27th International Conference on
Model Driven Engineering Languages and Systems

MODELS 24

22 - 27 September 2
Linz , Austria

Misconceptions about Potency-Based Deep Instantiation

Colin Atkinson, Thomas Kühne and Arne Lange

UNIVERSITÄT MANNHEIM

VICTORIA UNIVERSITY OF WELLINGTON
TE HERENGA WAKA

KIT
Karlsruhe Institut für Technologie

MULTI 2024 - 11th International Workshop on Multi-Level Modeling

Tuesday 24 September 2024

Linz, Austria

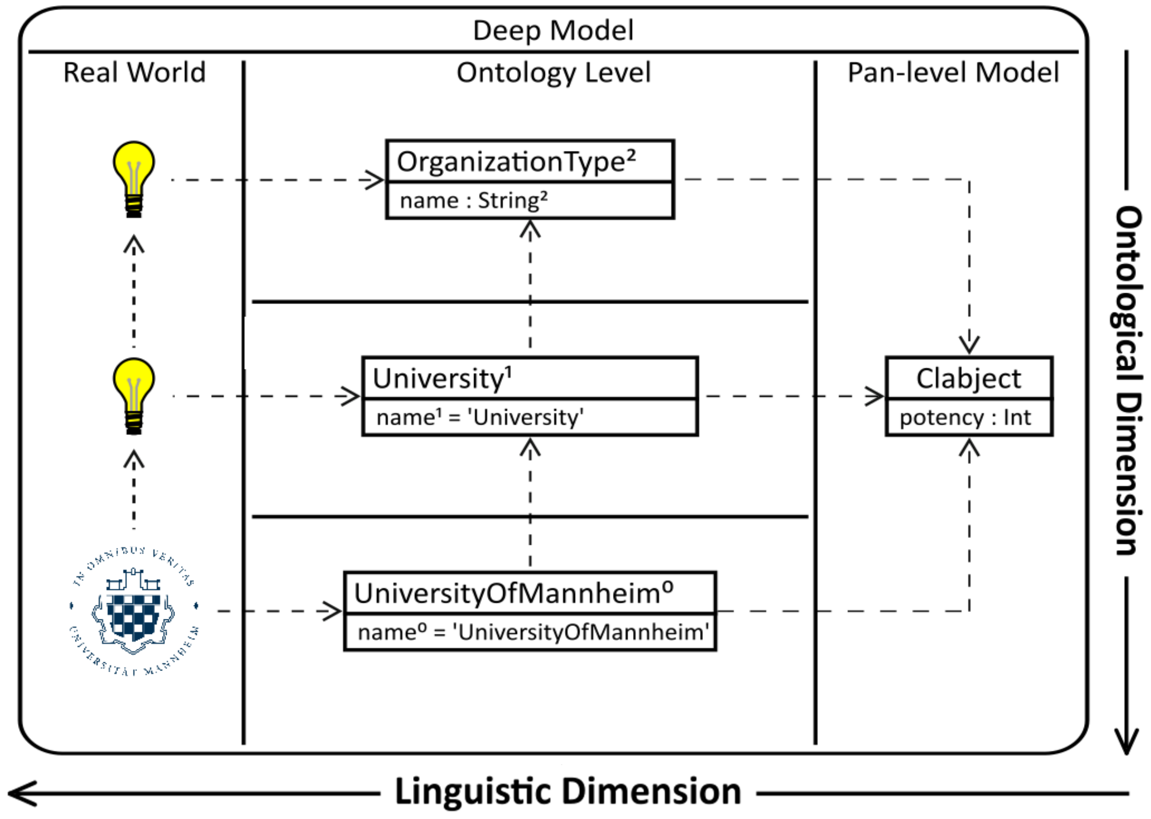
Potency-Based Deep Instantiation

Basic Principles

- an instance must be at the immediate level below its type
- specialization relationships must not cross level-boundaries
- every clabject has a potency which is a non-negative integer
- the potency of a clabject must be lower than that of its direct type
- the potency of a field (a.k.a. durability) must be one less than that of the corresponding field of the owning clabjects's direct type

Scope

- Classic deep instantiation (not including approaches that use different level segregation principles and/or potency mechanisms)



