to expose *how* the compound reading arises stepwise along pressure–air and gauge–measurement commitments [6, 7]. Resolving *gauge* against a generic *instrument* narrows frame ambiguity and documents an auditable derivation path that reviewers and tools can replay. That path is intentionally lightweight: it records semantic emergence for explanation and change management (cf. Proposition 3.1). The same spine is a natural host for later Δ-property declarations or other propagation policies once layer scope and compatibility are fixed [6].

Figure 5 visualises the graph alongside the equations above: maintenance teams can follow the intermediate *air_pressure* symbol

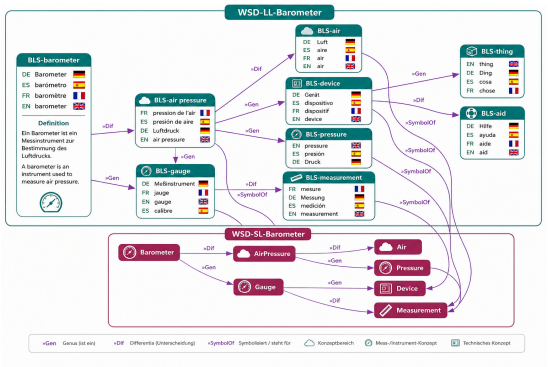


Figure 5: Unary barometer derivation graph.

when hierarchies evolve, and the layout reinforces why the coarse *instrument*(*pressure*(*air*)) shortcut hides decisions that the unary trace keeps explicit [11].

Hypothesis 5.1 (Layer versus level concentration). *Several recurring meta-level constructions may read more clearly once abstraction* layers *(distinct ontological jobs) are separated from intra-layer specialisation* levels*, and once strata beyond the structural band are held to a burden of irreducible domain justification rather than notation alone—the hypothesis Section 6 develops in exploratory form.*

Unary constructions and propagation annotations sit between authoring tools, RDF/OWL stores, and explanation services [4, 26]. Assistants may propose candidates; normalization precedes commitment [7]. Raster figures (`architecture.png`) remain optional ornamentation.

6 Ontologies Reloaded: Layers versus Levels

The following is intentionally *exploratory*: it is not offered as a replacement for the VAP/Δ contribution developed above, but as a workshop-level conceptual consequence that the propagation view *suggests*. VAP motivates this question because it relocates many apparently higher-level feature commitments into **AL1** propagation paths and Δ-policies (Section 3), which naturally raises a normalization perspective on when additional meta-level machinery still purchases irreducible domain semantics.

We distinguish **AL0**–**AL3** as the abstraction-layer vocabulary used throughout this note for propagation and worked examples (Section 3, with **AL0** labelling IL) from **OL3**–**OL0** as a complementary *functional* ontology stack used only in this section. **AL** labels mark *where* propagating commitments attach in the illustrations; **OL** labels mark *which ontological job* a layer performs (generic resource existence, structural grammar, domain meaning, concrete composition). The parallel numerals are deliberate but not synonymous: **OL** is a lens on roles, not a renaming of the **AL** strata.

This section remains deliberately bold yet provisional: it proposes a *normalisation hypothesis*, not a proof. Many MLM encodings risk conflating *abstraction layers* (distinct ontological functions) with *specialisation levels* (local hierarchies *inside* a layer), which can inflate meta-level infrastructure [2, 12, 13]. Under that lens, **OL3** hosts generic `Resource`; **OL2** hosts structural ontology primitives; **OL1** hosts domain meaning aligned with the **AL1** spine above; **OL0** hosts concrete resources and compositions.

Four functional layers (conceptual hypothesis). **OL3 (Resource.)** Universal referential existence—one structural band for "there is something", not domain laws. **OL2 (Structural primitives.)** `Class`, object/datatype properties, ADTs, `has`/`is` scaffolding—grammar of modelling. **OL1 (Domain semantics.)** Unrestricted domain specialisation: subclass and property hierarchies, derivational concept families, attribute paths, and VAP/Δ-propagation [6, 12]. **OL0 (Concrete resources.)** Bills of materials, containment trees, deployment graphs—deeply nested compositions that remain *structural* depth rather than additional semantic abstraction.

