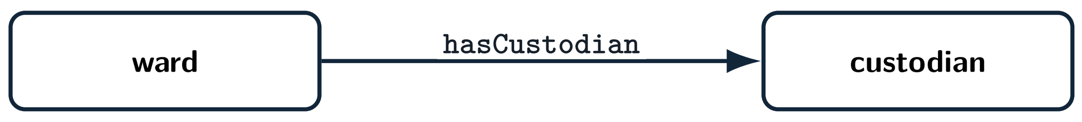
A PAMoLa Solution to the MULTI 2026 Vector Industry Challenge

MULTI 2026 Vector Industry Challenge

Shilpi Gupta, Mohammad Sadeghi, Monalisha Ojha, Colin Atkinson - Software Engineering Group, University of Mannheim, Mannheim, Germany | {shilpi.gupta, mohammad.sadeghi, monalisha.ojha, colin.atkinson}@uni-mannheim.de

1. What is PAMoLa?

PAMoLa is a **level-agnostic, self-describing modeling substrate** in which domain objects, classifiers, values, profiles and rules all live in one graph as **ordia**. Even the small protected `Ordium.Core` clade is *loaded as model content* rather than compiled into the tool. Among other things, PAMoLa supports the **multi-level modeling style** embodied by Melanee's LML. It therefore **separates two classification dimensions:** *Linguistic* classification is fixed by the Core (`Entity`, `Attribute`, `Marker`, ...), whereas *ontological* classification remains soft and unbounded. A domain type, its type and its instances are all ordinary content, so adding a classifier requires neither a schema change nor generated classes or redeployment.



exactly one, forming a rooted, acyclic tree

at most one (0..1), with potency reduced by one

The figure shows the two most important edge kinds used on this poster. **Every ordium has exactly one custodian**, which makes custody a rooted, acyclic tree, while a *ward* is simply an ordium held by a custodian. PAMoLa is **classification-style agnostic** because `hasDeclaredType`, `hasPotency`, `hasDurability` and `hasMutability` are **stored uninterpreted**. A profile defines how they are interpreted. For example, the **LML profile** interprets them as potency-stepped classification and feature projection, while another multi-level discipline could impose different rules over the same PAMoLa graph.

2. Existing PAMoLa and LML facilities satisfy most challenge conditions

Construct	Feature	Vector Challenge Compliance
`hasCustodian`	single-valued, rooted, acyclic	A part has **exactly one** owner. Shared or circular ownership is *unrepresentable*, which keeps the owner context well defined.
`hasDeclaredType`	single-valued, optional (0..1)	**Challenge Fig. 11** rules out the *invalid sharing of a part instance*. Since a part cannot have two declared types, no additional Vector rule is required. A part at the top of its chain may remain untyped, which matches the 0..1 multiplicity in challenge Fig. 3. Strictness is therefore *local* by construction.
hasPotency		Classification cannot skip or invert levels. One integer represents type, prototype and instance as potencies 2/1/0.
`hasDurability/` `hasMutability`		**Challenge §2.1.1** value projection along STO chains and the encouraged refinement are each represented by one integer. hasDurability determines how far an attribute is projected. hasMutability determines how long its value may be refined: 0 fixes the value, while n permits refinement during the next n instantiation steps.
`Categorisation`		**Challenge §2.1.1** kind admissibility is structural. Sealedness is a *kind* rather than a stored value that must be cross-checked, so an inadmissible pairing cannot be represented.

This separation matters because only *custody* is defined universally in the core. Potency stepping, single-valued typing and feature projection are all **LML profile** content over uninterpreted core properties. PAMoLa therefore meets Vector's requirements by selecting a classification discipline rather than hard-wiring one.

The Vector-specific extension therefore consists only of **congruence** and three bijection cases.

3. The one law that remains

Composition is represented by `hasCustodian`, while structured typeOf is represented by `hasDeclaredType`. A Type-Square is well formed when these two relations *commute*.

