On submodels and submetamodels with their relation

Bernard Carré, Gilles Vanwormhoudt, Olivier Caron

Manuscript

The final publication is available at Springer
http://link.springer.com
via:
http://dx.doi.org/10.1007/s10270-016-0540-2

To cite this paper:
B. Carré, G. Vanwormhoudt, and O. Caron.
On submodels and submetamodels with their relation
A uniform formalization through inclusion properties.
Software and Systems Modeling, 17 (4) pp.1105–1137
Springer, Oct 2018.

Software and Systems Modeling manuscript No.
(will be inserted by the editor)

On submodels and submetamodels with their relation

A uniform formalization through inclusion properties

Bernard Carré[1], **Gilles Vanwormhoudt**[1,2], **Olivier Caron**[1]

[1] CRIStAL, UMR CNRS 9189
University of Lille
France - 59655 Villeneuve d'Ascq Cedex
e-mail: {Bernard.Carre, Gilles.Vanwormhoudt, Olivier.Caron}@univ-lille1.fr
[2] Institut Mines-Télécom
France - 75634 Paris Cedex 13

Received: date / Revised version: date

Contents

Abstract Model-Driven Engineering (MDE) recognized software models as first class objects with their own relationships and operations, up to constitute full structured model spaces. We focus on inclusion capacities through the concepts of submodel and submetamodel which contribute a lot to the structuring effort. Submodels and submetamodels underly many MDE practices which require their precise characterization for plain control. A typical application is model management as offered by model repositories. On the basis of results on submodel inclusion we stated in a preceding paper, we concentrate on the special form of submodels which are submetamodels and their specific role in model space structuring. Pointing out that relating submodels and submetamodels is two way, their respective inclusion hierarchies will be systematically characterized and symmetricaly compared under the logical relationships of metamodel membership and model well-formedness. As a major result, it will be shown that submodel well-formedness w.r.t submetamodels closely relates to submodel invariance (a property which guarantees transitive structure preservation) applied at both levels. The uniform formalization offers algebraic grounding to better comprehension and control of model spaces which underly MDE activities. At a much more practical level, reusable technology which takes advantage of established results will be offered.

Key words Submodel – Submetamodel – Model Space – Set-Theoretic Formalization – Model Repository

1 Introduction

Model-Driven Engineering (MDE) recognized software models as first class objects with their own relationships and operations, up to constitute full structured model spaces [30]. As always when it comes to structuring issues, inclusion capacities are central. Applied to software models, this calls for precisely studying the concept of

submodel and its properties. A major property is transitivity to offer locality and modularity qualities to model spaces and their processing, and finally efficiency.

Submodels are omnipresent in MDE practices. Composition methodologies [15, 17] make use of submodels as components in system design. View and aspect oriented modeling [5, 27, 37, 44, 45, 58] offer ways to capture multiple dimensions of systems through the isolation of submodel concerns. Template and pattern guided modeling approaches [14, 31, 32, 56] aim at defining reusable model pieces through parameterization. More generally, submodels are at the heart of model management as offered by model libraries and model repositories.

Model libraries [6, 8, 17, 44] have the ambition to capitalize "off she shelf" modeling components, possibly coming from preceding methodologies, and then their reuse as constitutive parts of projects. Model repositories [12, 20, 42, 43] have the additional tasks to support much more practical engineering activities such as incremental model construction (or editing) [2, 16], versioning [54] and collaborative design [51]. They must deal with not only full models but also with unspecified ones, that is to say models which are not necessary well formed w.r.t to their metamodel. Typical examples are models with dangling edges (see samples in Figure 4) [40, 54]. Such "degenerated" models may come from model parts due to preceding methodologies and activities, from primitive operations that they currently use such as model differencing and intersection or more simply from the need of registering any intermediate and incomplete modeling effort.

Therefore, in most situations, model spaces are made up of artifacts ranging from full entire models to unspecified pieces. This requires the homogeneous formalization of all these model forms to enable their comparison for consistent structuring of spaces and controlling related processing. For this, we have stated in [12] a formalism for their uniform definition and then their characterization through inclusion properties. As a major result, the concept of *invariant* submodel which guarantees transitive model structure preservation was isolated. Applied to model repositories, as a typical motivating application[1], this facilitates population and classification tasks and then rich content based retrieval with the organization of results through inclusion graph synthesis.

In the foregoing, model space structuring exclusively relies on the internal inclusion properties between their member models. In other words, when referring to (meta) model levels of MDE, obtained structuring qualities are only due to comparing models by themselves at their only level. They make no or little reference to the meta level otherwise than they come from a common metamodel of origin which determines the model space to be structured and it is the desired outcome. Here we concentrate on the special form of submodels which are *submetamodels* and their specific role to determine and structure model spaces. The more metamodels are determinant in MDE to define model spaces, the more submetamodels are prominent to structure them.

Basically, submetamodels do facilitate (meta) model comprehension through metamodel partitioning such as the division of the UML standard specification itself in specialized forms of diagrams [1]. In the same vein, template and pattern guided modeling methodologies applied at the meta level [14, 21] aim at contributing to this structuring effort through the possibility of abstracting recurring submetamodels. Metamodel guided view or aspect oriented modeling approaches, and more generally model projection techniques [4, 11, 47], make use of submetamodels in order to specify their concerns for circumscribing specific parts of models they are interested in.

In the field of model repositories, kept as a typical motivating situation, submetamodels are able to enrich model management operations by offering guiding facilities. First of all, as full submodels, they must be treated equally in population, classification and retrieval facilities applied at their proper level. Secondly, due to their specific role for determining and structuring model spaces, they offer capacities to organize collections of registered (sub) models and facilitate their retrieval.

From this, the problem of also considering submetamodels for model space structuring appears to be twofold:

- Following the very idea of MDE that metamodels are models, how far (sub) metamodels can be defined the same way (sub) models are, to take benefit of submodel structuring properties at their own level?
- Assuming the preceding, how far submetamodel structuring at the meta level can be related to submodel structuring at the model level and reciprocally, due to model level interdependency? This particularly refers to the question of (sub) model well-formedness w.r.t pointed (sub) metamodels.

The formalism stated in [12] offers keys to address these issues and it is the subject of this paper. In Section 2, after a reminder on the formalism and its major properties which will be exploited, the problem will be formally motivated. In Section 3 it will be shown how (sub) metamodels are defined the same way (sub) models are and how (sub) models are related to (sub) metamodels. In particular the original concept of "proper metamodel" of a model will be introduced to bridge submodel and submetamodel inclusion properties.

Then, pointing out that the problem of relating submodels and submetamodels is two-way, it will be systematically and symmetrically examined under the two following logical relationships: (sub) metamodel membership of (sub) models in Section 4 and (sub) model well-formedness w.r.t (sub) metamodels in Section 5. As a motivation, two major symmetrical results will be:

[1] `http://www.cristal.univ-lille.fr/caramel/submodel`

- Property 9 : well formed models w.r.t a metamodel M are also well formed w.r.t any enclosing metamodel whose M is an invariant submetamodel and transitively, thanks to invariance submodel transitivity as stated [12] here applied at the meta level.
- Property 15 : submodel invariance preserves well-formedness.

Consequences on model space structuring and processing effectiveness thanks to the transitivity of these properties will be shown.

After that, in Section 6, application to model repositories and reusable technology within the Eclipse modeling environment which make use of the results will be presented. Quantified time benefits on a repository of about 3000 models are shown. Finally, in Section 7, related works and MDE issues which may profit from the stated results will be examined before concluding on perspectives.

2 From submodels to submetamodels

In [12] models are defined as sets of model elements *plus* the dependency constraints that they assert over these elements, reflecting their structure. Asserted constraints are represented by a partial ordering relationship local to the model. The formalism reveals powerful enough to define models, submodels or any model fragment, being well formed or not, in order to compare them through model inclusion, as required by many MDE practices synthesized in the paper.

It was noted that handling all these modeling artifacts in an homogenous manner makes difficult their representation using graphs. This is mainly due to the problem of representing malformed models which do not respect their metamodeling constraints, such as cardinalities or containment hierarchy ones, as previously identified in [40, 54]. For example, models with "dangling edges" [51] (exemplified in Figure 4) are problematic. They may come from model differencing (an operation currently used in model versioning [2, 54], collaborative design [51] and more generally model management [43]) or more simply from any intermediate and incomplete design effort.

Thanks to the formalism, nested concepts of *closed*, *covariant* and *invariant* submodels were isolated and precisely characterized. They allow to gradually obtain structure preserving submodel transitivity. Transitivity is important because it guarantees qualities such as locality and modularity of model spaces and their processing by MDE operations which manipulate submodels. Then, following the very idea of MDE that metamodels, meta-metamodels and so on are models, concepts may apply at the meta level as stated in Section 3.1 in order to take benefit of their structuring properties. Before that, let us recall the formalism and its main properties which will be exploited.

2.1 From subsets of model elements to submodels: a reminder

Let $ModelElements$ the set of all model elements of a modeling space, a model is defined as followed:

Definition *Model (Def. 1 of [12])*
A model m is a couple $(\widetilde{m}, \sqsubseteq_m)$ such as:

- $\widetilde{m} \subseteq ModelElements$ is the set of model elements of m
- $\sqsubseteq_m$: $\widetilde{m} \times \widetilde{m}$ is a partial ordering relationship, local to $\widetilde{m}$, which translates the dependency constraints asserted by m on its model elements. The standard notation of graph of a relation, here applied to $\sqsubseteq_m$, defined as $Gr(\sqsubseteq_m) = \{(x, y) \in \widetilde{m} \times \widetilde{m} \mid x \sqsubseteq_m y\}$ may be equivalently used.

Thanks to MDE technology, a modeling space can easily be determined by its metamodel which states the set of model elements ($ModelElements$) and possible dependency constraints which can apply upon these elements.

Example
Take metamodel of Figure 1 inspired by the one used in [12][2]. This metamodel stands for models constituted of classes put in packages. Classes may have attributes and operations (possibly with parameters) and be linked through binary associations. $ModelElements$ contains all instances of the metaconstructs *Package, Class, Attribute, Operation, Parameter, Association.*
For any model $m = (\widetilde{m}, \sqsubseteq_m)$ with $\widetilde{m} \subseteq ModelElements$, possible dependency constraints are:[3]

- A class C is put in a package P: $C \sqsubseteq_m P$
- A class C owns an attribute *att* or an operation *op*: $att \sqsubseteq_m C$, $op \sqsubseteq_m C$
- An operation *op* has a parameter p: $p \sqsubseteq_m op$
- An association *asso* relates to an ending class C: $asso \sqsubseteq_m C$

Figure 2 shows a model *hhs* issued from this metamodel for modeling "home heating systems". Following the formalism, its formal modeling is (graphically schematized in Figure 3):
$hhs = (\widetilde{hhs}, \sqsubseteq_{hhs})$:

- $\widetilde{hhs}$ ={*HHS, Boiler, temperature, on, off, pipes, Radiator, position, up, down, link, aBoiler, controls, Regulator, expectedT, regulate, sensitives, Sensor, ambientT, connect, aRegulator, notify*}

[2] For sake of continuity, examples of the present paper will be directly inspired from those of [12].

[3] $\sqsubseteq_m$ being an order, it is reflective so that we systematically have: $\forall x \in \widetilde{m}, x \sqsubseteq_m x$. For sake of simplicity we will not list these reflective dependencies. In graph representation this corresponds to "Hasse Diagram" simplification which ignores them but also systematic transitive links. It will be used in figures.