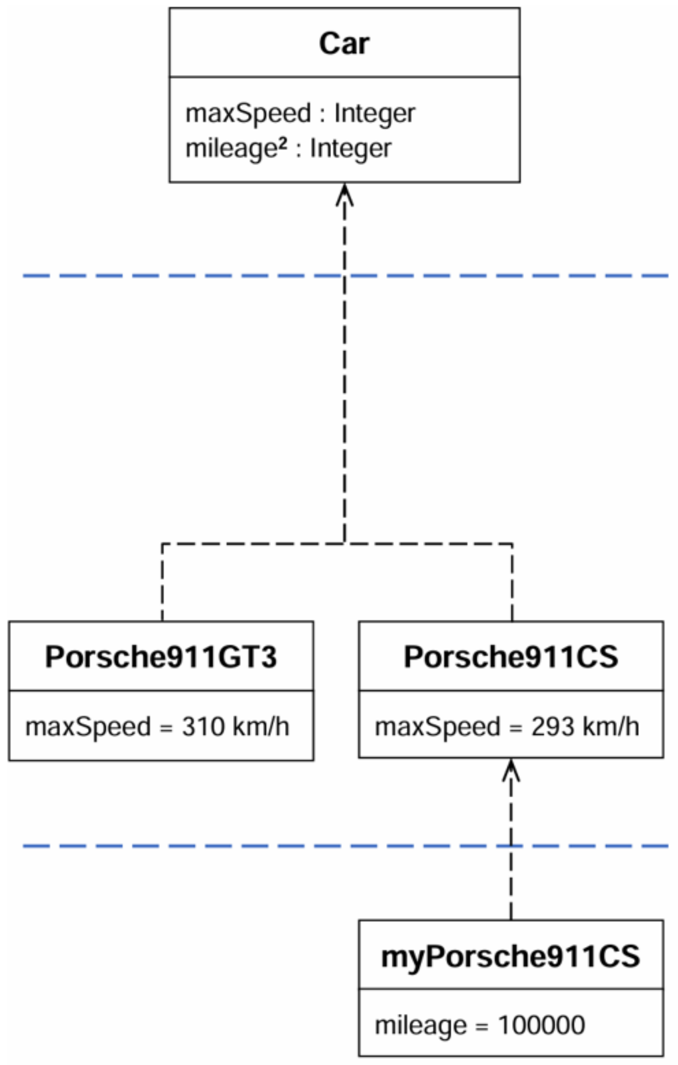
Inflexibility

Criticism

- *“Additional abstraction levels for some domain concepts cannot be introduced without requiring global model changes”*