Levels inside layers. A **layer** answers *which ontological job* is performed; a **level** answers *how finely* entities specialise *within* that job. Many specialisation levels may exist inside **OL1**; by contrast, **OL2** and **OL3** are treated here as each carrying exactly one structural band of primitives (sub-primitive refinements reorganise notation but do not automatically add new domain semantics).

6.1 Minimality of OL2 (proof sketch)

The companion briefing names an **AL2** inventory of structural primitives; in this paper that inventory is carried by **OL2**, while `Resource` remains at **OL3** (cf. the layer sketch above). The following is a *minimality proof sketch* under the intended modelling function of ontology-oriented artefacts: it is not a formal completeness theorem, and it does not deny useful higher meta-levels in MLM tooling—it only sharpens what must be argued if an extra layer is claimed to be *ontological* rather than organisational.

Structural necessity. **OL2** behaves as a structural grammar, not as a domain-semantics layer: it is *structurally necessary* yet *semantically non-proliferating* in the sense that removing any core primitive immediately shrinks what can be expressed *before* one even reaches **OL1** specialisation or **OL0** composition. Without `Class`, no typed carriers for laws or facts; without object properties (`OP`), no internal links between resources; without datatype properties (`DP`/`Attribute`), no literal-valued commitments; without `ADT`s, literals lose typed interpretation; without `has`, feature-bearing, mereological, or compositional structure cannot be stated; without `is`, instantiation, specialisation, and classification-style links collapse. Together, these commitments make **OL2** *minimally complete* for the structural jobs **OL1** then specialises and **OL0** then instantiates—without, by itself, introducing the open-ended semantic multiplicity hypothesised for **OL1**.

Burden on "additional layers". Any proposed abstraction layer *beyond* the structural ontology layer (**OL2**; Section 6.1)—toward towers of meta-metaclasses, recursive powertypes, or `PowerType(PowerType(·))` patterns (Section 7)—requires *additional justification*: one should exhibit an *irreducible* modelling function that cannot be reduced to structural ontology primitives plus **OL1**/**AL1** semantic specialisation plus concrete resource composition (**OL0**). If no such function is shown, the addition may *often* be read as infrastructural organisation, taxonomy administration, modelling convenience, or tool-level structure—rather than as an ontological abstraction layer that *automatically introduces* new domain semantics.

Informally (workshop shorthand, not a formal definition of *Expr*):

$$Expr(\mathbf{OL2} + \mathbf{OL1} + \mathbf{OL0} + Add) \;>\; Expr(\mathbf{OL2} + \mathbf{OL1} + \mathbf{OL0})$$

should be required only when the strict inequality concerns *irreducible domain* expressivity; organisational gain or notation alone does not settle the case.

Consequence for "higher-order" MLM.. Some ostensibly higher-order constructions may stem from mixing semantic specialisation (**OL1**) with infrastructural stratification (**OL2**/**OL3**)—a reading aligned with Section 7's caution on stacked `PowerType` towers versus unary propagation paths.

6.2 Epistemological dualism

Two epistemic regimes coexist:

- **Structural representation** (**OL2**–**OL3**): grammar and possibility of modelling.
- **Semantic representation** (**OL1**–**OL0**): domain distinctions and meaning-bearing compositions.

OL2/**OL3** make modelling *possible*; **OL1**/**OL0** make meaning *usable* on instances and artefacts. Semantic multiplicity concentrates in **OL1**; additional towers beyond **OL2** may still add organisation, typing administration, or tooling—without guaranteeing fresh domain semantics.

Tractatus-style shorthand (stylistic only).

4.0 Ontology is organised here as four functional layers (**OL3**–**OL0**).
4.1 **OL3** contains `Resource`.
4.2 **OL2** contains structural ontology primitives.
4.3 **OL1** contains semantic specialisation and propagation paths.
4.4 **OL0** contains concrete resources and compositions.
4.5 Semantic multiplicity is hypothesised to concentrate in **OL1**.
4.6 Meta-towers beyond **OL2** do not necessarily generate new domain semantics.

Positioning. This is not an anti-MLM claim: classical MLM, MLT, FMMLx, and deep-instantiation stacks remain indispensable references [2, 10, 12, 13]. The suggestion is narrower: MLM may *benefit* from disciplined separation between abstraction layers and intra-layer specialisation levels—especially when aligning VAP/Δ-propagation with ontology infrastructure.