– $\sqsubseteq_{hhs}$:
  – $Boiler \sqsubseteq_{hhs} HHS$
  – $Radiator \sqsubseteq_{hhs} HHS$
  – $Regulator \sqsubseteq_{hhs} HHS$
  – $Sensor \sqsubseteq_{hhs} HHS$
  – $temperature \sqsubseteq_{hhs} Boiler$
  – $on \sqsubseteq_{hhs} Boiler$
  – $off \sqsubseteq_{hhs} Boiler$
  – $pipes \sqsubseteq_{hhs} Boiler$
  – $pipes \sqsubseteq_{hhs} Radiator$
  – $position \sqsubseteq_{hhs} Radiator$
  – $up \sqsubseteq_{hhs} Radiator$
  – $down \sqsubseteq_{hhs} Radiator$
  – $aBoiler \sqsubseteq_{hhs} link$
  – $link \sqsubseteq_{hhs} Radiator$
  – $controls \sqsubseteq_{hhs} Radiator$
  – $controls \sqsubseteq_{hhs} Regulator$
  – $expectedT \sqsubseteq_{hhs} Regulator$
  – $regulate \sqsubseteq_{hhs} Regulator$
  – $sensitives \sqsubseteq_{hhs} Regulator$
  – $sensitives \sqsubseteq_{hhs} Sensor$
  – $ambientT \sqsubseteq_{hhs} Sensor$
  – $connect \sqsubseteq_{hhs} Sensor$
  – $aRegulator \sqsubseteq_{hhs} connect$
  – $notify \sqsubseteq_{hhs} Sensor$

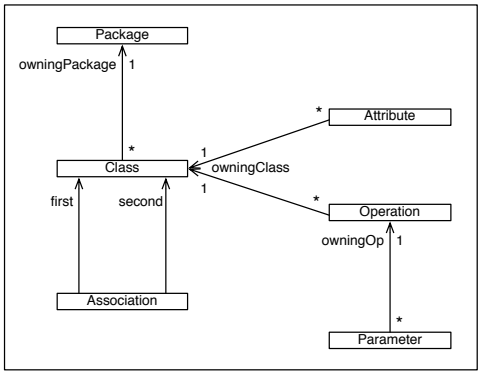

Fig. 1 Class metamodel

Then, for a model, say m, to be a submodel of another one, say m', it must at least satisfy the following inclusion conditions on its constituents:

– model elements of m are in m': $\widetilde{m} \subseteq \widetilde{m'}$
– constraints asserted by m on its model elements are also asserted by m': $\forall x, y \in \widetilde{m}, x \sqsubseteq_m y \Rightarrow x \sqsubseteq_{m'} y$, or equivalently: $Gr(\sqsubseteq_m) \subseteq Gr(\sqsubseteq_{m'})$.

This concept of submodel is transitive thanks to primary set inclusion transitivity applied to both sets of model elements and constraints. But it does not guaranty model structure preservation, indeed:

1. Elements of a model whose other ones depend due to its constraints may not be elements of the submodel.

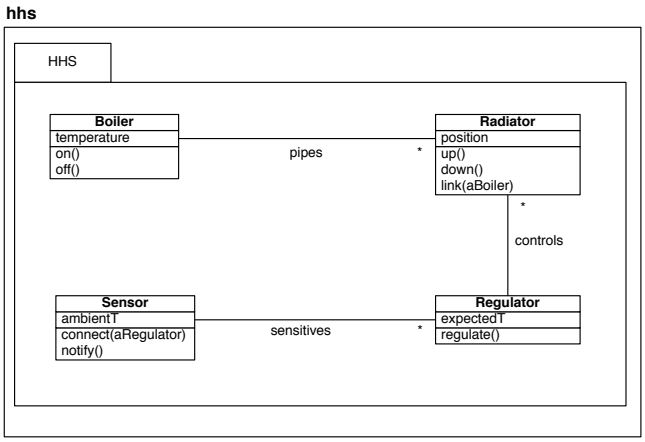

Fig. 2 Model *hhs* of home heating systems

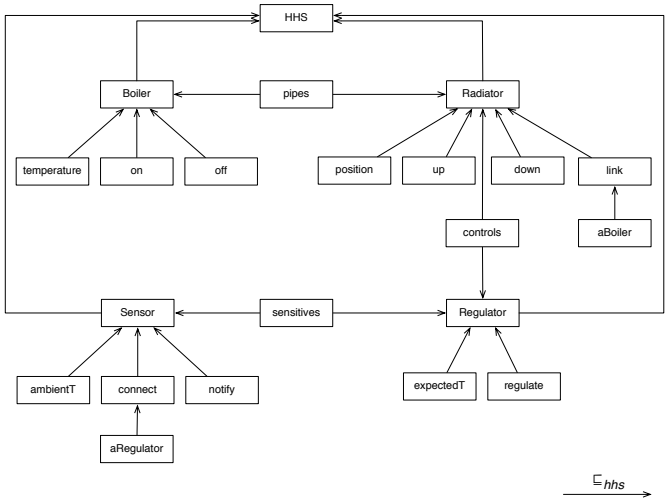

Fig. 3 Representation of *hhs* (Figure 2) in the formalism

2. Even if these elements are present in the submodel, related structuring constraints asserted by the model may be lacking in the submodel.

Examples
See Figure 4. Model $m1$ is:

– $\widetilde{m1} = \{HHS, Boiler, pipes\}$
– $Gr(\sqsubseteq_{m1}) = \{(Boiler, HHS), (pipes, Boiler)\}$.

$m1$ is a submodel of *hhs*:

– $\widetilde{m1} \subseteq \widetilde{hhs}$
– $Gr(\sqsubseteq_{m1}) \subseteq Gr(\sqsubseteq_{hhs})$

Though, $m1$ does not preserve the model structure of the surrounding model *hhs* if only because it does not contain model element *Radiator* whose its model element *pipes*, shared with this model, depends (Point 1).

Model $m2$ is:

- $\widetilde{m2} = \{HHS, Boiler, pipes, Radiator\}$
- $Gr(\sqsubseteq_{m2}) = \{(Boiler, HHS), (Radiator, HHS), (pipes, Boiler)\}$.

Here, model element *Radiator* is present. But this model does not assert the dependency constraint between *pipes* and *Radiator*, as specified by *hhs* (Point 2). No more than $m1$, included model $m2$ of *hhs* preserves the modeling structure of this surrounding model. Note that, following simple set inclusion, $m1$ is transitively a submodel of *hhs* through $m2$, though it is not structure preserving.

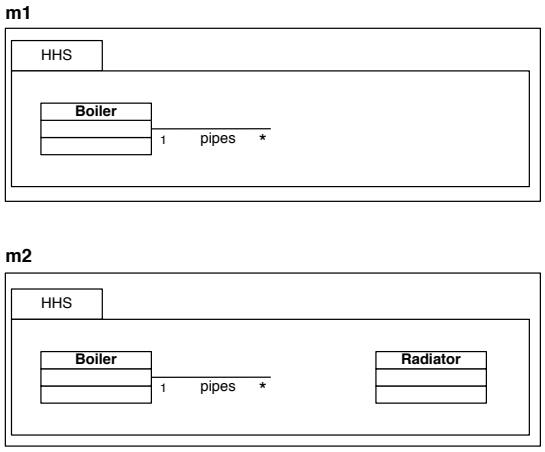

Fig. 4 Model parts of model *hhs* of Figure 2

To prevent Point 1, the concept of *closed subset of model elements of a model* was identified:

Definition *Closed subset of a model (Def. 2 of [12])*
Let $m = (\widetilde{m}, \sqsubseteq_m)$ a model, $s \subseteq ModelElements$ a set of model elements, we will say that s is *closed in* m iff:

- $s \subseteq \widetilde{m}$
- s is transitively closed under the $\sqsubseteq_m$ relationship: $closure_{\sqsubseteq_m}(s) = s$,
  where $closure_{\sqsubseteq_m}(s) = \{x \in \widetilde{m} \mid \exists y \in s, y \sqsubseteq_m^* x\}$, that is the set of model elements of m whose elements of s transitively depend, with respect to the dependency constraints asserted by m.

Application to models leads to the following definitions of *closed submodel* (only retaining their sets of model elements dimension) and *covariant submodel* (also considering the inclusion of their structuring constraints).

Definition *Closed submodel (Def. 3 of [12])*
A model $m = (\widetilde{m}, \sqsubseteq_m)$ is said to be a closed submodel of a model $m' = (\widetilde{m'}, \sqsubseteq_{m'})$ iff the set of model elements of m ($\widetilde{m}$) is closed in m'.

Definition *Covariant submodel (Def. 4 and 5 of [12])*
A model $m = (\widetilde{m}, \sqsubseteq_m)$ is said to be a covariant submodel of a model $m' = (\widetilde{m'}, \sqsubseteq_{m'})$ iff:

- $\widetilde{m}$ is closed in m'
- dependency constraints asserted by m are covariant with the m' ones, that is: $\forall x, y \in \widetilde{m}, x \sqsubseteq_m y \Rightarrow x \sqsubseteq_{m'} y$, or equivalently: $Gr(\sqsubseteq_m) \subseteq Gr(\sqsubseteq_{m'})$.

By definition, covariant submodels are closed ones. But these concepts are not transitive because *closure* is not. Though, it was shown that they constitute a step towards structure preserving submodel transitivity thanks to the property of "bound closure" (Property 1 of [12]): the closure in a surrounding model m of subsets of model elements of a closed submodel of m are bound by this submodel. Far beyond its theoretical interest, this property guaranties locality of submodel operation when only considering their sets of model elements.

Though it is not sufficient to ensure structure preserving submodel transitivity because structuring constraints asserted by models on their elements must be taken into account, as shown above. Then the identification of the *invariant submodel* concept:

Definition *Invariant submodel (Def. 6 and 7 of [12])*
A model $m = (\widetilde{m}, \sqsubseteq_m)$ is an invariant submodel of a model $m' = (\widetilde{m'}, \sqsubseteq_{m'})$ iff:

- $\widetilde{m}$ is closed in m'
- dependency constraints asserted by m are invariant with the m' ones, that is:
  $\forall x, y \in \widetilde{m}, x \sqsubseteq_m y \Leftrightarrow x \sqsubseteq_{m'} y$
  or equivalently: $Gr(\sqsubseteq_{m'})/\widetilde{m} = Gr(\sqsubseteq_m)$
  with $Gr(\sqsubseteq_{m'})/\widetilde{m} = \{(x, y) \in \widetilde{m} \times \widetilde{m} \mid x \sqsubseteq_{m'} y\}$.

Invariant submodels are covariant, so closed ones (Property 2 of [12]). Though obvious at this stage of the characterization, it is worth noting that this is required for constraint invariance to have a chance to apply (see second condition in the preceding definition). Indeed, all elements of the surrounding model m' in the definition whose elements of the candidate submodel m depend under $\sqsubseteq_{m'}$ must be present in this submodel for them to be able to participate to its proper structuring constraints. For example, model $m1$ of Figure 4 has no chance to be invariant in *hhs* because model element *Radiator*, whose association *pipes* depends under $\sqsubseteq_{hhs}$, is lacking.

But this time, as a major result, structure preserving submodel transitivity is obtained:

Property *Invariance transitivity (Prop. 3 of [12])*
Let three models $m = (\widetilde{m}, \sqsubseteq_m), m' = (\widetilde{m'}, \sqsubseteq_{m'})$ and $m'' = (\widetilde{m''}, \sqsubseteq_{m''})$, m is an invariant submodel of m' and m' is an invariant submodel of $m'' \Rightarrow m$ is an invariant submodel of m''.

Example
See Figure 5. Model $m3$ is:

- $\widetilde{m3} = \{HHS, Boiler, on, off, pipes, Radiator, link, aBoiler\}$.

– $Gr(\sqsubseteq_{m3}) = \{(Boiler, HHS), (on, Boiler), (off, Boiler), (Radiator, HHS), (link, Radiator), (aBoiler, link), (pipes, Boiler), (pipes, Radiator)$.

$m3$ is an invariant submodel of *hhs*. It is an example of a submodel which concentrates on class operations and is unaware of class attributes detailed implementation. Model $m4$ is:

– $\widetilde{m4} = \{HHS, Boiler, pipes, Radiator\}$.
– $Gr(\sqsubseteq_{m4}) = \{(Boiler, HHS), (Radiator, HHS), (pipes, Boiler), (pipes, Radiator)$.

$m4$ is an invariant submodel of $m3$, much more unaware of any class constituents. We let you verify that $m4$ is also an invariant submodel of *hhs*, firing the submodel invariance transitivity property.

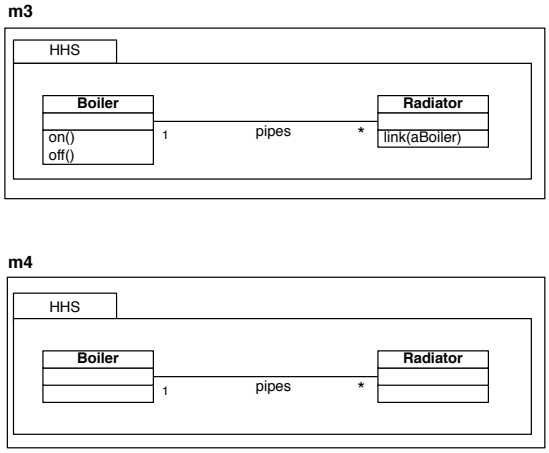

Fig. 5 Transitively invariant submodels of "home heating systems" model (Fig. 2)

2.2 From submodels to submetamodels: motivating the issue

It is essential to insist that preceding concepts of *closed, covariant* and *invariant* submodels with their properties apply to models and model fragments being well formed or not w.r.t their reference metamodel. It is the power of the formalism to embrace all these kinds of model in a uniform way to control concerned MDE practices, as motivated.