Example

- Want to add a concept of intermediate specificity
  - **Porsche911**



Claimed Starting Model

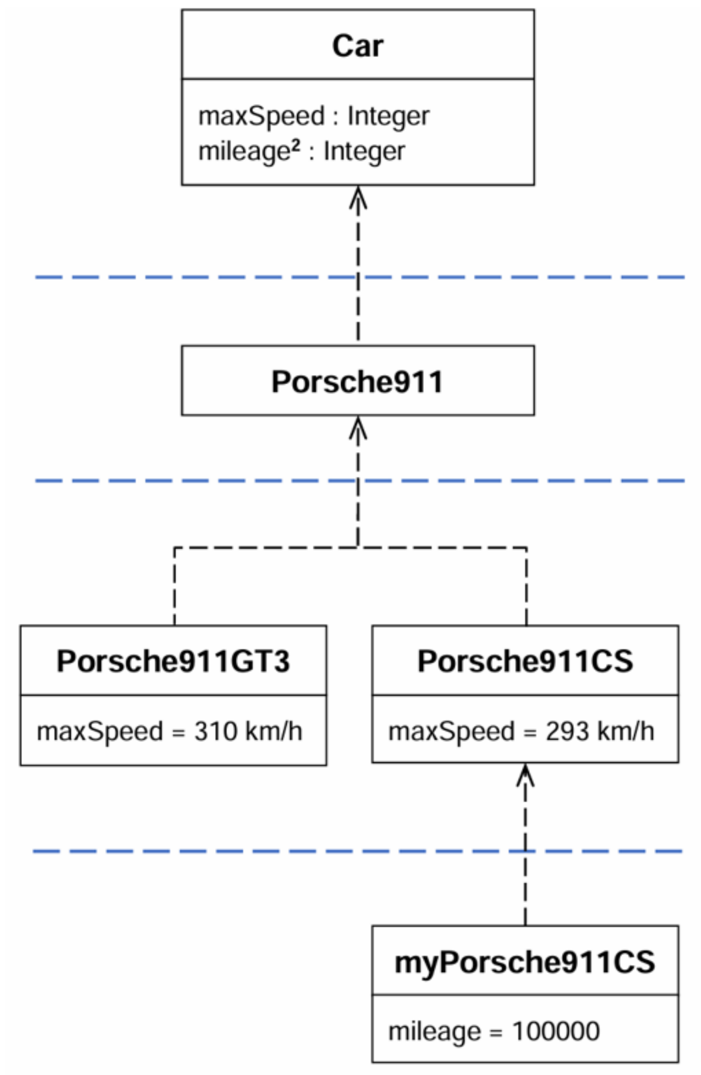
Inflexibility

Criticism

- *"Additional abstraction levels for some domain concepts cannot be introduced without requiring global model changes"*

Example

- Want to add a concept of intermediate specificity
  - **Porsche911**
- Claim is that a new classification level is needed



Claimed Solution

Inflexibility

Criticism

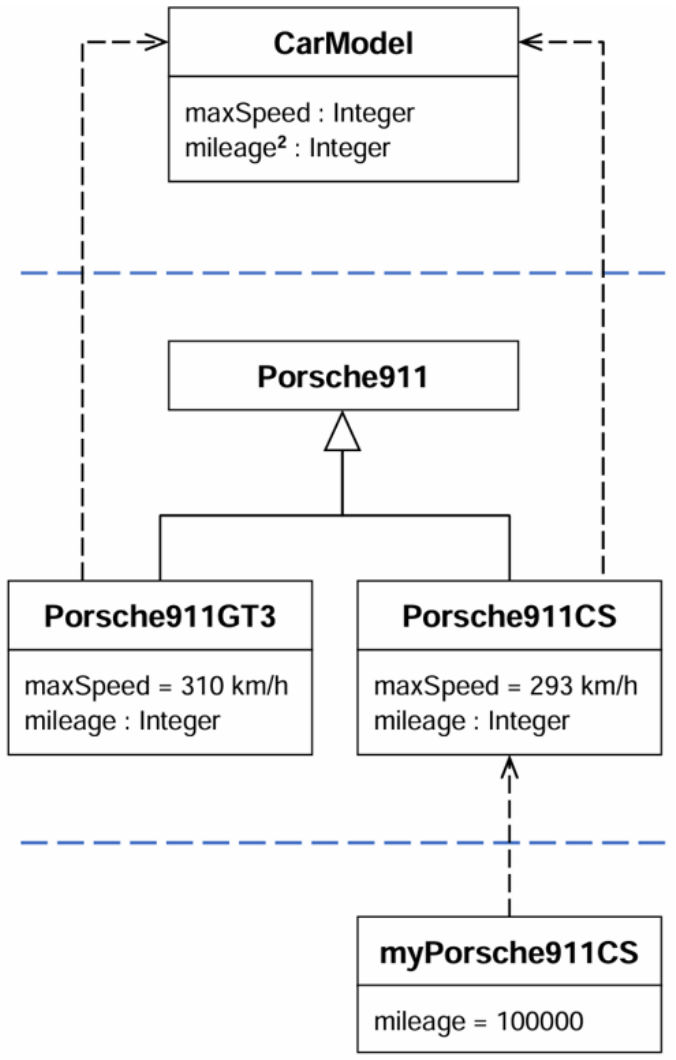
- *“Additional abstraction levels for some domain concepts cannot be introduced without requiring global model changes”*

Example

- Want to add a concept of intermediate specificity
  - **Porsche911**
- Claim is that a new classification level is needed

Response

- No new classification level is needed
  - natural relationship is specialization
  - **Car** should be **CarModel**