7 Discussion

No empirical results are reported. Evaluation may compare reduction of explicit intermediates against baselines, readability of unary traces for maintenance teams, semantic reuse across layers under potency constraints, qualitative modelling-effort indicators, redundancy metrics from comparable serialisations, and ontology-evolution traceability [13] without automation claims. Future work needs tighter formal semantics for transparency and blocking, propagation constraints and conflict handling when literals disagree, reference prototypes in conceptual editors, comparative studies with classical MLM formalisms [2], ontology-engineering pipelines into DL artefacts [4], and empirical protocols once tooling stabilises [7]. Conceptual editors could surface unary traces, graph visualisations of derivation depth, propagation toggles aligned with MLM stacks [13], and OWL exporter hooks [4]; companion notes cover derivational clusters and hybrid KG tooling [6, 7].

Classical MLM layering, reuse, and instantiation discipline stay anchored in [2, 13]; Deriver/VAP adds unary narration and propagation hooks in a complementary role [3]. That extension foregrounds

OL3	OL2	OL1	OL0
`Resource` *(one band)*	`Class`, OP, DP, ADT *(one band)*	domain, paths, Δ *(many levels)*	BoM / graphs *(struct. depth)*

Table 4: OL3–OL0 functional stack (hyp.) vs. AL propagation (Sec. 3).

readability and maintenance under change without redefining potency semantics or deep-instantiation theory [3, 12]. The note extends MLM with a property-centric question—which literals accompany derivations as Δ-properties, how they traverse **AL0**–**AL3**, and where materialised copies yield to propagating carriers—aligned with F6–F8 [12, 25].

MLT foregrounds instantiation semantics, type-order foundations, and powertype regularities [1, 10]; expressive MLM couples those commitments to OWL-shaped exports [9]. This workshop note foregrounds derivation-centric property semantics, readable unary traces, and reduced literal materialisation pressure alongside those foundations.

Expressive MLM surfaces regularities through explicit cross-level relations, compatibility tuples, opposite attributes, and synchronisation idioms [9, 12]. Where the derivation spine already fixes the narrative, part of those commitments may remain *visible* as propagated literals instead of duplicating parallel relational encodings [6]; unary chains may absorb recurring regularities without eliminating powertypes or associations [10]. That derivation-centric, property-centric emphasis targets redundant literal repetition along traces [25]; layered instantiation obligations remain authoritative [1].

Higher-order powertypes and the abstraction boundary of `Class`. Powertypes remain a productive MLM device [1, 10, 13]; the following is therefore offered only as a conceptual hypothesis, not as a theorem. A first-order reading such as `PowerType(car)` may still support *domain*-oriented commitments (manufacturer, homologation class, market segment, production period, body style): the powertype still behaves like a family-level abstraction one can argue about in the application domain.

By contrast, a second-order pattern such as `PowerType(PowerType(car))` becomes harder to motivate through genuinely *new* domain-specific data properties: many regularities are already carried at the **AL1** schema spine by attribute paths, propagation-oriented relations, and Δ-visibility along unary refinements (cf. Section 3). Additional higher-order layers may therefore contribute organisational structure, taxonomy administration, or tooling infrastructure rather than fresh ontology—*without* claiming that powertypes are useless overall.

Research question. Do higher-order powertypes correspond to genuine ontological distinctions, or do they mainly reify organisational structures already implicit in derivational semantic propagation paths?

One can *speculate*—provocatively but not dogmatically—that `PowerType(PowerType(X))` approaches the abstraction boundary of generic `Class` itself: first-order powertypes still resemble domain families, while higher orders increasingly collapse toward generic classification capability. Generic attributes such as `.Name`

are weak witnesses for that boundary: `.^natPerson` naturally supports `.Firstname` and `.Lastname`, yet a universal `.Name` is not automatically a stable, domain-independent semantic anchor across arbitrary specialisations. Domain-specific readings concentrate at **AL1**; stacked powertype machinery risks becoming purely infrastructural unless each layer is independently motivated in the domain.