To be convinced, take major concept of invariant submodel and its transitivity property. Figure 5 showed transitively invariant submodels of the surrounding model *hhs* of Figure 2 which appear to be well formed. In comparison, reconsider submodels of Figure 4 which appear to be not. Neither they are invariant in *hhs*[4]. From this comparison, it might be intuitively thought that the invariance of submodels of Figure 5 is directly due to their well-formedness contrary to submodels of Figure 4. But it is not so. See Figure 6 which takes up not well formed model $m1$ and $m2$ of Figure 4:

– On the left, though $m1$ and $m2$ are not well formed, we let you verify that $m1$ is an invariant submodel of $m2$ which is itself invariant in new and not well formed (even more not invariant) submodel $m2'$ of *hhs* and, as expected, invariance transitivity applies: $m1$ is invariant in $m2'$.
– Right side of the figure shows the mixing of well formed and not well formed models under the submodel invariance property. Models $m0$ and $m0'$ appear to be well formed contrary to $m2'$ (as seen above). $m0$ is invariant in $m0'$ which is itself invariant in $m2'$ and indeed $m0$ is invariant in $m2'$ which is not well formed w.r.t to the originated metamodel MM of Figure 1, neither it is an invariant submodel of *hhs*.

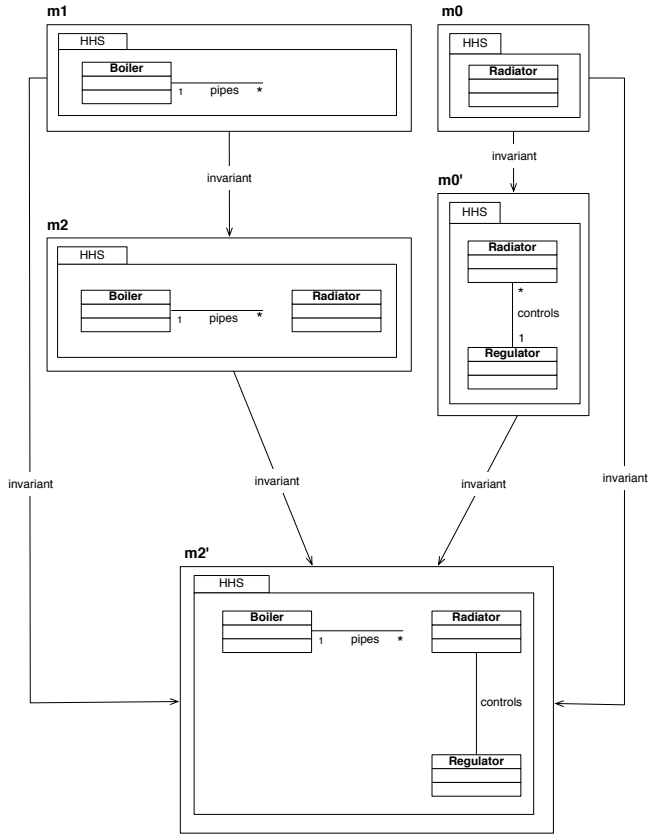

Fig. 6 Submodel invariance vs. well-formedness

As a conclusion, it is important to understand that stated concepts and properties apply to models by themselves with no or little reference to the originated metamodel other than the assumption that they belong to the modeling space that it determines. Relaxing the link to

[4] Remind that they were introduced as submodels which do not preserve structure to motivate the problem. Backwardly, due to the formalism, structure preserving submodels were formulated as invariant ones.

the meta level allows to extend the range of (sub) models under study to well formed as well as not well formed ones and state general model inclusion properties which apply to them all.

Now is the problem: what about relative position of submodels in relation to submetamodels? Thanks to the preceding formalism, this problem can be studied under the following major issues. As far as model well-formedness is questioned, this calls for the formalization of the relationship between models and their metamodel. Following the very idea of MDE that metamodels are models, it must be possible to define them the same way models were by the stated formalism. As a consequence, they may inherit stated submodel properties at their own level. So that *submetamodels* may be considered and then the additional issue of (sub) model well-formedness w.r.t submetamodels. All this will be studied in the following, leading to a uniform formalization of submodels, submetamodels and their relation through inclusion properties. To help in this task, the original concept of *proper metamodel of a model* will be isolated. It makes the bridge between submodel and submetamodel properties.

3 A uniform definition of (sub) models and (sub) metamodels with their relation

3.1 (sub) metamodels are (sub) models

The very idea of MDE is that metamodels, meta metamodels and so on are models. Following this, the preceding formalism applies to them all considering that:

1. $ModelElements$ is here the set of all model elements at any level so including model elements, metamodel elements, meta-metamodel elements and so on...
2. Each model element is linked to another one, one level above, which is the "metaconstruct" that it instantiates.

Point 2 can be specified by a function named *meta* defined as follows:

Definition 1
$meta : ModelElements \rightarrow ModelElements$
$x \mapsto X$ *such as* X *is the metaconstruct that* x *instantiates or conversely* x *is an instance of* X*.*

Function *meta* is total (no model element exists without its metaconstruct) and single valued (a model element has a single metaconstruct). Its iteration over sets of model elements allows to consider:

- $\forall s \subseteq ModelElements$, $meta(s)$ is the set of metaconstructs that elements of s instantiate
- its specific application to models: let a model $m = (\widetilde{m}, \sqsubseteq_m)$, $meta(\widetilde{m})$ is the set of metaconstructs that m instantiates.

Example
Figure 7 shows a metamodel MM for models consisting of classes (with attributes and parameterized operations) belonging to packages, mainly the way OOP does. Though simple, this metamodel will be sufficient to be used as a running example throughout the presentation of the formalism and theoretical results.

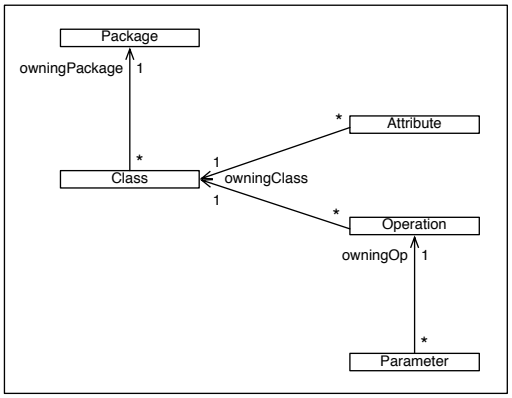

Fig. 7 Metamodel MM

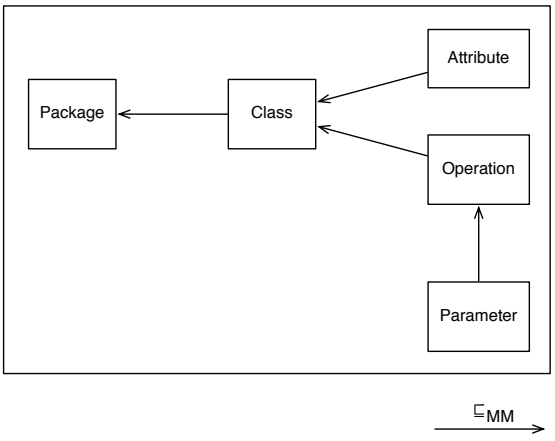

Fig. 8 MM metaconstructs and their structural dependencies

Modeling MM using the formalism is (Figure 8):

- $\widetilde{MM} =$ $\{Attribute, Parameter, Operation, Class, Package\}$
- $\sqsubseteq_{MM}$: $\widetilde{MM} \times \widetilde{MM}$
  - $Attribute \sqsubseteq_{MM} Class$
  - $Parameter \sqsubseteq_{MM} Operation$
  - $Operation \sqsubseteq_{MM} Class$
  - $Class \sqsubseteq_{MM} Package$

Figure 9 shows model samples issued from this metamodel using UML graphical convention on the left and

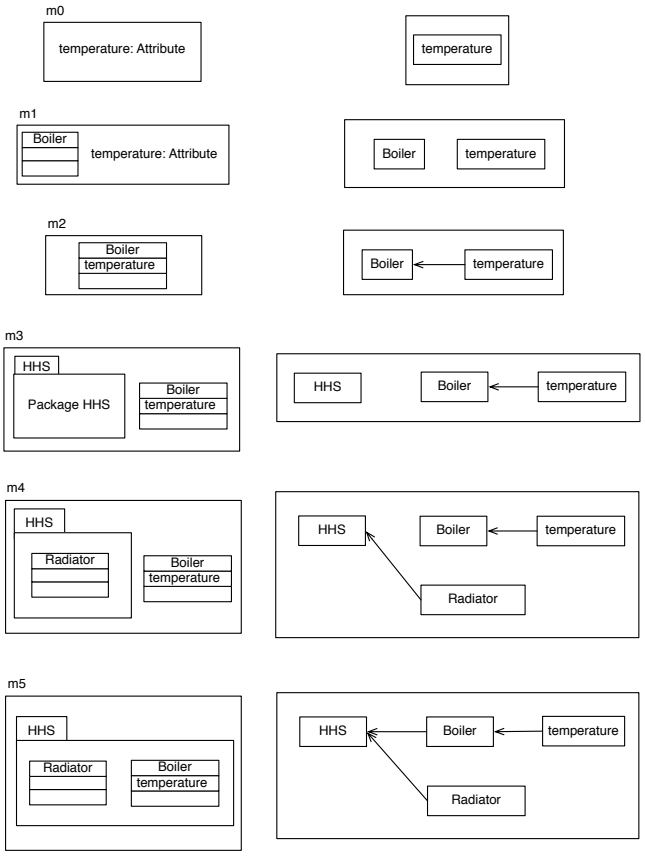

Fig. 9 Model samples

their "Hasse Graph" like representation on the right. New elements of $ModelElements$ to be considered are $temperature$, $Boiler$, $Radiator$, HHS with the following specification of function $meta$:

- $meta(temperature) = Attribute$
- $meta(Boiler) = meta(Radiator) = Class$
- $meta(HHS) = Package$.

Then details of the model samples:

- $m0$
  - $\widetilde{m0} = \{temperature\}$
  - $\sqsubseteq_{m0}$: $\emptyset$
  - $meta(\widetilde{m0}) = \{Attribute\}$
- $m1$
  - $\widetilde{m1} = \{Boiler, temperature\}$
  - $\sqsubseteq_{m1}$: $\emptyset$
  - $meta(\widetilde{m1}) = \{Class, Attribute\}$
- $m2$
  - $\widetilde{m2} = \{Boiler, temperature\}$
  - $\sqsubseteq_{m2}$
    - $temperature \sqsubseteq_{m2} Boiler$
  - $meta(\widetilde{m2}) = \{Class, Attribute\}$
- $m3$
  - $\widetilde{m3} = \{HHS, Boiler, temperature\}$
  - $\sqsubseteq_{m3}$
    - $temperature \sqsubseteq_{m3} Boiler$
  - $meta(\widetilde{m3}) = \{Package, Class, Attribute\}$
- $m4$
  - $\widetilde{m4} = \{HHS, Boiler, Radiator, temperature\}$
  - $\sqsubseteq_{m4}$
    - $temperature \sqsubseteq_{m4} Boiler$
    - $Radiator \sqsubseteq_{m4} HHS$
  - $meta(\widetilde{m4}) = \{Package, Class, Attribute\}$
- $m5$
  - $\widetilde{m5} = \{HHS, Boiler, Radiator, temperature\}$
  - $\sqsubseteq_{m5}$
    - $temperature \sqsubseteq_{m5} Boiler$
    - $Boiler \sqsubseteq_{m5} HHS$
    - $Radiator \sqsubseteq_{m5} HHS$
  - $meta(\widetilde{m5}) = \{Package, Class, Attribute\}$

As an intuitive introduction to the issues, observe that:

- w.r.t function $meta$, sets of metaconstructs of model elements of these samples are included in model elements of MM.
- dependency constraints asserted by these samples on their model elements conform to the ones asserted by metamodel MM on their metaconstructs. It will be said that these models are "member of" the metamodel (Section 4).
- only $m5$ applies all the constraints asserted by MM on its metaconstructs and, as so, appears to be well formed w.r.t this metamodel. Other model samples do not. For example $m3$ and $m4$ do not put $Class$ $Boiler$ in a $Package$ (possibly HHS) as specified by metamodel MM. This concept of (sub) model well-formedness w.r.t (sub) metamodels will be formally stated and deeply studied in Section 5.