Actual Solution

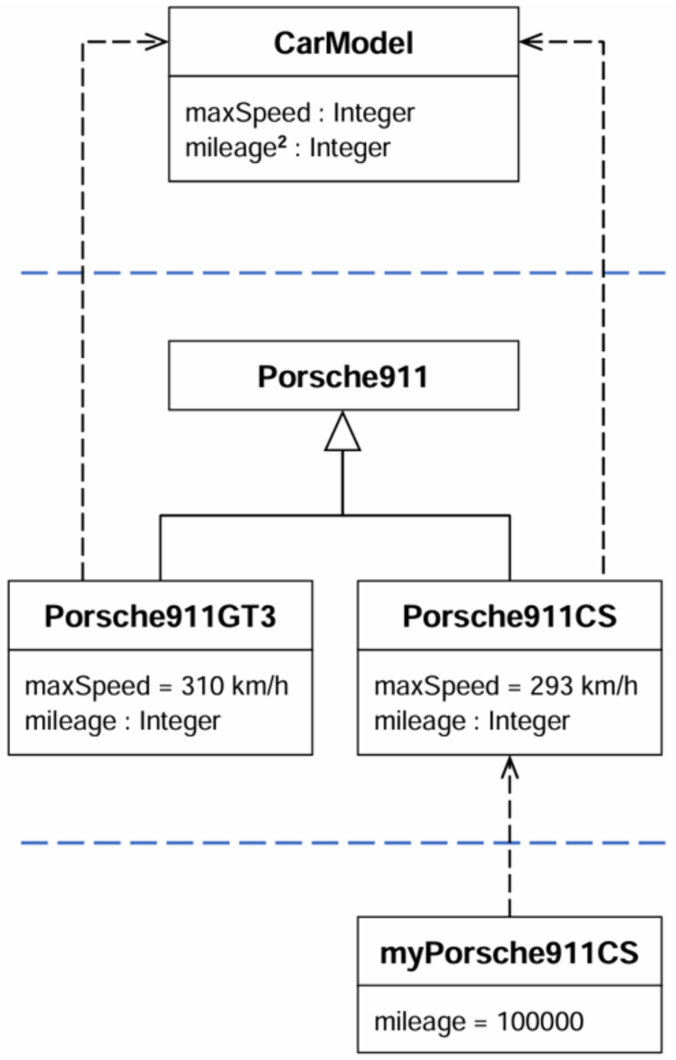
Level Instability

Criticism

- *“...there is an inherent (built-in) level coupling between adjacent levels, since the instance facet of a level class model is a partial instance of its immediate higher level...”*
- Changes to potencies of higher-level clabjects can impact clabjects several levels below
  - makes lower levels instable

Response

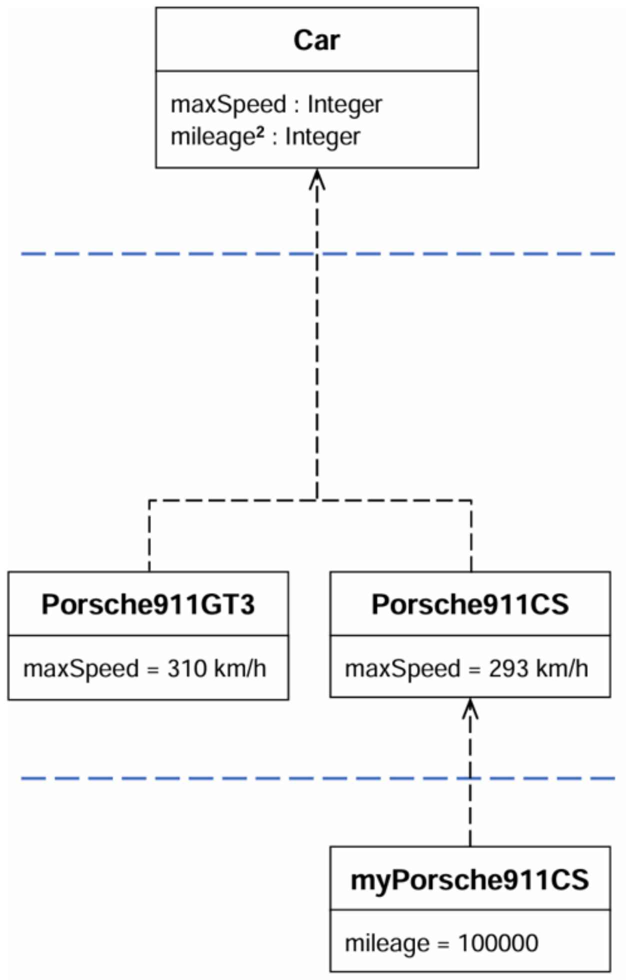
- It is true that lower levels in a deep instantiation hierarchy are highly dependent on higher level
  - but this is usually a good thing, since instances are fundamentally dependent on their types
- Can be mitigated by emendation services



Confounding MLM Relationships

Criticism

- *"...in the potency approach there is an implicit introduction of a generalization relationship ... hidden within the instance-of relationship overlain by a potency decrement..."*
  - Don't **Porsche911GT3** and **Porsche911CS** inherit **mileage** form **CarModel** ?