Observation 7.1 (Meta-layer inflation). *When feature-shaped artefacts are reified across potency- or meta-object stacks without an independent domain rationale, bookkeeping for indices, slots, and synchronisation can dominate the narrative; unary-first propagation paths concentrate many commitments at **AL1**, which sharpens when higher orders still purchase irreducible semantics [12, 13].*

Potency fixes instantiation depth [3]; Δ-properties govern how literals propagate along refinements. OWL/DL remains the reference for classification [4, 19, 26]; unary traces precede axiomisation. FCA [14] and UFO [17, 18] remain reference frames. Concept repositories evolve through small unary refinements [13]; propagation supports traceability of change without claiming autonomous synthesis.

In FMMLx-style feature meta-modelling, explicit potency indices may govern where feature declarations or slot values are recorded [12]. Inserting an intermediate abstraction can, in explicit encodings, require shifting or reinterpreting indices above and below the insertion point; property values may then be repeatedly materialised at several levels. Clabject-oriented styles combine type and instance facets [2, 13]; potency, durability, and mutability may be annotated at multiple levels, and inserting a new intermediate class or changing effective depth may require re-numbering or re-annotating P/D/M values along the chain—maintenance-heavy annotation churn as depth and evolution grow. M-object approaches make meta-level information explicit via separate representatives [13]; under evolution they may require additional meta-objects, links, or synchronisation constraints, and slot or metadata encodings may be repeated across layers, which can contribute to meta-object proliferation and synchronisation overhead.

Consider a simple chain *a* is *b*. A later refinement may yield *a* is *c* is *b* after inserting an intermediate abstraction [24]. In potency- or deep-instantiation-heavy serialisations, such insertions can introduce non-local maintenance effects: edits local to one symbol may still oblige updates elsewhere to keep indices, slots, and meta-objects coherent. Potency-indexed and meta-object-heavy encodings therefore expose hierarchy-evolution sensitivity [13, 25].

Derivation-aware semantics with VAP/Δ-properties aim to keep selected literals semantically connected along derivational structures where policies permit, which may reduce repeated explicit re-materialisation for those carriers while explicit depth annotations remain mandatory wherever instantiation depth is first-class [3, 10]. Deriver/VAP stresses readable unary traces; Δ-properties propagate under declared policies; literal propagation does not by itself force global potency re-numbering, yet explicit depth is still required when depth must remain inspectable [1].

AL1 spine (unary + Δ)	**Powertype** stack (extra orders)
Readable literal/path lines at **AL1**.	Higher orders may reify implicit propagation groupings.

Table 5: AL1 spine vs. stacked `PowerType` orders (schematic).

Durability, mutability, and where the engineering pain sits. Do durability and mutability primarily solve intrinsic domain problems, or do they mainly manage complexity introduced by deep-instantiation-oriented modelling infrastructures? The question is intentionally provocative: once features are modelled as multi-level artefacts with lifecycle semantics, durability and mutability are natural companions [12, 13]. Yet much published exposition still foregrounds slot survival, meta-level synchronisation, re-indexing, and persistence of feature encodings across levels—concerns that can be *partly* orthogonal to irreducible domain laws and closer to infrastructure induced by feature reification and deep-instantiation bookkeeping. VAP is *not* put forward as a universal substitute for durability, mutability, or potency-first foundations [3, 10]; it instead raises an engineering-oriented simplification hypothesis: a substantial subset of practical cases may admit policy-governed *semantic accessibility* of literals along unary derivations, shifting attention from propagating feature-shaped objects across meta-levels toward controlling which literals remain readable where—without denying scenarios where explicit D/M annotations remain essential.

Figure 6 visualises FMMLx-, clabject-, and M-object-oriented idioms as a compact lens on the evolution scenario above, without judging potency-oriented accounts incorrect [1, 3].

Propagation policies remain partly heuristic; full formal semantics and benchmarks belong to future work. Δ-properties presuppose explicit unary traces and policy, not unconstrained subclass inheritance [4, 6, 25]. Prefixes are controlled surface syntax (Section 3). Scope and layer compatibility must be declared to avoid scope mismatches. Powertype consolidation (Section 4) coexists with established MLM [24].