3.2 From metaconstructs of a model to the concept of proper metamodel

The instantiation function $meta$ applied to model elements of a model, say m, allows to consider the set of metaconstructs, $meta(\widetilde{m})$, that it instantiates. On top of this, also considering dependency constraints on these metaconstructs that a model induces from its own ones leads to the concept of "proper metamodel of a model". The concept is essential because it allows sufficient representation of models at the meta level when necessary. We will see all along the paper, that it is the basis of well characterization of metamodel membership (Section 4) and model well-formedness (Section 5), due to respective inclusion (meta) model properties. After defining the concept, its application to submodels will be examined.

Proper metamodel of a model
The proper metamodel of a model m is the specific metamodel which consists of the metaconstructs of m's model elements plus the dependency constraints on these metaconstructs induced by m's ones at the meta level. More formally:

Definition 2 *The proper metamodel of a model m is the metamodel $\widehat{m} = (\widetilde{\widehat{m}}, \sqsubseteq_{\widehat{m}})$ (also noted $pmm(m)$) with :*

- $\widetilde{\widehat{m}} = meta(\widetilde{m})$, *the set of metaconstructs that m instantiates*
- $\sqsubseteq_{\widehat{m}}$: $\widetilde{\widehat{m}} \times \widetilde{\widehat{m}}$, *the dependency constraints on these metaconstructs that m specially respects:*
  $X, Y \in \widehat{m}, X \sqsubseteq_{\widehat{m}} Y$
  $\Leftrightarrow \exists\, x, y \in \widetilde{m}, meta(x) = X, meta(y) = Y, x \sqsubseteq_m y.$

Example See Figure 10 which reminds model samples of Figure 9 and shows their corresponding proper metamodel on the right:

- $\widehat{m0} = (\{Attribute\}, \emptyset)$
- $\widehat{m1} = (\{Attribute, Class\}, \emptyset)$
- $\widehat{m2} = (\{Attribute, Class\}, \{(Attribute, Class)\})$
- $\widehat{m3} = (\{Attribute, Class, Package\},$
  $\{(Attribute, Class)\})$
- $\widehat{m4} = \widehat{m5} = (\{Attribute, Class, Package\},$
  $\{(Attribute, Class), (Class, Package)\})$

Observe that models $m4$ and $m5$ have the same proper metamodel. But, contrary to $m4$, $m5$ appears to be well formed w.r.t it and the originated metamodel MM of Figure 8, as already mentioned at the end of the preceding section. This intuitively motivates the issue of well-formedness checking through the concept of proper metamodel which will be studied later (Section 5).

Proper metamodel of submodels
Now consider submodels of a model. What about the inclusion properties of their respective proper metamodels? Following property ensures that proper metamodels of submodels of a model are submetamodels of its proper one (Figure 11[5]).

Property 1
Let two models m and sm, sm is a submodel of $m \Rightarrow$ the proper metamodel of sm ($\widehat{sm}$) is a submodel of the proper metamodel of m ($\widehat{m}$).

Proof
$\widehat{sm}$ is a submodel of $\widehat{m}$ iff:

1. $\widetilde{\widehat{sm}} \subseteq \widetilde{\widehat{m}}$
2. $Gr(\sqsubseteq_{\widehat{sm}}) \subseteq Gr(\sqsubseteq_{\widehat{m}})$

[5] As far as possible, a figure will be associated to each property in order to pictorially schematize its content. Bold double arrows (which evoke implication) will be used to represent deduced relationships.

Proof of 1
$\widetilde{sm} \subseteq \widetilde{m} \Rightarrow meta(\widetilde{sm}) \subseteq meta(\widetilde{m})$ with $meta(\widetilde{sm}) = \widetilde{\widehat{sm}}$ and $meta(\widetilde{m}) = \widetilde{\widehat{m}}$.

□

Proof of 2
That is: $\forall X, Y \in \widetilde{\widehat{sm}},\ X \sqsubseteq_{\widehat{sm}} Y \Rightarrow X \sqsubseteq_{\widehat{m}} Y$. Indeed:

- $X, Y \in \widetilde{\widehat{sm}} = meta(\widetilde{sm}), X \sqsubseteq_{\widehat{sm}} Y \Rightarrow$
  $\exists\, x, y \in \widetilde{sm}, X = meta(x), Y = meta(y), x \sqsubseteq_{sm} y$
  by definition of $\widehat{sm}$ being the proper metamodel of sm
- $x, y \in \widetilde{sm} \subseteq \widetilde{m}$ and $Gr(\sqsubseteq_{sm}) \subseteq Gr(\sqsubseteq_m)$ being given (sm is a submodel of m) : $x \sqsubseteq_{sm} y \Rightarrow x \sqsubseteq_m y$
- so that $x, y \in \widetilde{m}$ and $x \sqsubseteq_m y \Rightarrow meta(x) = X \sqsubseteq_{\widehat{m}} meta(y) = Y$ by definition of $\widehat{m}$ being the proper metamodel of m.

□

We let you verify the property on submodel samples of Figure 10. Of course this inclusion property applies to closed, covariant and invariant submodels. Though, it is important to observe that no more than simple inclusion of proper metamodels of submodels (as established by preceding Property 1) can be guaranteed whatever their possible richer inclusion qualities. Remind that these qualities are nested ones and depend at least on the submodel, say sm, to be closed in the model, say m. But we do not have: sm closed in $m \Rightarrow \widehat{sm}$ closed in $\widehat{m}$. See Figure 10, a counterexample is $m2$ compared to $m4$ ($Package$ is lacking):

- $m2$ is closed (more, it is invariant) in $m4$, indeed:
  - $\widetilde{m2} = \{temperature, Boiler\}$
  - $Gr(\sqsubseteq_{m4}) = \{(temperature, Boiler),$
    $(Radiator, HHS)\}$
  - $closure_{\sqsubseteq_{m4}}(\widetilde{m2}) = \{temperature, Boiler\} = \widetilde{m2}$
- but $\widehat{m2}$ is not closed in $\widehat{m4}$:
  - $\widehat{m2} = meta(\widetilde{m2}) = \{Attribute, Class\}$
  - $Gr(\sqsubseteq_{\widehat{m4}}) = \{(Attribute, Class), (Class, Package)\}$
  - $closure_{\sqsubseteq_{\widehat{m4}}}(\widetilde{\widehat{m2}}) = \{Attribute, Class, Package\}$
    $\neq \widetilde{\widehat{m2}}$.

Indeed, there is no reason that particular submodels are closed in a model at the model level to conclude that their respective proper metamodels are at the meta level in the general case. As explained above, this applies a fortiori to covariant and invariant submodels which are basically closed ones. Take preceding examples:

- $\widehat{m0}$ to $\widehat{m2}$ are not closed in originated metamodel MM ($Package$ is lacking in all of three, $Class$ is also lacking in $\widehat{m0}$)
- $\widehat{m3}$ is closed but only covariant in MM ($Class$ is not depending on $Package$).
- $\widehat{m4} = \widehat{m5}$ are invariant in MM.

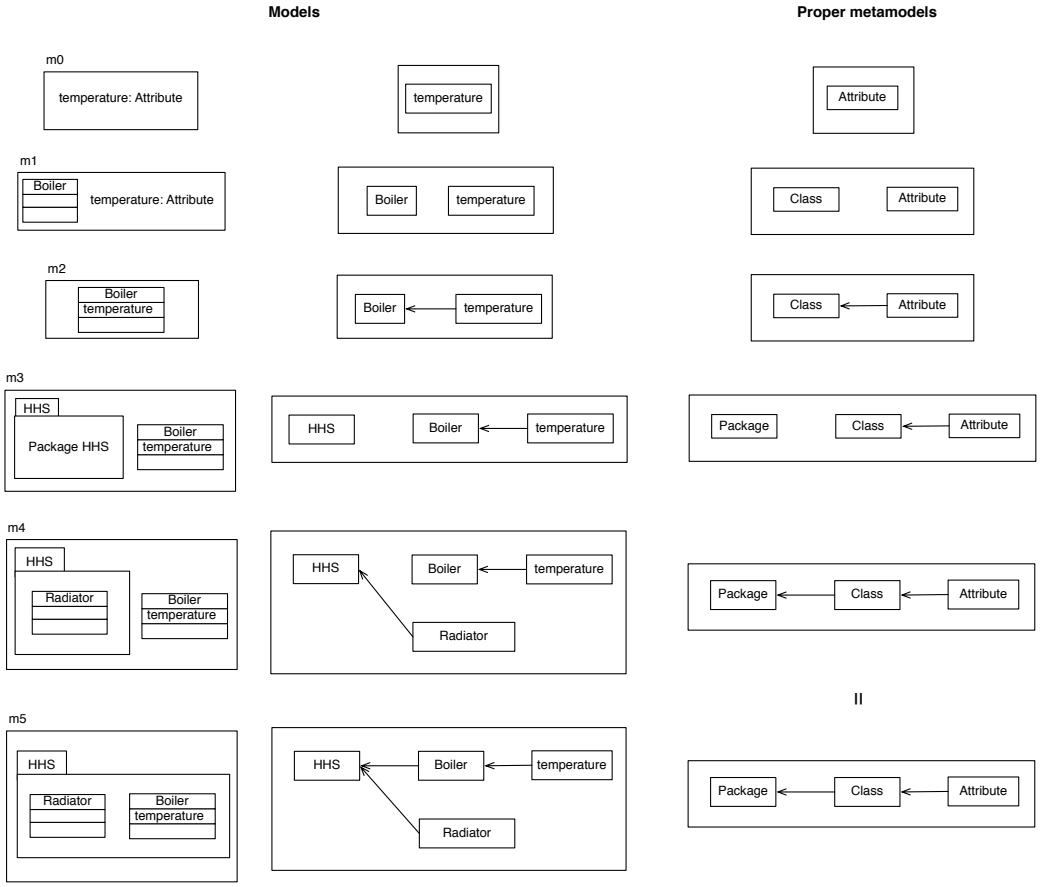

Fig. 10 Proper metamodels

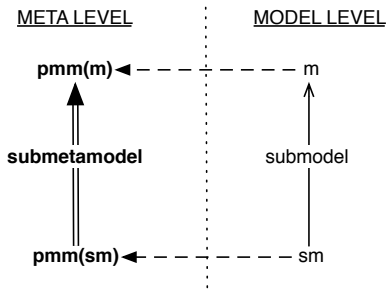

Fig. 11 Inclusion of proper metamodel of submodels (Property 1).

4 Metamodel membership

Comparing submodels to submetamodels leads to the following question: how far (sub) models can be considered to conform to (sub) metamodels? Following the definition that a (meta) model is the association of a set of model elements plus the structuring constraints that it asserts over them, the question is twofold:

– how far the metaconstructs instantiated by a (sub) model compare to (sub) metamodel ones?

– how far the structuring constraints asserted by a (sub) model on its model elements respect (sub) metamodel corresponding ones?

The concept of "metamodel membership" defined below allows to identify (sub) models which *necessarily* conform to (sub) metamodels. They have all the model elements and assert all the constraints over these elements. Though, as a preamble of Section 5, we will see that, as far as (sub) model well-formedness conformance to (sub) metamodels is also searched, "metamodel membership" is necessary but not sufficient.

Before that, let us concentrate on this primary concept of "metamodel membership" and state its own useful properties relative to positioning qualities of (sub) models in comparison to (sub) metamodels. After defining the concept, its formalization using the core concept of "proper metamodel" will be established (Section 4.1). Then its application to submetamodels (Section 4.2) and submodels (Section 4.3) will be examined.

4.1 Definition

Definition 3 *Let two models M and m, we will say that "m is a member model of M" or that "M is a metamodel of m" iff*

- $meta(\widetilde{m}) \subseteq \widetilde{M}$
- $\forall x, y \in \widetilde{m},\ x \sqsubseteq_m y \Rightarrow meta(x) \sqsubseteq_M meta(y)$.

First item means that all model elements of m are instances of the metaconstructs of metamodel M (which could have more). Second item means that constraints asserted by m on its model elements necessarily conform to metamodeling constraints asserted by M on its metaconstructs. In other words, m does not "invent" constraints on its model elements other than those asserted by M[6].