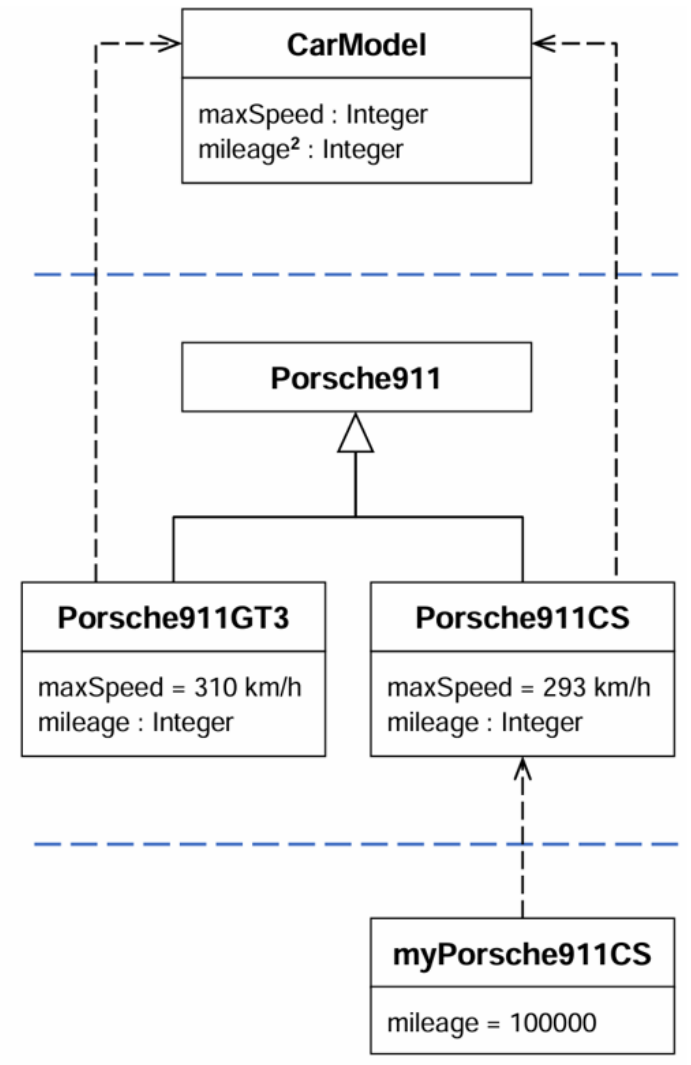
Confounding MLM Relationships

Criticism

- *"...in the potency approach there is an implicit introduction of a generalization relationship ... hidden within the instance-of relationship overlain by a potency decrement..."*
  - Don't **Porsche911GT3** and **Porsche911CS** inherit **mileage** form **CarModel** ?

Response

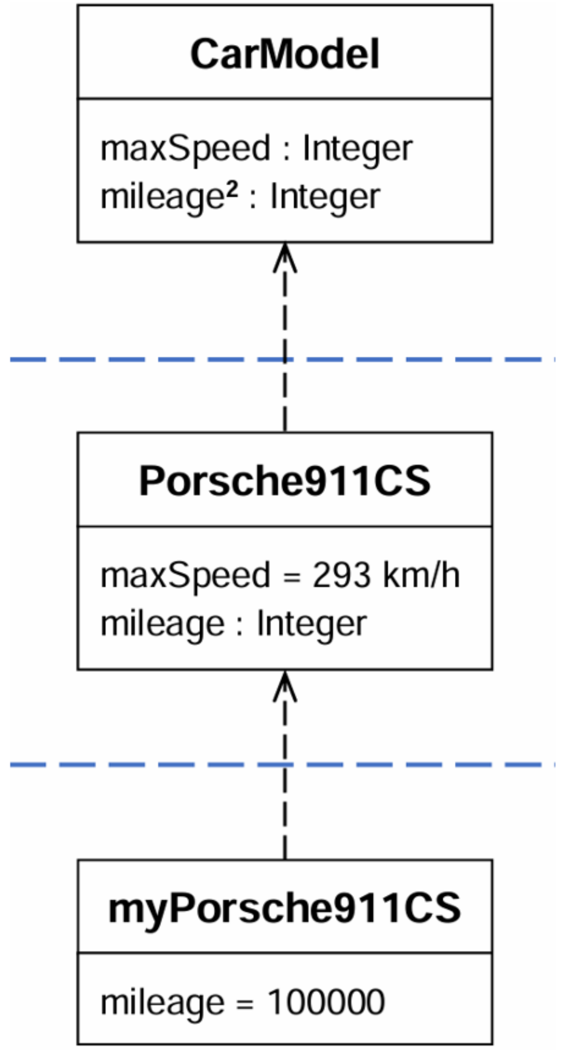
- Inheritance implies attainment of the "same" basic entity
  - features with different potencies are fundamentally different, although related, entities
- Unification is taking place, just as for clabjects
  - but the fact that a feature has a type facet doesn't mean that it has to be inherited