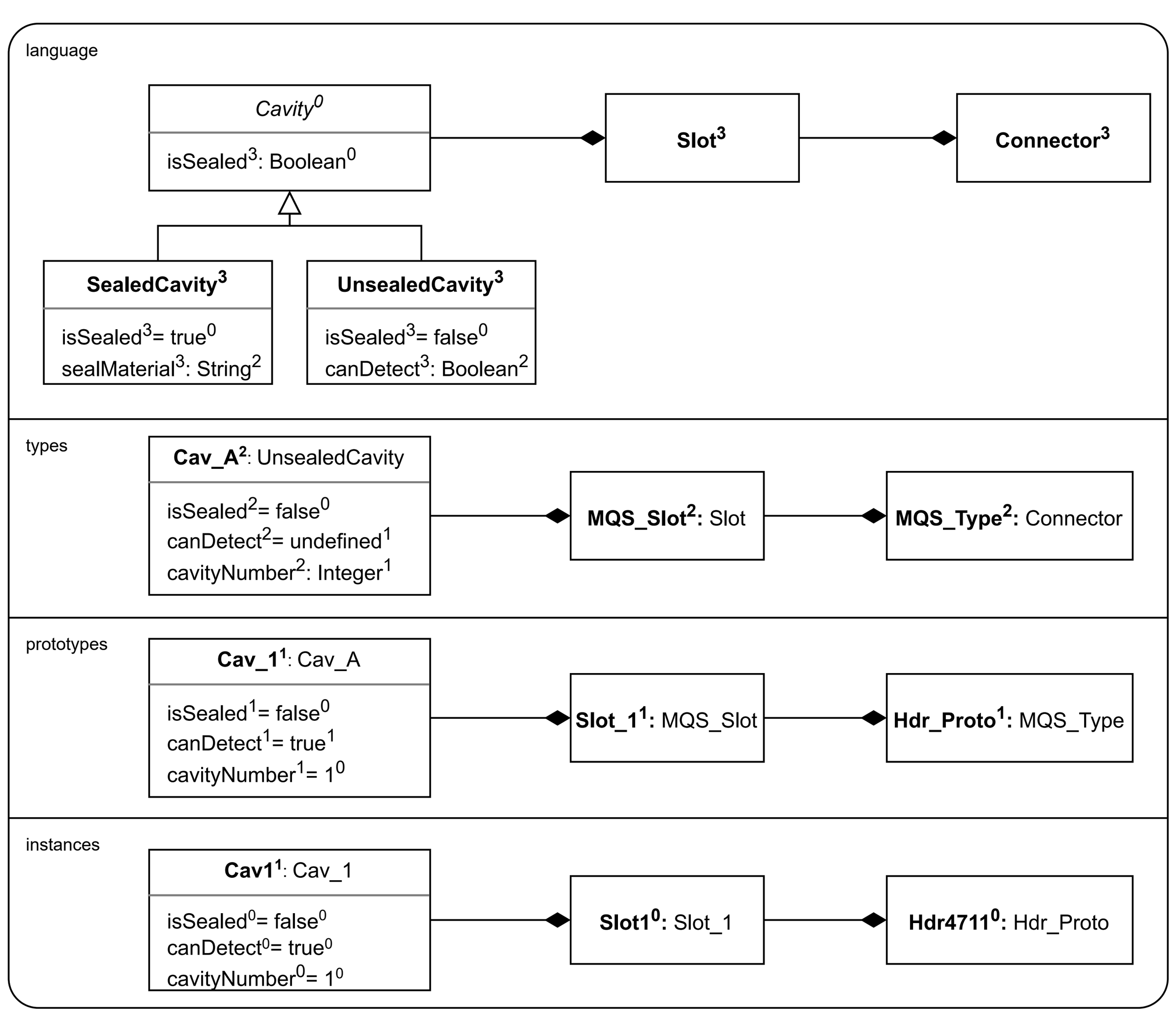
$$
\begin{tikzcd}
\begin{array}{c} C \\ \textbf{the whole} \end{array} \arrow[r, "\texttt{hasDeclaredType}"] & \tau(C) \\
\begin{array}{c} x \\ \textbf{a part of } C \end{array} \arrow[u, "\texttt{hasCustodian}"] \arrow[r, "\texttt{hasDeclaredType}"'] & \tau(x) \arrow[u, "\texttt{hasCustodian}"']
\end{tikzcd}
$$

$$\kappa(\tau(x)) = \tau(\kappa(x))$$

The custodian of a ward's declared type is the declared type of that ward's custodian.