Example
Take model samples of Fig. 9 and the metamodel MM of Fig. 8. All model samples are member of MM but with the following specificities:

- all their model elements are instances of metaconstructs of MM, but not all the metaconstructs of MM are necessarily instantiated:
  - none of the models instantiates *Operation*
  - $m0$ has no $Class$, $m0$, $m1$ and $m2$ have no $Package$.
- all their asserted constraints conform to metamodeling ones of MM on instantiated metaconstructs. But they do not necessarily respect all of them:
  - either because they have no concerned model element, for example:
    - constraints on *Operation* are irrelevant in all samples
    - constraints on $Class$ are irrelevant in $m0$
  - or because instances of metaconstructs are there but do not respect the corresponding constraints. For example in $m3$ and $m4$ *Class Boiler* is not put in a package, possibly HHS.

Last item is showing that a model may be a member of a metamodel without satisfying all of its metamodeling constraints, following the second condition of the definition. It will be the difference between metamodel membership and metamodel well-formedness as studied later (Section 5). Take model $m5$ as an intuitive example of the issue. It has the same model elements as $m4$ but, contrary to this model, it does satisfy all the modeling constraints imposed by metamodel MM and, as so, appears to be well formed w.r.t this metamodel.

[6] More theoretically, in that case function *meta* must be *monotonic* when applied to model elements of m with the structure that this model asserts. A monotonic function is a special case of the more general concept of *(homo)morphism* of set theory when used in the field of order (and also graph) theory. In the general case (set theory) morphisms are structure preserving functions between structured sets. Within the field of order theory, structuring is ordering. It is the case here, comparing structured sets $(\widetilde{m}, \sqsubseteq_m)$ and $(\widetilde{M}, \sqsubseteq_M)$ under the *meta* function where $\sqsubseteq_m$ and $\sqsubseteq_M$ are ordering relationships.

Model membership and the concept of proper metamodel

By construction, a model, say m, is member of its proper metamodel. Simply compare Definition 2 and Definition 3 applied to proper metamodel $\widehat{m}$ of m :

- $meta(\widetilde{m}) = \widetilde{\widehat{m}}$ (Definition 2) $\Rightarrow meta(\widetilde{m}) \subseteq \widetilde{\widehat{m}}$ (Definition 3)
- second condition of Definition 2 ($\forall x, y \in \widetilde{m},\ x \sqsubseteq_m y \Leftrightarrow meta(x) \sqsubseteq_{\widehat{m}} meta(y)$) implies second condition of Definition 3 applied to metamodel $\widehat{m}$.

Now consider any metamodel, typically an overall reference metamodel of a MDE project, and the question of model membership in this metamodel. An essential characteristic of the concept of proper metamodel is that it makes equivalent metamodel membership of models, across the metamodeling stack, to metamodel inclusion of their proper metamodel within the only meta level. It is established by following Property 2, schematized by Figure 12.

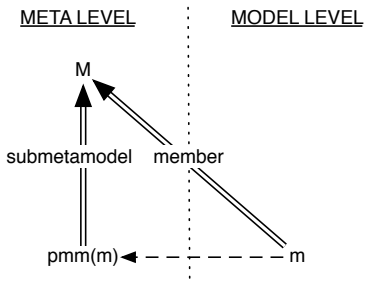

Fig. 12 Metamodel membership and proper metamodel inclusion equivalency.

Property 2
Let two models m and M, m is member of $M \Leftrightarrow$ the proper metamodel of m ($\widehat{m}$) is a submodel of M.

Proof of the necessary side:
m is a member model of $M \Rightarrow \widehat{m}$ is a submodel of M.

$\widehat{m}$ is a submodel of M iff:

1. $\widetilde{\widehat{m}} \subseteq \widetilde{M}$, indeed:
   - $\widetilde{\widehat{m}} = meta(\widetilde{m})$ by definition of proper metamodel $\widehat{m}$ of m
   - with $meta(\widetilde{m}) \subseteq \widetilde{M}$, m being a member of M (Definition 3).
2. $\forall X, Y \in \widetilde{\widehat{m}},\ X \sqsubseteq_{\widehat{m}} Y \Rightarrow X \sqsubseteq_M Y$, indeed:

- $X, Y \in \widetilde{\widehat{m}},\ X \sqsubseteq_{\widehat{m}} Y \Rightarrow$
  $\exists\, x, y \in \widetilde{m}, meta(x) = X, meta(y) = Y, x \sqsubseteq_m y$,
  by definition of proper metamodel $\widehat{m}$ of m
- but m is member of M by hypothesis so that:
  - $X = meta(x), Y = meta(y) \in \widetilde{\widehat{m}} = meta(\widetilde{m}) \subseteq \widetilde{M} \Rightarrow X, Y \in \widetilde{M}$
  - with $x \sqsubseteq_m y \Rightarrow meta(x) = X \sqsubseteq_M meta(y) = Y$.

□

Proof of the sufficient side:
$\widehat{m}$ is a submodel of $M \Rightarrow m$ is a member model of M.

m is member of M iff:

1. $meta(\widetilde{m}) \subseteq \widetilde{M}$, indeed:
   - $\widetilde{\widehat{m}} \subseteq \widetilde{M}$, $\widehat{m}$ being a submodel of M
   - with $\widetilde{\widehat{m}} = meta(\widetilde{m})$ by definition of proper metamodel $\widehat{m}$ of m.
2. $\forall x, y \in \widetilde{m},\ x \sqsubseteq_m y \Rightarrow meta(x) \sqsubseteq_M meta(y)$, indeed:
   - $x, y \in \widetilde{m},\ x \sqsubseteq_m y \Rightarrow meta(x), meta(y) \in \widetilde{\widehat{m}}$, $meta(x) \sqsubseteq_{\widehat{m}} meta(y)$ by definition of $\widehat{m}$ being the proper metamodel of m
   - but $\widehat{m}$ is a submodel of M by hypothesis so that:
     - $meta(x), meta(y) \in \widetilde{\widehat{m}} \subseteq \widetilde{M} \Rightarrow$
       $meta(x), meta(y) \in \widetilde{M}$
     - with $meta(x) \sqsubseteq_{\widehat{m}} meta(y) \Rightarrow meta(x) \sqsubseteq_M meta(y)$.

□

4.2 Submetamodel membership

Now consider submetamodels, say SM, of a metamodel M whose a model, say m, is member. Following Property 3 basically states that m is also member of the overall metamodel M (Figure 13).

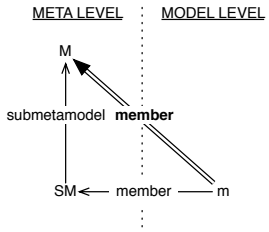

Fig. 13 Model membership of submetamodels (Property 3)

Property 3
Let three models M, SM and m, m is member of SM and SM is a submodel of $M \Rightarrow m$ is member of M.

Proof
Trivial:

- m is member of $SM \Leftrightarrow \widehat{m}$ is a submodel of SM (Property 2)
- SM being given as a submetamodel of M, $\widehat{m}$ is itself a submetamodel of M
- and then: $\widehat{m}$ is a submetamodel of $M \Rightarrow m$ is member of M (Property 2).

□

Though obvious, this property leads to important consequences when one considers the *poset*[7] of submetamodels of M whose model m is member, partially ordered through submodel inclusion. By definition this poset has a greatest element which is metamodel M itself. But, more interestingly, it also gets a smallest element which is precisely the proper metamodel of m. This is established by following Property 4 and Figure 14 schematizes the resulting membership hierarchy of m.

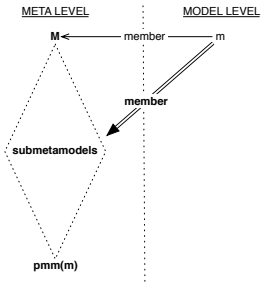

Fig. 14 Membership hierarchy.

Property 4
Let a metamodel M and a model m, m is member of $M \Rightarrow$ the proper metamodel of m ($\widehat{m}$) is the "smallest" (i.e. unique minimal) submetamodel of M whose m is member.

Proof
Suppose there exists a submetamodel of M, say SM ($\widetilde{SM} \subseteq \widetilde{M}, Gr(\sqsubseteq_{SM}) \subseteq Gr(\sqsubseteq_M)$), which is "smaller" than submetamodel $\widehat{m}$ of M but m is member of SM. SM may be smaller than $\widehat{m}$ either because :

1. $\widetilde{SM} \subsetneq \widetilde{\widehat{m}}$
   but, by hypothesis, m is member of SM so that (Definition 3) $meta(\widetilde{m}) \subseteq \widetilde{SM}$, with $meta(\widetilde{m}) = \widetilde{\widehat{m}}$ by definition of $\widehat{m} = pmm(m)$. Then the contradiction: $\widetilde{SM} \subsetneq \widetilde{\widehat{m}} = meta(\widetilde{m}) \subseteq \widetilde{SM}$.
2. $\widetilde{SM} \subseteq \widetilde{\widehat{m}}$ but $Gr(\sqsubseteq_{SM}) \subsetneq Gr(\sqsubseteq_{\widehat{m}})$
   Suppose $\exists\, X, Y \in \widetilde{SM},\ X \sqsubseteq_{\widehat{m}} Y$ but not $X \sqsubseteq_{SM} Y$

[7] A *poset* is a set of things equipped with a partial order, here (meta) models arranged by submodel inclusion partial ordering. Depending on the partial order properties, a *poset* may have bound elements : "greatest" one(s) and/or "smallest" one(s).

– $X, Y \in \widetilde{SM} \subseteq \widetilde{\widehat{m}}$ and $X \sqsubseteq_{\widehat{m}} Y \Rightarrow \exists x, y \in \widetilde{m} \mid meta(x) = X, meta(y) = Y, x \sqsubseteq_m y$ by definition of $\widehat{m} = pmm(m)$
– by hypothesis, m being member of SM: $x \sqsubseteq_m y \Rightarrow meta(x) \sqsubseteq_{SM} meta(y)$
– but $meta(x) = X$ and $meta(y) = Y$, so the contradiction.

□

An intuitive explanation of the phenomenon is that, when considering an overall metamodel and member models of this metamodel, proper metamodels of these models have the characteristic that they are circumscribing ("simplifying") the metamodel to constitutive submetamodels which directly concern them, while respecting (sub) metamodel membership.

4.3 Submetamodel membership of submodels

Complementary to the preceding section which dealt with submetamodel membership of models, examine now the question of submetamodel membership of submodels. Conjunction of Property 1 (on the inclusion of proper metamodels of submodels) with Property 2 (on the equivalency between metamodel membership of a model and metamodel inclusion of its proper metamodel) automatically gives that when a model is member of a metamodel, so are all of its submodels. It is stated by following Property 5, see Figure 15.

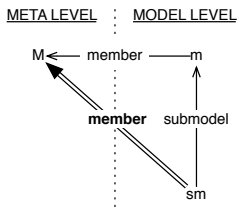

Fig. 15 Submodel membership (Property 5)

Property 5
Let three models m, sm and M. Model m is member of M and sm is a submodel of $m \Rightarrow$ sm is member of M.

Proof
Trivial:

– m is member of $M \Leftrightarrow \widehat{m}$ is a submetamodel of M (Property 2)
– sm is a submodel of $m \Rightarrow \widehat{sm}$ is a submetamodel of $\widehat{m}$ (Property 1) so that $\widehat{sm}$ is a submodel of M
– and then $\widehat{sm}$ is a submetamodel of $M \Leftrightarrow sm$ is member of M (Property 2, once more).

□

In line with Property 4 which leads to the structure of the membership hierarchy of a model (shown in Figure 14) application to submodels, say sm, of a model, say m, member of a metamodel, say M, leads to the following observations, synthesized in Figure 16.

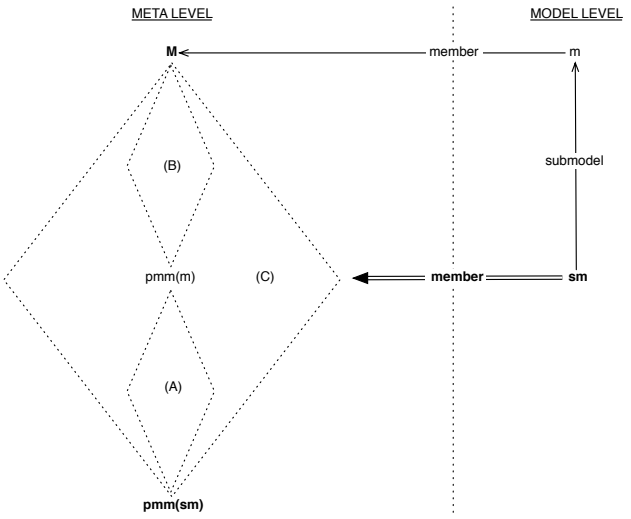

Fig. 16 Submodel membership hierarchy.