Confounding Concepts

Criticism

- *“...both the element and the element kind are confounded when using the potency-based approach...”*
  - Doesn’t **CarModel** represent both a supertype and metatype of **Porsche911CS** ?



Deep Instantiation

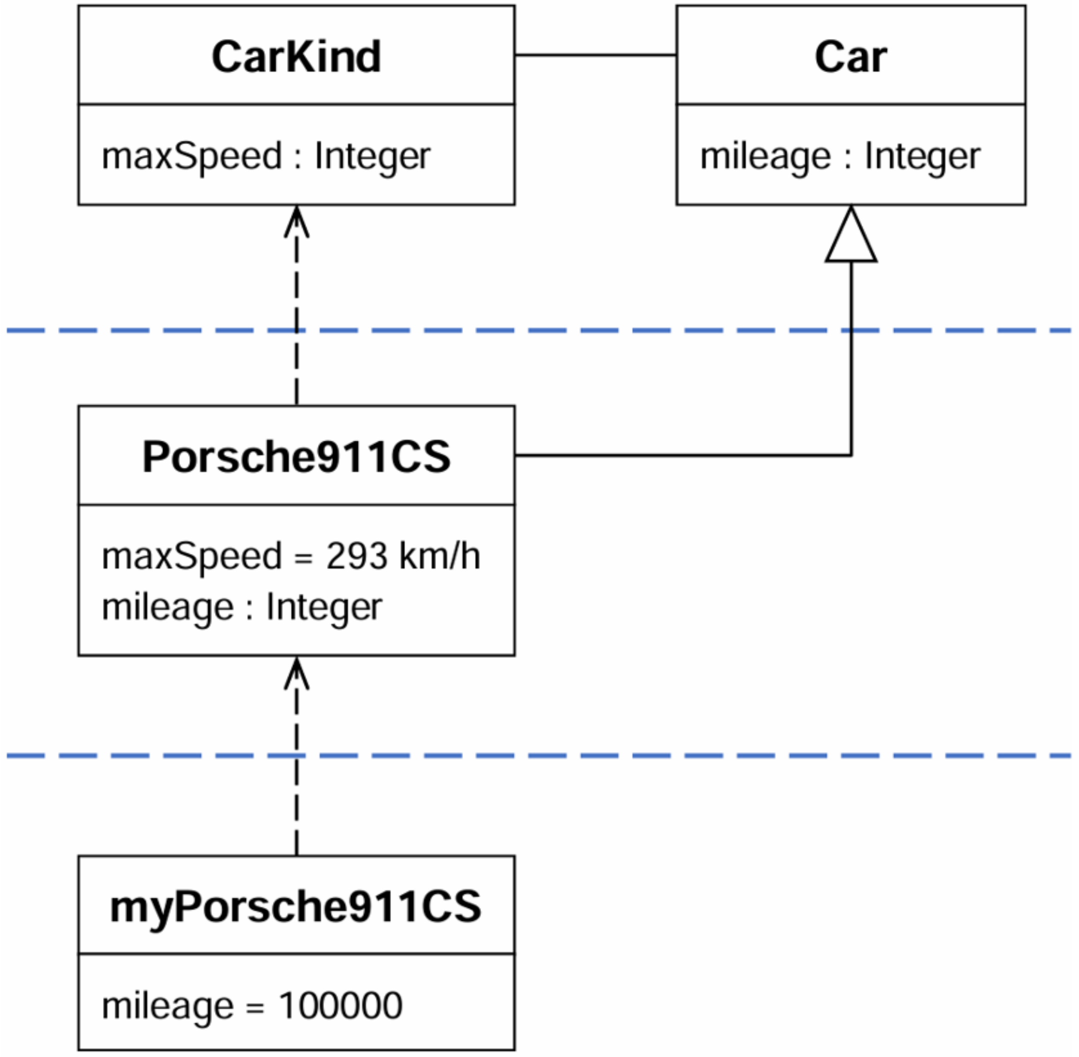
Confounding Concepts

Criticism

- *"...both the element and the element kind are confounded when using the potency-based approach..."*
  - Doesn't **CarModel** represent both a supertype and metatype of **Porsche911CS** ?

