Ensuring Consistency via Deep Composition Semantics

A Contribution to the MULTI Vector Industry Challenge

Thomas Kühne
Victoria University of Wellington

Arne Lange
Karlsruhe Institute of Technology

Challenge: Ensuring structural integrity

Enforcing STO Structure

The challenge does not mandate a specific relationship between ConnectorType, ConnectorPrototype, and ConnectorInstance. Hence, ignoring the resulting semantic tension implied by the current concept names, we chose classification as the relationship between them to support *feature projection*.

Due to our use of the OCA, there is no need to specify a linguistic type model like the one in Fig. 4 of the challenge description. The only equivalent elements are “CavityType” at the top right, so that the type of the “isSealed” property can be defined, and the attribute definitions in “CavityB_Prototype” (name taken from the challenge description) so that they can be inherited by any type needing them. The diagram to the right therefore shows an application (at the “M_1-level”) of the linguistic type model, i.e., a hypothetical modeling scenario, as indicated by the “X15” choices for connector-, slot-, and cavity-concepts. We chose a traditional layout that orients the classification vertically, leading to a 90° anti-clockwise rotation compared to Fig. 4 of the challenge description.

Note that we use a predefined “CavityB_Prototype” type to introduce the “cavityNumber” and “canDetect” properties via attributes. Alternatively, we could have introduced the attributes at “CavityTypeX15”, which may be desirable, as it supports further customization at the application level (rather than the language definition). Modelers have a choice which approach to prefer and could even instantiate elements like “CavityTypeX15” from a predefined type that specifies deep attributes. This would require multiple classification support, though.

The fourth option of introducing attributes such as “isSealed”, “cavity-Number”, and “canDetect” uses MDMLM, i.e., orthogonal ontological classification [4]. Instead of using a deep type with a deep dual filed (cf. “Style A” in the diagram to the right), this approach uses a shallow type that directly classifies all entities needing the property (cf. “Style B” in the diagram). For brevity, we directly specified the required property value (“false”) here, but of course it would be possible to separate the attribute declaration and the value specification.

Bijection Requirement for Parts

- The bijection requirement as such is straightforwardly achieved by using black-diamond composition.
  1. no sharing of wholes or parts is possible
  2. wholes and specified parts are non-optional
- Ensuring the STO structure is repeated for parts, in the sense that instances are constrained to have mirroring parts, is achieved by using deep connectors (cf. the purple instanceOf relation-ships in the diagram).

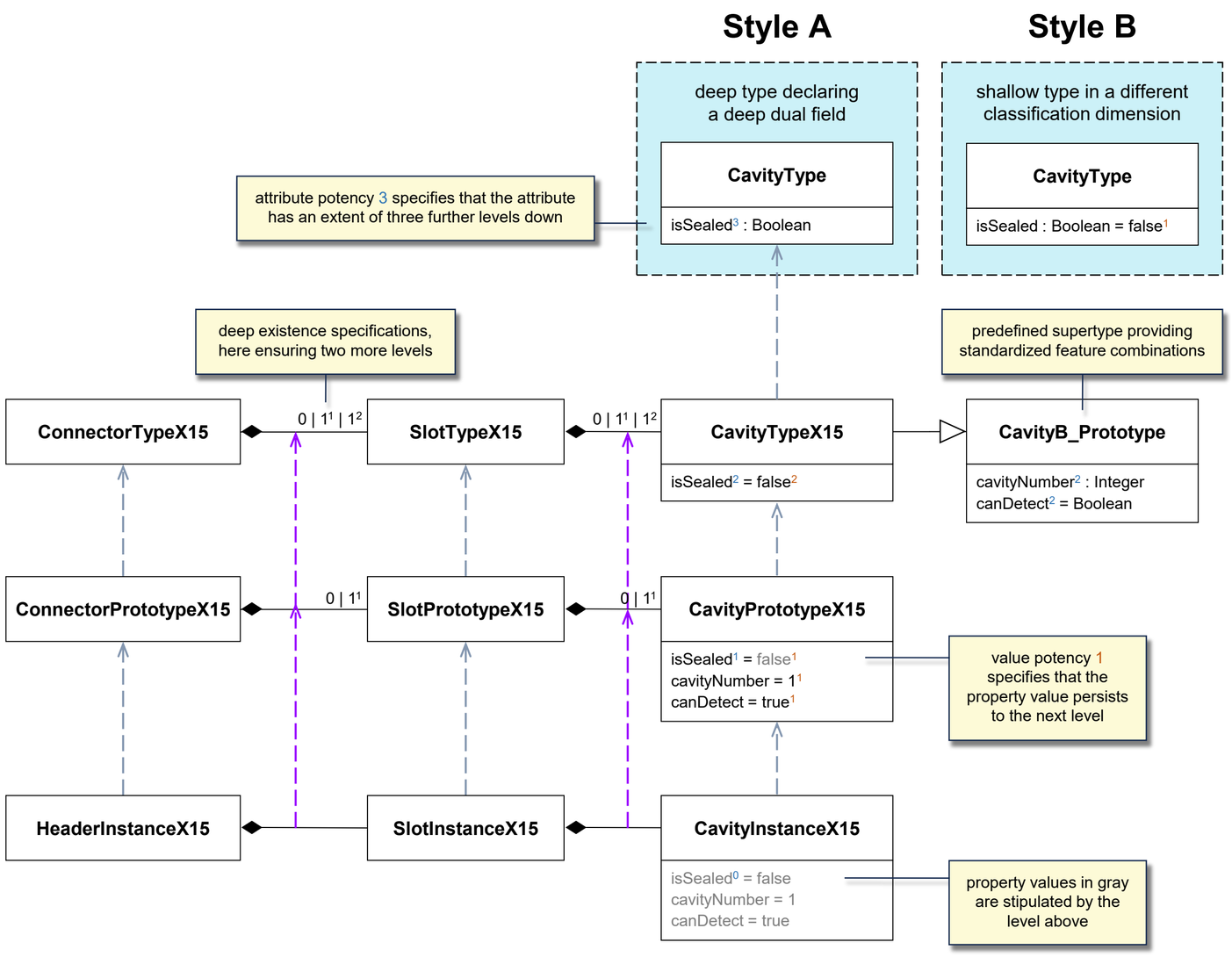
  We are using the “Style A” approach again; “Style B” would also be available.