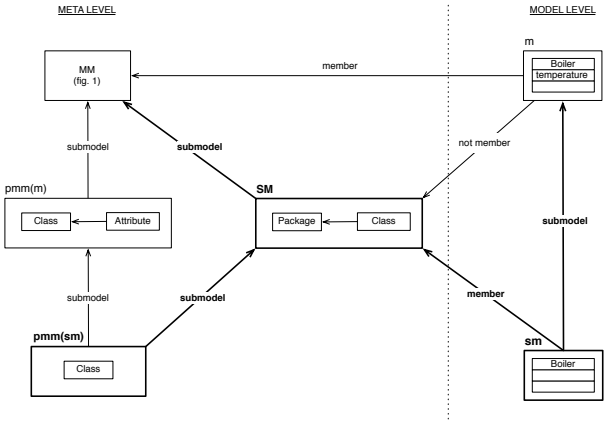

Fig. 17 sm member of SM, outside the sphere of $pmm(m)$.

Being a submodel of m which is by construction member of its proper metamodel $pmm(m)$ (Section 4.1), sm is member of this metamodel (Property 5). As so, it is also member of any submetamodel of $pmm(m)$ which enclose its own proper metamodel ($pmm(sm)$) thanks to Properties 3 and 4, and $pmm(sm)$ is the smallest one. Schematically it is the application of Figure 14 to model sm as a member of metamodel $pmm(m)$ (zone A in Figure 16).

Being member of $pmm(m)$, sm is transitively member of all submetamodels of M which are in the mem-

bership hierarchy of m thanks to Property 3 (zone B in the figure). Indeed sm being member of $pmm(sm)$ (by construction) which is a submetamodel of $pmm(m)$ (Property 1), sm is member of $pmm(m)$ thanks to Property 3. Transitively, $pmm(sm)$ is a submetamodel of all super-metamodels of $pmm(m)$, so that same Property 3 applies to them all, and particularly to those which are in the membership hierarchy of m. Note that metamodels of zone A are submetamodels of $pmm(m)$ and then of all metamodels of zone B.

Finally, as a submodel of m which is member of M, sm is member of M (Property 5) and of all submetamodels of M which enclose its proper metamodel $pmm(sm)$ thanks to Properties 3 and 4 (Figure 14 applied to sm once more). Note that this includes preceding ones (zones A and B) but not exclusively. Indeed, there may exist submetamodels of M whose sm is a member which are not in these zones (zone C). An example is given in Figure 17, in reference with overall metamodel MM of Figure 8. Model m is member of MM, sm is a submodel of m and take submetamodel SM of MM. sm is member of SM (SM encloses $pmm(sm)$) which is neither a submetamodel of $pmm(m)$ nor it encloses it (m is not a member of SM). As an intuitive conclusion, smaller is the model, bigger is its membership hierarchy.

5 Model well-formedness w.r.t a metamodel

Preceding section allowed to consider member (sub) models of (sub) metamodels, being well formed or not. It is worth noting that when comparing (sub) models by themselves "metamodeling constraints are only necessary ones", as pointed out in [12] (p. 874), and preceding statements on "metamodel membership" (Section 4) were sufficient for that.

But when also questioning well-formedness conformance of (sub) models w.r.t (sub) metamodels, metamodeling constraints must also be sufficiently verified. After identifying and defining the concept of model well-formedness w.r.t a metamodel (Section 5.1), its relation to submetamodels and the core concept of proper metamodel will be stated in Section 5.2. Then questions relative to (sub) metamodel well-formedness of (sub) models will be examined in Sections 5.3 and 5.4.

5.1 Definition

For a model $m = (\widetilde{m}, \sqsubseteq_m)$ member of a metamodel $M = (\widetilde{M}, \sqsubseteq_M)$ to be well formed w.r.t to it, it must:

1. contain all the necessary model elements imposed by the constraints of M
2. assert all the constraints imposed by M on its model elements.

About Condition 1

It means that if a model element of m instantiates a metaconstruct of M, say X, all the metaconstructs whose X transitively depends under $\sqsubseteq_M$ must be instantiated in m. In other words the set of metaconstructs instantiated by $\widetilde{m}$, that is $meta(\widetilde{m})$, must be closed in M: $closure_{\sqsubseteq_M}(meta(\widetilde{m})) = meta(\widetilde{m})$. This condition is necessary to give a member model of a metamodel the chance to instantiate the metamodeling constraints imposed by this metamodel (Condition 2).

Example
See model samples of Fig. 9 which are member models of metamodel MM of Fig. 8:

- $m0$ does not satisfy the condition, indeed:
  $\widetilde{m0} = \{temperature\}, meta(\widetilde{m0}) = \{Attribute\}$
  but $closure_{\sqsubseteq_{MM}}(meta(\widetilde{m0}))$
  $= \{Attribute, Class, Package\}$
  $\neq meta(\widetilde{m0})$
- idem for $m1$ and $m2$ which have the same set of model elements:
  $meta(\widetilde{m1}) = meta(\widetilde{m2}) = \{Attribute, Class\}$
  but $closure_{\sqsubseteq_{MM}}(\{Attribute, Class\})$
  $= \{Attribute, Class, Package\} \neq \{Attribute, Class\}$
- $m3$, $m4$ and $m5$, which have the same image under $meta$, do:
  - $\widetilde{m3} = \{temperature, Boiler, HHS\}$,
    $meta(\widetilde{m3}) = \{Attribute, Class, Package\}$
  - $\widetilde{m4} = \widetilde{m5}$
    $= \{temperature, Boiler, Radiator, HHS\}$,
    $meta(\widetilde{m4}) = meta(\widetilde{m5})$
    $= \{Attribute, Class, Package\}$
  - $closure_{\sqsubseteq_{MM}}(\{Attribute, Class, Package\})$
    $= \{Attribute, Class, Package\}$.

One can see that, despite the fact that these three latter models satisfy the condition, only $m5$ appears to be well formed (in $m3$ and $m4$, $Boiler$ is not put in a package). This will be checked by Condition 2.

About Condition 2

It means that for any model element of m, say x, if M imposes that its metaconstruct $X = meta(x)$ depends on another one, say Y, m must instantiate this constraint. That is to say x must depend on another model element of m, say y, instance of Y. At this point, it is important to see that Y must necessarily be in $meta(\widetilde{m})$. It is ensured by Condition 1, as explained above in its rationale. Condition 2 only adds the requirement that m must have a model element instance of Y (such as y) which instantiates the constraint.

Example
Consider model samples of Figure 9 which satisfy Condition 1, that is to say $m3$, $m4$ and $m5$. As expected, Condition 2 allows to only retain $m5$ as well formed. Indeed $m3$ and $m4$ do not assert that class *Boiler* is put in a package (possibly HHS).

Then the definition of a well formed member model of a metamodel:

Definition 4
A model $m = (\widetilde{m}, \sqsubseteq_m)$ *member of a metamodel* $M = (\widetilde{M}, \sqsubseteq_M)$ *is well formed w.r.t it iff:*

- $closure_{\sqsubseteq_M}(meta(\widetilde{m})) = meta(\widetilde{m})$
- $\forall x \in \widetilde{m}, \forall Y \in \widetilde{M}, meta(x) \sqsubseteq_M Y \Rightarrow \exists y \in \widetilde{m} \mid meta(y) = Y, x \sqsubseteq_m y.$

5.2 Model well-formedness and the concept of proper metamodel

The concept of proper metamodel of a model was introduced (Section 3.2) as a sufficiently identifying characteristic of (sub) models at the meta level when their relation to (sub) metamodels are under question. This was exploited in Section 4 about the question of (sub) metamodel membership of (sub) models. It was shown (Property 2) that membership of models is equivalent to metamodel inclusion of their proper metamodel. Application to submodels and submetamodels was then derived.

Let us do the same analysis for the present problem of (sub) model well formedness w.r.t (sub) metamodels. As a major result, it will be shown (major Property 8) that well formedness of a model, say m, w.r.t a metamodel, say M, is equivalent to well formedness of m w.r.t its proper metamodel *plus* the invariance of this proper metamodel in M.

For that, examine the situation progressively. This will lead to state intermediate useful properties. We saw in Section 4.1 that, by construction, a model is member of its proper metamodel. Though, it is important to note that it is not necessarily well formed w.r.t it. Compare model samples of Figure 10 to their respective proper metamodel:

- models $m0$ to $m3$ and $m5$ are well formed w.r.t their proper metamodel
- but $m4$ is not and it is a counterexample: class *Boiler* is not put in a package (possibly HHS). This meta constraint was retained by the proper metamodel of $m4$ due to another part of this model, namely class *Radiator* being in package HHS.

Now consider any metamodel, such as MM of Fig. 8 and the question of well-formedness of member models, such as preceding samples (Figure 10), by comparing their proper metamodel to MM:

- $m0$ is not well formed w.r.t MM (attribute *temperature* must be owned by a class which must be owned by a package) and $\widehat{m0}$ is only included in MM but not closed: *Class* and *Package* are lacking.
- $m1$ is not well formed (attribute *temperature* must be owned by a class, possibly *Boiler*, which must be owned by a package which is lacking) and $\widehat{m1}$ is only included in MM but not closed: *Package* is lacking.
- $m2$ is not well formed (class *Boiler* must be owned by a package which is lacking) and $\widehat{m2}$ is only included in MM but not closed: *Package* is lacking.
- $m3$ is not well formed (class *Boiler* must be owned by a package, possibly HHS). This time $\widehat{m3}$ is closed in MM but only covariant, not invariant, in MM: constraint $(Class, Package)$ is lacking (see Figure 18).
- $m4$ and $m5$ have the same proper metamodel which is an invariant submodel of MM (see Figure 18) but $m5$ is well formed w.r.t this metamodel contrary to $m4$ (class *Boiler* must be owned by a package, possibly HHS).

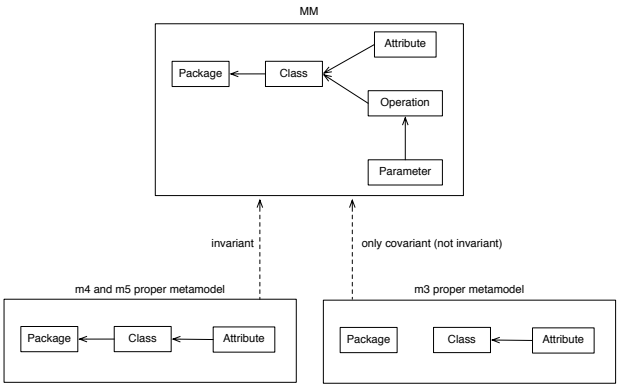

Fig. 18 Proper metamodels of models $m3, m4, m5$ compared to MM metamodel

From these examples, we can observe that model well-formedness w.r.t a metamodel is closely related to submodel invariance of their proper metamodel in this metamodel at the meta level. It is established by following Property 6. Model $m5$ is well formed w.r.t metamodel MM and indeed its proper metamodel is an invariant submodel of MM. Proper metamodels of models $m0$ to m3 are not invariant in MM and indeed these models are not well formed w.r.t MM. Model $m4$ is off the scope of the property and it will be examined afterwards. Intuitively it exemplifies that submodel invariance of proper metamodels in a metamodel (within the meta level) is necessary but, of course, not sufficient to guaranty model well-formedness at the model level.

Property 6 Let M a metamodel of a model m, m is well formed w.r.t $M \Rightarrow \widehat{m}$ is an invariant submodel of M.

Proof $\widehat{m}$ is an invariant submodel of M iff:

1. $closure_{\sqsubseteq_M}(\widetilde{\widehat{m}}) = \widetilde{\widehat{m}}$
2. $\forall X, Y \in \widetilde{\widehat{m}} \mid X, Y \in \widetilde{M}, X \sqsubseteq_{\widehat{m}} Y \Leftrightarrow X \sqsubseteq_M Y$

Proof of 1

– $\widetilde{\widehat{m}} = meta(\widetilde{m})$ by definition of $\widehat{m}$ being the proper metamodel of m so that:
$closure_{\sqsubseteq_M}(\widetilde{\widehat{m}}) = closure_{\sqsubseteq_M}(meta(\widetilde{m}))$
– but $closure_{\sqsubseteq_M}(meta(\widetilde{m})) = meta(\widetilde{m})$ by definition of m being a well formed member model of M.