Response

- A clabject can be "forced" to have type-facet features (with potency > 0) by -
  1. its supertype
  2. a constraint
  3. its metatype (through deep instantiation)
- Just because (1) is a well-established mechanism doesn't give it conceptual superiority or exclusive correctness



Powertype Pattern

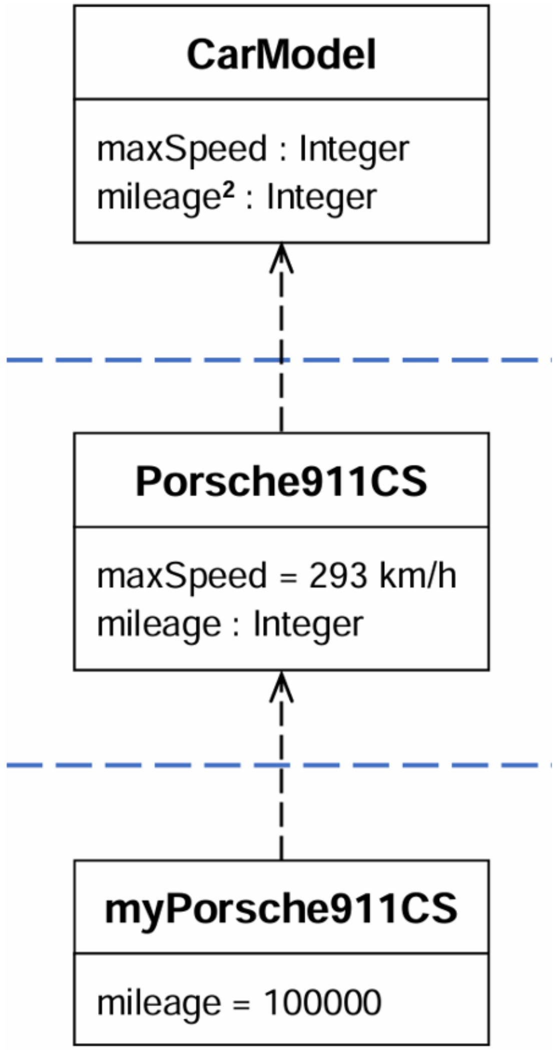
Missing Generalization

Criticism

- Deep instantiation inherently excludes generalization relationships
- which -
  - inappropriately *“...hide elements by collapsing them into a single object at the topmost level”*
  - leads to *“...conceptual mismatch with the domain... in models where such a concept is relevant.”*

Response

- Deep instantiation provides an additional way to force clabjects to have type-facet features but does not prohibit or override the other ways
  - generalisation is possible and encouraged when it best matches the domain