  Ensuring that all required deep specifications exist could be achieved by instantiating the top level concepts from types that require the com-position relationship to exist.

  Congruence of the composition structure can be enforced by creating quadruples of type/instances pairs along with their parts, using a type template of the form shown in the red rectangle of Fig. 3 in the challenge description.

Technological Basis

- Our solution is built on the *Orthogonal Classification Architecture* **[1]**
  - a linguistic type level supports user-defined ontological types
- We assume *deep attributes* support, including deep mutability **[2]**
  - attribute presence can be stipulated across multiple levels and the visibility of dual field values can be extended to lower levels
- We assume *deep connector* support **[3]**
  - relationship presence can be stipulated deeply across multiple levels
- We optionally employ *Orthogonal Ontological Classification* **[4]**
  - elements may be multiply classified from different classification dimensions
- We assume support for *composition semantics*
  - regular lifetime synchronization between a whole and its composed parts, and exclusive ownership of parts.



Trade-Off Analysis

- Classification relationships between Connector types, prototypes, and instances may not be onto-logically justifiable, depending on ultimate interpretation and naming. Pragmatically, our choice works very well in terms of straightforwardly supporting *feature projection* (value propagation) and ensuring the bijection requirements for parts. However, whether “prototype” is an appropriate intermediate concept name when using this approach is worth considering.
- Feature projection is possible but optional. Value overriding/refinement is supported, as long as new values have the same type. The deep dual field approach (“Style A”) only supports contiguous ranges of constant values, whereas the MDMLM approach allows freely designating elements; however, then relying on modeler responsibility to achieve contiguous ranges.
- The absence of an explicit linguistic type model (cf. Fig. 4 of the description) affords modelers with more flexibility. Yet, the same integrity guarantees can be achieved through various options as outlined above.
- Introducing predefined attribute sets as suggested in the diagram (cf. “CavityB_Prototype”) via inheritance may be unfamiliar at first to modelers without respective experience. It does avoid the need for multiple classification, though, and is only one of several ways to inject required attributes.
- Constraints are only required in the form of type invariants, in case it is necessary to make the use of certain concepts (e.g., a sealed cavity) contingent on the value of properties.
- We acknowledge that we assume comprehensive technological support and that there is currently no tool that implements all of it simultaneously. All concepts involved, however, have been independently proposed before and we are unaware of any roadblocks that may stand in the way of implementing the entirety of the required support.

  In any event, the result is a solution that not only supports the requirements but does so in a clean and concise manner, with minimal need for hand-written constraints.

References

[1] C. Atkinson and T. Kühne, *“Model-Driven Development: A Metamodeling Foundation”*, IEEE Software, vol. 20, no. 5, pp. 36–41, 2003

[2] T. Kühne, J. P. A. Almeida, C. Atkinson, M. Jeusfeld, G. Mezei, “*Field Types for Deep Characterization in Multi-Level Modeling*”, MODELS 2023 Companion, MULTI 2023, pp 639-648, 2023

[3] C. Atkinson, R. Gerbig, T. Kühne, *“A Unifying Approach to Connections for Multi-Level Modeling”*, Proceedings of MODELS 2015, pp. 216-225, IEEE, 2015

[4] T. Kühne, *“Multi-Dimensional Multi-Level Modeling”*, Software and Systems Modeling, Vol. 21, pp. 543-559, January 2022