□

Proof of 2
$X \sqsubseteq_{\widehat{m}} Y \Rightarrow X \sqsubseteq_M Y$ is stated by definition of $\widehat{m}$ being the proper metamodel of m. It remains to prove that: $\forall X, Y \in \widetilde{\widehat{m}} \mid X, Y \in \widetilde{M}, X \sqsubseteq_M Y \Rightarrow X \sqsubseteq_{\widehat{m}} Y$.

By definition of $\widehat{m}$ being the proper metamodel of m which is member of M, $\widetilde{\widehat{m}} \subseteq \widetilde{M}$ so that $X, Y \in \widetilde{\widehat{m}} \Rightarrow X, Y \in \widetilde{M}$. Now suppose $\exists\, X, Y \in \widetilde{\widehat{m}} \subseteq \widetilde{M}, X \sqsubseteq_M Y$ and not $X \sqsubseteq_{\widehat{m}} Y$.

– $X \in \widetilde{\widehat{m}} \Rightarrow \exists\, x \in \widetilde{m} \mid meta(x) = X$
– m being well formed w.r.t M, $x \in \widetilde{m}$ and $meta(x) = X \sqsubseteq_M Y \Rightarrow \exists\, y \in \widetilde{m} \mid meta(y) = Y, x \sqsubseteq_m y$
– by definition of $\widehat{m}$ being the proper metamodel of m: $x \sqsubseteq_m y \Rightarrow meta(x) = X \sqsubseteq_{\widehat{m}} meta(y) = Y$, then the contradiction.

□

Property 6 is important because it gives the basis to establish the relationship between model well-formedness w.r.t a metamodel and proper metamodel relative positioning. Though, it is worth noting that invariant inclusion of the proper metamodel of a model in a metamodel is not sufficient to guaranty its well formedness w.r.t this metamodel. A counterexample is $m4$: its proper metamodel is invariant in MM but $m4$ is not well formed w.r.t MM. Then the question: Compared to $m5$ which shares the same proper metamodel as $m4$, what is missing to $m4$ to be well formed w.r.t MM? Intuitive answer is the well-formedness or not of models w.r.t to their proper metamodel: contrary to $m5$, $m4$ is not well formed w.r.t their common proper metamodel. And it is indeed the missing condition as established by the following property.

Property 7 Let M a metamodel, m a model, m is a well formed member model of $\widehat{m}$ and $\widehat{m}$ is an invariant submodel of $M \Rightarrow m$ is a well formed member model of M.

Proof
Thanks to Property 2, m being a member model of $\widehat{m}$ and $\widehat{m}$ being a submodel of M, m is member of M. It remains to show that it is also well formed w.r.t M, that is:

1. $meta(\widetilde{m})$ is closed in M
2. $\forall\, x \in \widetilde{m}, \forall\, Y \in \widetilde{M}$
$meta(x) \sqsubseteq_M Y \Rightarrow \exists\, y \in \widetilde{m} \mid meta(y) = Y, x \sqsubseteq_m y$.

Proof of 1
This directly comes from the invariance of $\widehat{m}$ in M which ensures that the set of model elements of $\widehat{m}$ ($\widetilde{\widehat{m}}$) is closed in M. Knowing that $\widetilde{\widehat{m}} = meta(\widetilde{m})$ by definition of $\widehat{m}$ gives the result.

□

Proof of 2
By contradiction suppose $\exists\, x \in \widetilde{m}, Y \in \widetilde{M}$ such as $meta(x) \sqsubseteq_M Y$ but $\nexists\, y \in \widetilde{m} \mid meta(y) = Y, x \sqsubseteq_m y$.

– $x \in \widetilde{m}$, $meta(x) \in meta(\widetilde{m}) = \widetilde{\widehat{m}}$ by definition of $\widehat{m}$
– $\widehat{m}$ being a submodel of M, $\widetilde{\widehat{m}} \subseteq \widetilde{M}$ then $meta(x) \in \widetilde{M}$
– $meta(x) \in \widetilde{M}, meta(x) \sqsubseteq_M Y$ (by supposition) $\Rightarrow$ $Y \in closure_{\sqsubseteq_M}(\widetilde{\widehat{m}})$
– $\widehat{m}$ being given as an invariant submodel of M:
  – $closure_{\sqsubseteq_M}(\widetilde{\widehat{m}}) = \widetilde{\widehat{m}}$ so that $Y \in \widetilde{\widehat{m}}$
  – $meta(x), Y \in \widetilde{\widehat{m}} \subseteq \widetilde{M}$:
  $meta(x) \sqsubseteq_M Y \Rightarrow meta(x) \sqsubseteq_{\widehat{m}} Y$
– but m is also given as well formed w.r.t $\widehat{m}$ so that $x \in \widetilde{m}, Y \in \widetilde{\widehat{m}}, meta(x) \sqsubseteq_{\widehat{m}} Y$
$\Rightarrow \exists\, y \in \widetilde{m} \mid meta(y) = Y, x \sqsubseteq_m y$
then the contradiction.

□

Moreover, equivalency can be stated between model well formedness w.r.t a metamodel and model well formedness w.r.t to their proper metamodel *plus* submodel invariance of these proper metamodels in the metamodel. This is stated below by announced major Property 8 which completes the preceding one (Figure 19).

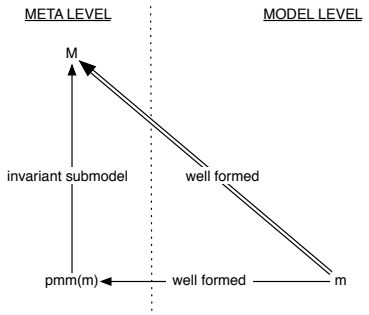

Fig. 19 metamodel well-formedness = pmm well-formedness + pmm invariance (Property 8)

Property 8 Let M a metamodel, m a model, m is a well formed member model of $\widehat{m}$ and $\widehat{m}$ is an invariant submodel of $M \Leftrightarrow m$ is a well formed member model of M.

Proof
Necessary side is Property 7. Let us prove the sufficient side. m being a well formed member model of M, $\widehat{m}$ is an invariant submodel of M thanks to Property 6. It remains to show that m is well formed w.r.t $\widehat{m}$, that is:

1. $meta(\widetilde{m})$ is closed in $\widehat{m}$: This is trivial since, by definition of $\widehat{m}$, $meta(\widetilde{m}) = \widetilde{\widehat{m}}$.
2. $\forall x \in \widetilde{m}, \forall Y \in \widetilde{\widehat{m}}$, $meta(x) \sqsubseteq_{\widehat{m}} Y \Rightarrow \exists y \in \widetilde{m} \mid meta(y) = Y, x \sqsubseteq_m y$:
   - $x \in \widetilde{m} \Rightarrow meta(x) \in \widetilde{\widehat{m}}$ (by definition of $\widehat{m}$).
   - $meta(x), Y \in \widetilde{\widehat{m}}$ and $\widehat{m}$ being a submodel of M (as seen above thanks to Property 6) : $meta(x), Y \in \widetilde{M}$ and $meta(x) \sqsubseteq_{\widehat{m}} Y$ $\Rightarrow meta(x) \sqsubseteq_M Y$.
   - but m is well formed w.r.t M so that: $x \in \widetilde{m}, Y \in \widetilde{M}, meta(x) \sqsubseteq_M Y$ $\Rightarrow \exists y \in \widetilde{m} \mid meta(y) = Y, x \sqsubseteq_m y$ then the result.

□

Examples
Let us verify the property on the preceding model samples:

- Model $m5$ is well formed w.r.t MM and indeed it is well formed w.r.t $\widehat{m5}$ which is invariant in MM
- Model $m4$ is not well formed w.r.t $\widehat{m4} = \widehat{m5}$ and indeed $m4$ is not well formed w.r.t MM.
- Models $m0$ to $m3$ are well formed w.r.t their proper metamodel but these metamodels are not invariant submodels of MM and indeed these models are not well formed w.r.t MM.

As a conclusion, Property 8 is essential because it alternatively defines model well-formedness w.r.t a metamodel only by way of the simple, though essential, core concept of proper metamodel of a model. Far beyond the theoretical result, it is particularly effective at a practical level to simplify well-formedness checking w.r.t an overall reference metamodel, say M, because:

- In the checking, it allows to replace overall metamodel M by the proper metamodel of the candidate model which is the smallest submetamodel of M whose it is a member, so minimizing the operation.
- Testing the invariance of a submetamodel can be done only once and for all for the benefit of all models which share the same proper metamodel.

Such a factorization is effective compared to checking the well-formedness of candidate models taken in isolation with reference to the whole metamodel each time[8].

5.3 Model well-formedness w.r.t submetamodels

In the preceding, model well-formedness w.r.t a metamodel was defined and characterized thanks to the concept of proper metamodel. It was shown (Property 8) that well-formedness w.r.t a metamodel is directly related to submodel invariance of proper metamodels in this metamodel.

Now what about model well-formedness w.r.t submetamodels, that is:

- How far well formed models w.r.t submetamodels of a metamodel can be guaranteed to be well formed w.r.t this metamodel?
- Conversely, considering well formed member models of a metamodel, w.r.t which submetamodels they can be also guaranteed to be well formed?

Consider well formed models w.r.t to submetamodels. It is interesting to point out that, thanks to submodel invariance transitivity (Property 3 of [12]) applied at the meta level, well formed models of invariant submetamodels of a metamodel are guaranteed to be well formed in this metamodel and recursively. It is stated by following Property 9 (see Figure 20).

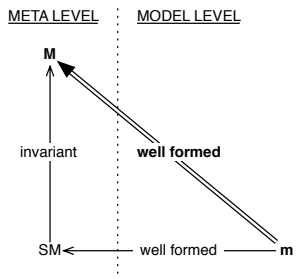

Fig. 20 Model well-formedness due to invariant submetamodels (Property 9)

Property 9 Let M and SM two metamodels, m a well formed member model of SM, SM is an invariant submetamodel of $M \Rightarrow m$ is a well formed member model of M.

Proof
Thanks to Property 8, m being well formed w.r.t SM, it is well formed w.r.t $\widehat{m}$ and $\widehat{m}$ is an invariant submodel of SM. But SM is invariant in M so that $\widehat{m}$ is itself invariant in M thanks to invariance transitivity (Property 3 of [12]). Then apply Property 8 again: m is well formed w.r.t $\widehat{m}$ and $\widehat{m}$ is an invariant submodel of $M \Rightarrow m$ is well formed w.r.t M.

□

Let us insist on the fact that submetamodel invariance of SM in M is necessary for the property to apply. Otherwise, well formed member model m of SM cannot be guaranteed to be well formed in M. As a counterexample, see metamodel in Figure 21 which is a submetamodel of initial metamodel MM (Figure 8). This submetamodel is only covariant but not invariant in MM

[8] See quantitatitave evaluation of the results in Section 6.3.

(classes are not constrained to be put in a package). Sample model $m3$ of Figure 10 is well formed w.r.t this submetamodel but, indeed, it is not well formed w.r.t MM. We let you verify that $m3$ is well formed w.r.t its proper metamodel which is invariant in the submetamodel but not in MM.

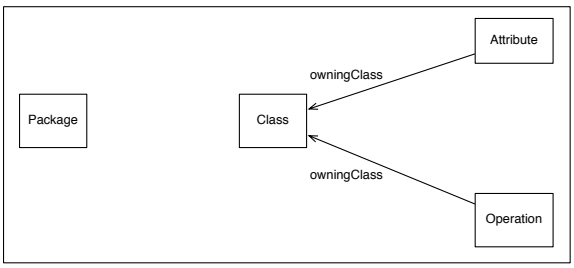

Fig. 21 Not invariant submetamodel of MM (Fig. 8)

Conversely, take well formed models of a metamodel. What about their well-formedness w.r.t submetamodels of this metamodel? Following Property 10 ensures that such models are also well formed w.r.t any submetamodel whose they are simply member.

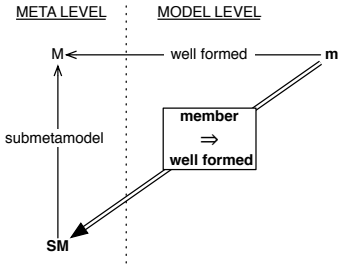

Fig. 22 Well-formedness w.r.t submetamodels (Prop. 10)

Property 10 Let M a metamodel, SM a submetamodel of M, m a well formed member model of M:
m is member of $SM \Rightarrow m$ is well formed w.r.t SM.