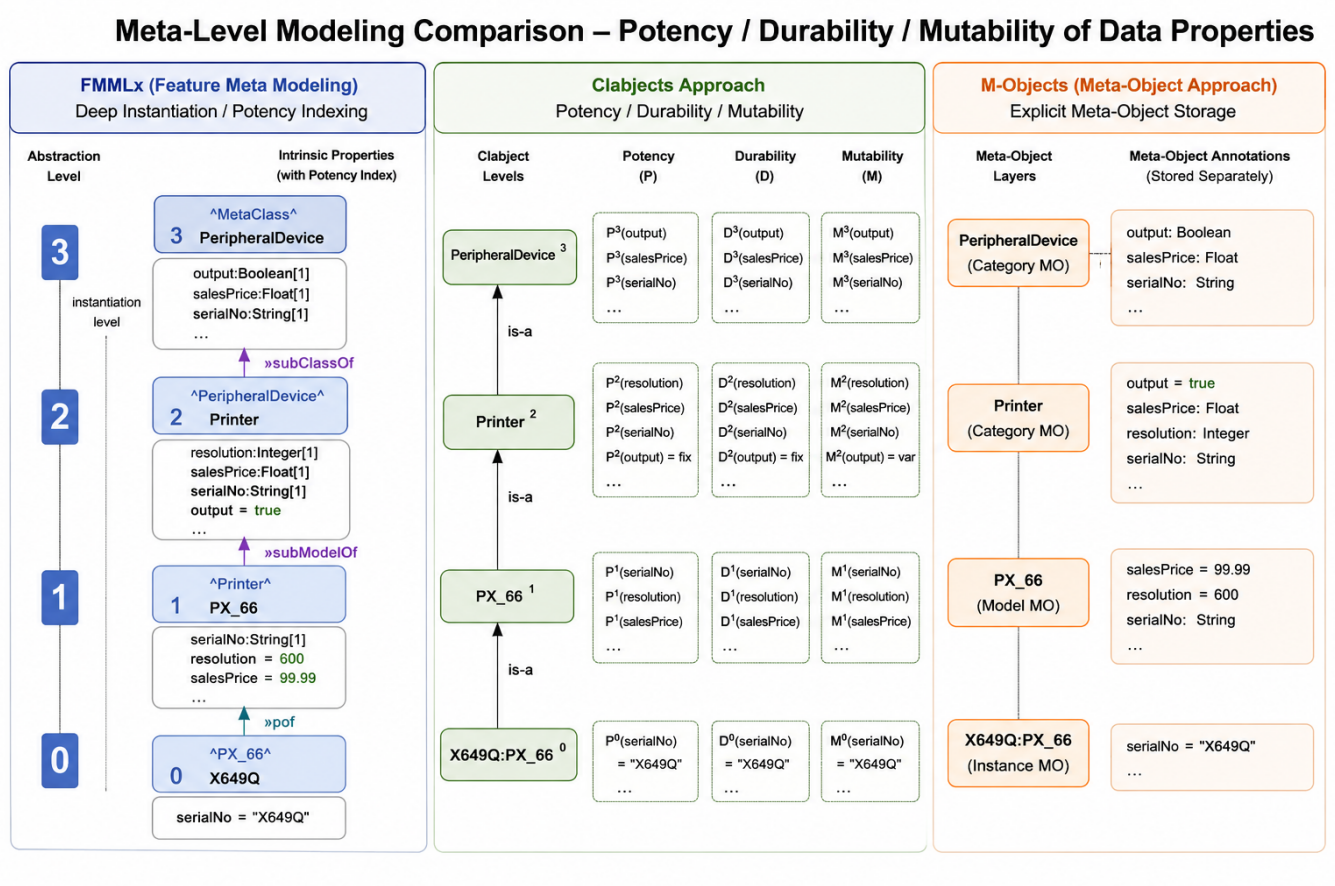


Figure 6: P/D/M evolution idioms vs. derivation-centred literal tracing.

Approach	Primary focus	Compositional semantics	Propagation	Mat./maint. pressure
Classical MLM	Levels, potency [2]	Typing / instantiation rules	Across layers	Medium–high (explicit intermediates)
OWL/DL	Axioms + reasoning [4]	DL constructors	Inherited assertions	Varies (axiom-encoded)
FCA	Lattices [14]	Intent–extent structure	Attribute implication	Low (implicit intents)
Potency / deep inst.	Potency, durability, mutability; clabjects [2, 13]	Deep inst. / potency-indexed	Potency/durability indexed features	Rises on insertion; re-index/slot-meta churn (domain vs infra entangled)
Deriver/VAP	Unary constructions [6]	$A(B)$ chains	Along derivations	Fewer literal copies if transparency fits; may ease *some* D/M re-annotation (not a potency substitute)

Table 6: Qualitative comparison (non-ranking).