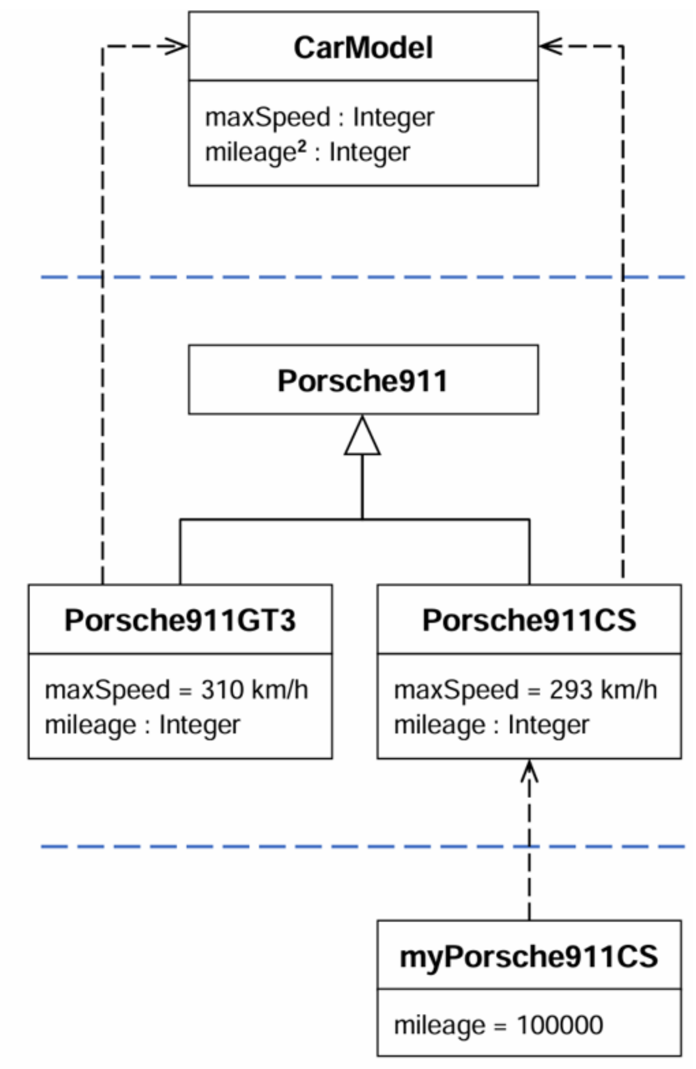
Accidental Complexity

Criticism

- *"...the emphasis on concise models may increase accidental complexity by hiding relevant domain objects as (unnamed) facets of objects at higher-levels of abstraction."*
- leads to *"construct overload"*

Response

- Deep instantiation allows generalisations (i.e., a clabject to inherit type-facet features from supertypes)
- When a domain features natural supertypes and metatypes for a concept (e.g., Breed and Dog for Collie)
  - omitting Dog may lead to an incomplete model, but not accidental complexity
  - Deep instantiation obviates many modeling concepts



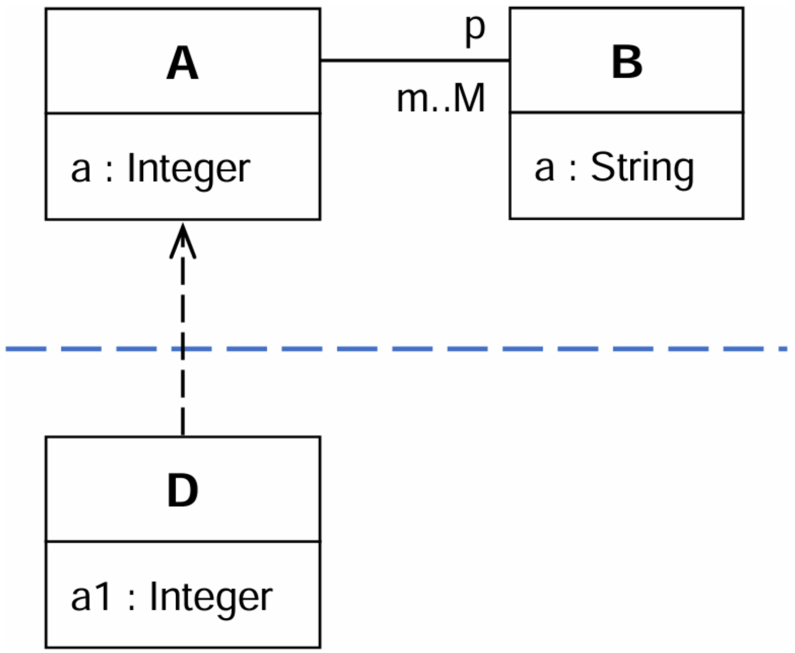
Type Safety

Criticism

- *“...the MLM community has not reached a consensus on the critical issues of clabject typing...”*
- While deep instantiation *“...mechanisms offer flexibility by allowing control of features along modeling levels, they raise issues of type computation and type safety.”*

Example

- What is the type facet of **D** ?



Response

- The type-facet of D depends on the potencies of the shown features, which make demands about -
  - feature presence (governed by classic intension satisfaction requirement)
  - feature potency (governed by potency rules)

Conclusion

- The modelling constructs offered by all programming languages embody pragmatic trade-offs
  - different trade-offs have pros and cons for different use cases, goals and domains
  - deep instantiation certainly has room for improvement

However

- The aforementioned criticisms are largely based on assumptions that do not apply to deep instantiation
- Suboptimal criticisms -
  - apply assumptions from approaches with different level-definition concepts (e.g. concretization)
  - or are based on the notion that generalisation is superior irrespective of the concepts in the domain