`hasCustodian` is the *universal* spine because it represents both part-of and lives-in. We therefore restrict the law to `componentWard`, which is derived from a `WardRole.component` marker on the language concepts and inherited along both classification and specialisation. **Two markers are enough to guard the complete model.**

4. The solution as an LML projection



Reading down the cavity column shows that `isSealed` is declared ***once*** as $\langle isSealed^3 : Boolean^0 \rangle$. Durability 3 projects it across all four levels, whereas mutability 0 prevents any lower-level change. Vector's Fig. 4 repeats the same attribute **eleven times** and then requires a constraint to keep the copies consistent.

5. The PAMoLa solution (HUTN FHAS)

This clade corresponds to the *complete* Vector metamodel in challenge Fig. 4. Its nesting directly represents `hasCustodian`.

```
EN {hasName ConnectorLanguage,
  EN {hasName Connector, hasPotency 3,
    EN {hasName Slot, hasPotency 3,
      MK {hasName componentWard, hasTag Ordium.Core.WardRole.component, refersTo Slot},
      EN {hasName Cavity, hasPotency 0,
        AT {hasName isSealed, hasDataType Boolean, hasDurability 3},
        MK {hasName componentWard, hasTag Ordium.Core.WardRole.component, refersTo Cavity}
      },
      EN {hasName UnsealedCavity, hasPotency 3,
        AT {hasName isSealed, hasDataType Boolean, hasValue false, hasDurability 3},
        AT {hasName canDetect, hasDataType Boolean, hasDurability 3, hasMutability 2}
      },
      EN {hasName SealedCavity, hasPotency 3,
        AT {hasName isSealed, hasDataType Boolean, hasValue true, hasDurability 3},
        AT {hasName sealMaterial, hasDataType String, hasDurability 3, hasMutability 2}
      },
      CA {hasName CavityKinds, isDisjoint true, isComplete true,
        IC {hasName general, isGeneralisation true, hasRelatum Cavity},
        IC {hasName unsealed, hasRelatum UnsealedCavity},
        IC {hasName sealed,   hasRelatum SealedCavity}
      }
    }
  }
}
```

6. The remaining rules

```
% custody carries part-of AND lives-in, so guard the law.
% componentRole is derived from WardRole.component markers
% on Slot and Cavity and inherited ...
% down BOTH the classification and specialisation axes.
componentWard(?X, ?C) :- hasCustodian(?X, ?C),
    componentRole(?X).

typedWardPair(?X, ?T, ?C, ?TC) :-
    componentWard(?X, ?C), hasDeclaredType(?X, ?T),
    hasDeclaredType(?C, ?TC).

% CONGRUENCE -- challenge Fig. 6, 7
violation(?X, wardTreeCongruence) :-
    typedWardPair(?X, ?T, ?C, ?TC),
    ~hasCustodian(?T, ?TC).

% BIJECTION -- challenge Fig. 8, 9, 10
usedWard(?T, ?C) :- typedWardPair(_, ?T, ?C, _).

violation(?X, wardTypeMissing) :-        % totality Fig. 8
    componentWard(?X, ?C), hasDeclaredType(?C, _),
    ~hasDeclaredType(?X, _).

violation(?T, wardInstanceMissing) :-    % surjectivity Fig. 9
    componentWard(?T, ?TC), hasDeclaredType(?C, ?TC),
    ~usedWard(?T, ?C).

violation(?T, wardTypeShared) :-         % injectivity Fig. 10
    typedWardPair(?X1, ?T, ?C, _),
    typedWardPair(?X2, ?T, ?C, _), ?X1 != ?X2.
```

PAMoLa's internal Datalog engine evaluates these rules directly on the clade, so no export or translation is required.

The Vector-specific extension omits rules already supplied by the LML profile. The LML profile interprets `hasDurability` and `hasMutability` to govern value projection and refinement, while its `categorisation` rules govern kind admissibility and declaredTypeSingleValued enforces functionality (Fig. 11).

No kind index is needed because a ward's kind is *derived* from its type chain rather than stored beside it. Pairing a cavity instance with a prototype from the wrong cavity kind is therefore unrepresentable, which keeps the bijection rules straightforward.