8 Conclusion

This workshop note sketches *property-centric* propagation across **AL0**–**AL3** with F6–F8 emphasis [12]: PDP/MDP/UDP motivate literal accumulation; TDP/VAP may cut redundant re-materialisation where policies align [6, 25]. Propagated literals on unary traces can shrink parallel relational encodings in some idioms [9]. MLT and expressive MLM stay primary for instantiation-centric semantics [9, 10]; Deriver/VAP adds a propagation-oriented layer [5–7].

Cumulative scope. Readability and maintenance under evolution, without replacing MLT or OWL/DL [3, 4]; the propagation kit (Proposition 3.1) is operational and non-exhaustive; Δ-policies govern literal visibility on refinements, not universal per-individual DL inheritance [4, 6]. O4Top orients exports; Deriver supplies unary narratives [5, 6]; UFO and classical MLM rationales frame the stance [2, 13, 18].

Finally, Hypothesis 5.1 and Section 6 articulate the same exploratory normalization impulse in fuller form: layers versus levels, and the burden of justification on further strata.

Declaration on Generative AI

Generative AI tools were used interactively for editorial assistance, language refinement, restructuring support, and consistency checking. The scientific content, conceptual contributions, formal structures, examples, citations, and final manuscript were developed, reviewed, and validated by the author. The tools were not used as autonomous authors or as sources of scientific claims.

References

[1] João Paulo A. Almeida, Claudenir M. Fonseca, and Victorio A. Carvalho. 2017. A Comprehensive Formal Theory for Multi-level Conceptual Modeling. In *Proceedings of the 36th International Conference on Conceptual Modeling (ER 2017) (Lecture Notes in Computer Science, Vol. 10650)*. Springer, Cham, 280–294. doi:10.1007/978-3-319-69904-2_23

[2] Colin Atkinson and Thomas Kühne. 2001. The Essence of Multilevel Metamodeling. In *Proceedings of the 4th International Conference on The Unified Modeling Language (UML 2001) (Lecture Notes in Computer Science, Vol. 2185)*. Springer, Berlin, 19–33. doi:10.1007/3-540-45441-1_3

[3] Colin Atkinson and Thomas Kühne. 2003. Model-Driven Development: A Metamodeling Foundation. *IEEE Software* 20, 5 (2003), 36–41.

[4] Franz Baader, Diego Calvanese, Deborah L. McGuinness, Daniele Nardi, and Peter F. Patel-Schneider. 2010. *The Description Logic Handbook*. Cambridge University Press.

[5] Hermann Bense. 2014. The Unique Predication of Knowledge Elements and their Visualization and Factorization in Ontology Engineering. In *Formal Ontology in Information Systems: Proceedings of the Eighth International Conference (FOIS 2014)*. IOS Press, Amsterdam, 241–250. doi:10.3233/978-1-61499-438-1-251

[6] Hermann Bense. 2024. How to Build a Controlled Vocabulary of Concepts for Deep Semantic Search. In *Proceedings of the 2024 Intelligent Systems Conference (IntelliSys) (Lecture Notes in Networks and Systems)*, Kohei Arai (Ed.). Springer, 143–160.

[7] Hermann Bense, Sebastian Lothary, and Matthias L. Hemmje. 2026. A-LLM: Definitional Primitives and a Transformation Calculus for Semantic Quality in Hybrid Large Language Model–Knowledge Graph Systems. In *Proceedings of the 12th Intelligent Systems Conference (IntelliSys 2026)*. Amsterdam, The Netherlands. Companion work: axiomatic A-LLM framework, FoC, RDF/OWL mapping, evaluation.

[8] Hermann J. Bense. 2023. Streamlining Conceptual Modeling. In *Intelligent Systems Conference (IntelliSys) (Lecture Notes in Networks and Systems, Vol. 822)*. Springer Nature, Cham, 326–344. doi:10.1007/978-3-031-47721-8_22

[9] Freddy Brasileiro, João Paulo A. Almeida, Victorio A. Carvalho, and Giancarlo Guizzardi. 2016. Expressive Multi-level Modeling for the Semantic Web. In *The Semantic Web — ISWC 2016, Part I (Lecture Notes in Computer Science, Vol. 9981)*. Springer, Cham, 53–69. doi:10.1007/978-3-319-46523-4_4

[10] Victorio A. Carvalho and João Paulo A. Almeida. 2018. Toward a well-founded theory for multi-level conceptual modeling. *Software & Systems Modeling* 17, 1 (2018), 205–231. doi:10.1007/s10270-016-0538-9

[11] Charles J. Fillmore. 1982. Frame Semantics. In *Linguistics in the Morning Calm*. Hanshin Publishing, 111–137.

[12] Claudenir M. Fonseca, João Paulo A. Almeida, Giancarlo Guizzardi, and Victorio A. Carvalho. 2021. Multi-level conceptual modeling: Theory, language and application. *Data & Knowledge Engineering* 134 (2021), 101894. doi:10.1016/j.datak.2021.101894

[13] Ulrich Frank. 2022. Multi-level modeling: cornerstones of a rationale. *Software and Systems Modeling* 21 (2022), 451–480. doi:10.1007/s10270-021-00955-1

[14] Bernhard Ganter and Rudolf Wille. 1999. *Formal Concept Analysis: Mathematical Foundations*. Springer, Berlin.

[15] Peter Gärdenfors. 2000. *Conceptual Spaces: The Geometry of Thought*. MIT Press.

[16] Peter Gärdenfors. 2014. *The Geometry of Meaning*. MIT Press.

[17] Giancarlo Guizzardi. 2005. *Ontological Foundations for Structural Conceptual Models*. University of Twente, Enschede.

[18] Giancarlo Guizzardi, Alessander Botti Benevides, Claudenir M. Fonseca, Daniele Porello, João Paulo A. Almeida, and Tiago Prince Sales. 2022. UFO: Unified Foundational Ontology. *Applied Ontology* 17, 1 (2022), 167–210. doi:10.3233/AO-210256

[19] Ian Horrocks, Peter F. Patel-Schneider, and Frank van Harmelen. 2003. From SHIQ and RDF to OWL: The making of a Web Ontology Language. *Journal of Web Semantics* 1, 1 (2003), 7–26.

[20] Matthias Jarke, Ralf Gallersdörfer, Manfred A. Jeusfeld, and Martin Staudt. 1995. ConceptBase—a deductive object base for meta-data management. *Journal of Intelligent Information Systems* 4, 2 (1995), 167–188.

[21] Barbara Liskov and Stephen Zilles. 1974. Programming with Abstract Data Types. In *Proceedings of the ACM SIGPLAN Symposium on Very High Level Languages*. ACM, New York, NY, USA, 50–59.

[22] John Mylopoulos, Alex Borgida, Matthias Jarke, and Manolis Koubarakis. 1990. Telos: Representing Knowledge About Information Systems. *ACM Transactions on Information Systems* 8, 4 (1990), 325–362. doi:10.1145/102833

[23] Bernd Neumayr, Katharina Grün, and Michael Schrefl. 2009. Multi-Level Domain Modeling with M-Objects and M-Relationships. In *Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modelling (APCCM 2009) (Conferences in Research and Practice in Information Technology, Vol. 96)*. Australian Computer Society, Darlinghurst, Australia, 107–116.

[24] Bernd Neumayr, Michael Schrefl, and Bernhard Thalheim. 2011. Modeling Techniques for Multi-level Abstraction. In *The Evolution of Conceptual Modeling*, Roland Kaschek and Lois Delcambre (Eds.). Lecture Notes in Computer Science, Vol. 6520. Springer, Berlin, Heidelberg, 68–92. doi:10.1007/978-3-642-17505-3_4

[25] Alain Pirotte, Esteban Zimányi, David Massart, and Tatiana Yakusheva. 1994. Materialization: A Powerful and Ubiquitous Abstraction Pattern. In *Proceedings of the 20th International Conference on Very Large Data Bases (VLDB)*. Morgan Kaufmann, Santiago de Chile, 630–641.

[26] W3C. 2012. OWL 2 Web Ontology Language Document Overview. https://www.w3.org/TR/owl2-overview/.