7. The resulting model size

Since Vector's Fig. 4 is a metamodel, we compare it with the metamodel-level part of the PAMoLa solution, namely the ConnectorLanguage clade shown in Box 5.

	Vector Fig. 4	PAMoLa
classifiers (`EN`)	23	6
relationships (`RE`)	37	**0**
participations (`PA`)	74	**0**
aggregation markers (`MK`)	17	2
attributes (`AT`)	29	5
categorisations / intercalations	0	1 / 3
total ordia	**180**	**17**

The duplicated families. Figure 4 gives `Header` and `WiringConnector` structurally identical instance families that type onto the same prototypes. The source for Fig. 4 merely appends `W` to the second family (`SlotAInstance` and `SlotAInstanceW`). Because Fig. 4 must encode both connector families within a two-level metamodel, it cannot represent them there as two instances of the same prototype. It therefore duplicates eight classifiers and six associations. PAMoLa instead uses two potency-0 clabjects that share the same declared type.

Of Vector's 37 relationships, 17 encode composition and 20 emulate type links. PAMoLa represents them through `hasCustodian` and `hasDeclaredType`, respectively, so no relationship ordia are required.

8. Answers to the challenge questions

Aspect	Answer
(a) Fundamental relationships	The fundamental relations are ontological *classification* (`hasDeclaredType` + `hasPotency`), *custody* (`hasCustodian`) and *specialisation* (`Categorisation`). STO is classification structured by custody rather than Type-Object emulation.
(b) Strict or non-strict?	*Locally strict.* When a ward is typed, its component-ward structure is checked against that of its type, although a chain-top ward may remain untyped (0..1). Adjacent STO levels are strict, while depth and non-compositional edges remain unconstrained.
(c) Vital constructs	`hasCustodian`, `hasDeclaredType`, `hasPotency`, `hasDurability`/`hasMutability`, `Categorisation`, and rules-as-content.
(d) Integrity conditions	Congruence and ward bijection are checked on load, before an edit transaction is committed, after repair and on demand. Projection, refinement and kind admissibility are already governed by the LML profile and therefore require no additional Vector-specific checks.

We separate diagnosis from repair because the challenge gives types no priority over instances. The rules therefore identify a broken square without assigning blame, after which an explicit policy determines the repair.

9. Assumptions and trade-offs

Assumption / trade-off	Consequence
Composite ownership is *exclusive*	Despite the 0..* owner-end multiplicity in Figure 3, we treat the filled diamond as authoritative. Shared ownership would make custody inapplicable.
A/B/C are placeholder identifiers	Section 2.1 of the challenge identifies A, B and C as placeholders. The A- and B-families are therefore clones rather than distinct concepts.
Custody fixes only the basic one-to-many cardinality	Each ward has exactly one custodian, while a custodian may have any number of wards. Any tighter lower or upper bound, such as "at most 12 cavities", must be expressed by a rule.
Not every domain relation is custody	Mating relationships, electrical connections and trace links are *not* compositions. They remain ordinary relationships layered over the ward spine.

10. Take-away

The challenge encodes a three-level domain in a two-level tool.

Connectors have three ontological levels, namely *type*, *prototype* and *instance*, whereas EMF provides only metamodel classes and model objects. The third level must therefore be **emulated**. **Type-Object** replaces classification with an ordinary `type` association, and each ⟨concept × level × family⟩ combination requires its own classifier. **This expands the connector, slot and cavity structures into 23 classifiers.**

PAMoLa represents these levels natively. Potency distinguishes the levels, custody represents the parts, durability projects the features, and mutability controls where their values may be refined. Consequently, only **commutativity** and **bijection** need to be stated.

Because the composition-preservation law is **parametric in the relation**, the same pattern can also be reused wherever a hierarchy must preserve a corresponding part structure. Applied to `hasDeclaredType`, it operates across ontological levels; applied to specialisation, it operates within a level.

[1] Lange, Reichmann, Kühne, Maier, Weber. *MULTI Vector Industry Challenge*. MODELS Companion, 2026. [2] Atkinson, Kühne. *Reducing accidental complexity in domain models*. SoSyM 7(3), 2008. [3] PAMoLa. 2026. URL: https://pamola.net.