Proof
m is well formed w.r.t SM iff (Property 8):

1. m is well formed w.r.t to $\widehat{m}$: this is true because m is well formed w.r.t metamodel M whose it is a member.
2. $\widehat{m}$ is an invariant submodel of SM, indeed:
   - Thanks to Property 8, m being well formed w.r.t M, $\widehat{m}$ is an invariant submodel of M.
   - m being member of SM, $\widehat{m}$ is a submodel of SM (Property 2).
   - $\widehat{m}$ being a submodel of SM which is a submodel of M and $\widehat{m}$ being invariant in $M \Rightarrow \widehat{m}$ is an invariant submodel of SM. This is due in fact to a much more general property of submodels stated below (Property 11, sketched in Figure 23), here applied at the meta level to metamodels $\widehat{m}$, SM and M.

□

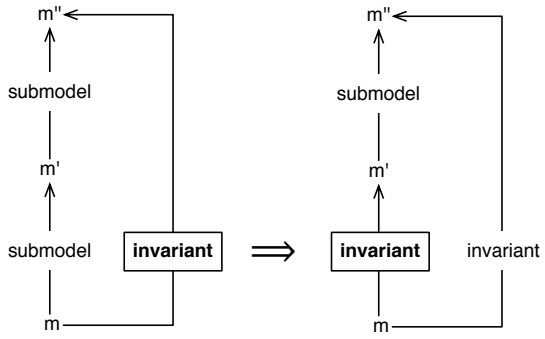

Fig. 23 Sketch of Property 11

Property 11 Let three models m, m' and m'' such as m is a submodel of m' and m' is a submodel of m'', m is an invariant submodel of $m'' \Rightarrow m$ is an invariant submodel of m'.

Proof
m is an invariant submodel of m' iff:

1. $\widetilde{m}$ is closed in m': $closure_{\sqsubseteq_{m'}}(\widetilde{m}) = \widetilde{m}$.
2. $Gr(\sqsubseteq_m) = Gr(\sqsubseteq_{m'})/\widetilde{m}$.

Proof of 1
m being a submodel of m', $\widetilde{m} \subseteq closure_{\sqsubseteq_{m'}}(\widetilde{m})$ is obvious. Let us prove that $closure_{\sqsubseteq_{m'}}(\widetilde{m}) \subseteq \widetilde{m}$.

- m' being a submodel of m'':
  $closure_{\sqsubseteq_{m'}}(\widetilde{m}) \subseteq closure_{\sqsubseteq_{m''}}(\widetilde{m})$.
- but m being an invariant submodel of m'':
  $closure_{\sqsubseteq_{m''}}(\widetilde{m}) = \widetilde{m}$.

□

Proof of 2

(a) $Gr(\sqsubseteq_m) \subseteq Gr(\sqsubseteq_{m'})/\widetilde{m}$, indeed:
   - m is a submodel of m': $Gr(\sqsubseteq_m) \subseteq Gr(\sqsubseteq_{m'})$ with $\widetilde{m} \subseteq \widetilde{m'}$ so that $Gr(\sqsubseteq_m)/\widetilde{m} \subseteq Gr(\sqsubseteq_{m'})/\widetilde{m}$
   - but $Gr(\sqsubseteq_m)/\widetilde{m} = Gr(\sqsubseteq_m)$

(b) $Gr(\sqsubseteq_{m'})/\widetilde{m} \subseteq Gr(\sqsubseteq_m)$, indeed:
   - $Gr(\sqsubseteq_{m'}) \subseteq Gr(\sqsubseteq_{m''})$, m' being a submodel of m''
   - $Gr(\sqsubseteq_{m'})/\widetilde{m} \subseteq Gr(\sqsubseteq_{m''})/\widetilde{m}$, m being a submodel of m' which is a submodel of m'' ($\widetilde{m} \subseteq \widetilde{m'} \subseteq \widetilde{m''}$)
   - but m is an invariant submodel of m'' so that $Gr(\sqsubseteq_{m''})/\widetilde{m} = Gr(\sqsubseteq_m)$, then the result.

□

It is worth noting that Property 10 applies to any submetamodel of a metamodel whose m is member with no other condition than simple inclusion in this metamodel. Of course this applies, a fortiori, to submetamodels which are closed, covariant or invariant in the metamodel.

Examples
Take model m of Figure 24 which is well formed w.r.t to metamodel MM of Figure 8. Then consider alternative submetamodels of MM of Figures 25 and 26 to check the property.

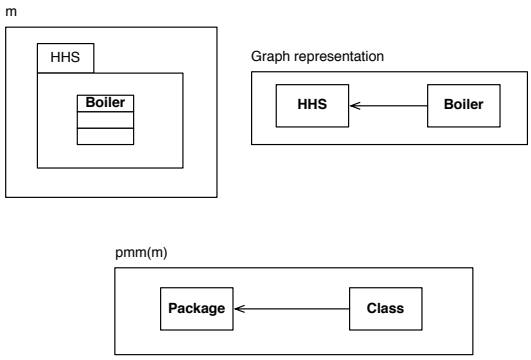

Fig. 24 Model sample (Property 11)

First, consider submetamodels of MM whose m is even not member such as $SM0$ or $SM0'$ of Figure 25. Model m is getting no chance to be well formed w.r.t them, indeed:

- metaconstruct *Package* of model element HHS of m is lacking in $SM0$ so that $meta(\widetilde{m})$, which is closed in MM, has no chance to be closed in $SM0$.
- in submetamodel $SM0'$ the metaconstruct is present, this time. So that metaconstructs of m are closed in $SM0'$. But this submetamodel does not specify the constraint that classes must be contained into packages. Though, m asserts this constraint between its class *Boiler* and package HHS. So that m, which asserts a constraint not specified by metamodel $SM0'$ is not member of it.

Now take submetamodels whose m is member and verify that it is well formed w.r.t them, whatever their inclusion qualities. See Figure 26:

- submetamodel $SM1$ is only included but not closed in MM since metaconstruct *Operation* whose *Parameter* metaconstruct depends is lacking. Model m, which is unaware of operations and their parameter constituents, is member of this submetamodel and indeed remains well formed w.r.t it.
- in submetamodel $SM2$ metaconstruct *Operation* is now present so that $SM2$ is covariant in MM: all model elements of MM whose model elements of $SM2$ depend in MM are present in $SM2$ ($\widetilde{SM2}$ is closed in MM) and asserted constraints of $SM2$ are in MM. But $SM2$ is not invariant in MM. It does not specify that *Operation* depends on the metaconstruct *Class*. Here again model m remains a well formed member model of this only covariant submetamodel.
- finally $SM3$, which adds the dependency between *Operation* and *Class*, is an invariant submetamodel of MM whose m is member and well formed again.

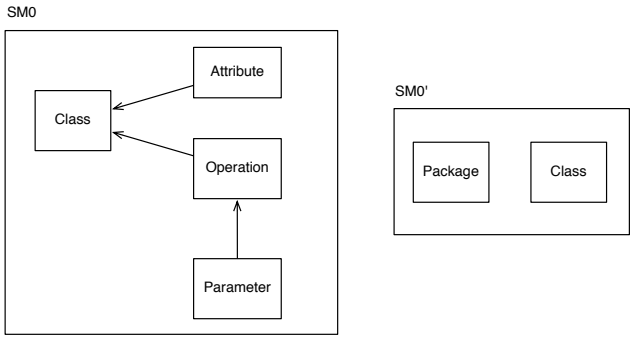

Fig. 25 m not member so not well formed w.r.t $SM0$, $SM0'$

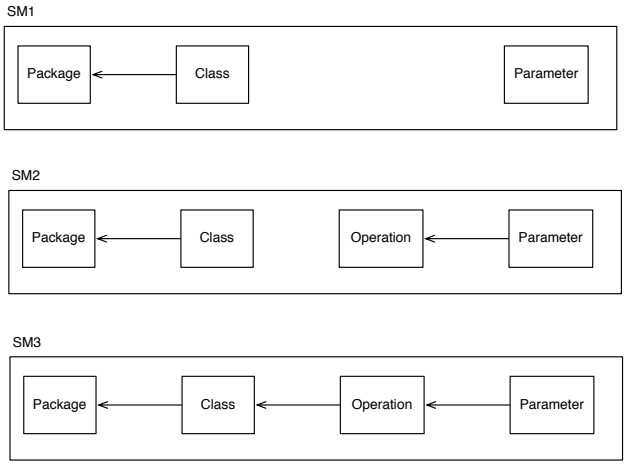

Fig. 26 Not closed ($SM1$), covariant ($SM2$) and invariant ($SM3$) submetamodels whose m is member, so well formed.

As a conclusion, let us synthesize all this in the structure of the submetamodel hierarchy of a metamodel, say M, whose a model, say m, is well formed (see Figure 27). Property 10 retains all the submetamodels of M whose m is a member, that is its membership submetamodel hierarchy (Figure 14). This includes in particular its proper metamodel and it is consistent with Property 8 : m being well formed w.r.t M, it is well formed w.r.t its proper metamodel. Proper metamodel of m has also the specific features that it is the "smallest" of these submetamodels and it is invariant in M (Property 8 again).

Recursively, this applies to submetamodels of the hierarchy so that proper metamodel of m is also invariant in all of them (but remind that themselves are not necessarily invariant in M, see discussion about Property 10).

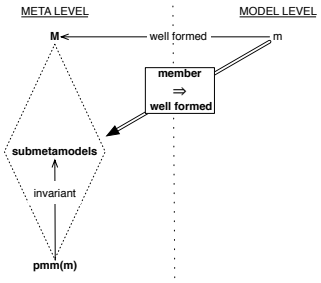

Fig. 27 Well-formedness submetamodel hierarchy structure

5.4 Submodel well-formedness w.r.t (sub) metamodels

In preceding Section 5.3 model well-formedness w.r.t submetamodels of a reference metamodel was studied. It was shown that model well-formedness w.r.t (sub) metamodels closely relates to submodel invariance and transitivity applied at the meta level by way of proper metamodels. The structure of the submetamodel hierarchy of a metamodel whose a model is well formed was characterized thanks to established properties (see Figure 27 for a synthetic view of the results).

Present section is studying the symmetrical question of well-formedness of submodels of a well formed model w.r.t a metamodel (and its submetamodels). According to the proximity between the concepts of model well-formedness and submodel invariance, intuition is that invariant submodels of a well formed model w.r.t a metamodel are also well formed. It is indeed the case and it will be stated by major Property 15 of the section. Let us establish the result progressively.

From submodel invariance to submodel well-formedness w.r.t proper metamodels

Knowing that invariance of proper metamodels in a metamodel is a condition for model well-formedness (Property 8), what about the invariance of proper metamodels of invariant submodels? Remind that invariance of submodels in a model does not automatically give the invariance of their proper metamodel in the proper metamodel of the model in the general case (see discussion about Property 1). But when the model is well formed w.r.t its proper metamodel this interestingly occurs, as established by following Property 12 (see Figure 28).

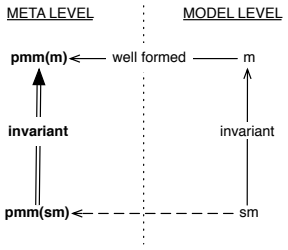

Fig. 28 Sketch of Property 12

Property 12 Let sm an invariant submodel of a model m, m is well formed w.r.t its proper metamodel ($\widehat{m}$) $\Rightarrow$ the proper metamodel of sm ($\widehat{sm}$) is an invariant submodel of the proper metamodel of m ($\widehat{m}$).

Proof
The proper metamodel of sm ($\widehat{sm}$) is an invariant submodel of the proper metamodel of m ($\widehat{m}$) iff:

1. The set of model elements of $\widehat{sm}$ ($\widetilde{\widehat{sm}}$) is closed in $\widehat{m}$, that is to say: $closure_{\sqsubseteq_{\widehat{m}}}(\widetilde{\widehat{sm}}) = \widetilde{\widehat{sm}}$.
2. $Gr(\sqsubseteq_{\widehat{m}})/\widetilde{\widehat{sm}} = Gr(\sqsubseteq_{\widehat{sm}})$.

Proof of 1
sm being a submodel of m, $\widehat{sm}$ is a submodel of $\widehat{m}$ (Property 1) so that $\widetilde{\widehat{sm}} \subseteq closure_{\sqsubseteq_{\widehat{m}}}(\widetilde{\widehat{sm}})$ is obvious. It remains to show that: $closure_{\sqsubseteq_{\widehat{m}}}(\widetilde{\widehat{sm}}) \subseteq \widetilde{\widehat{sm}}$.
By contradiction, suppose:
$\exists X \in closure_{\sqsubseteq_{\widehat{m}}}(\widetilde{\widehat{sm}}), X \notin \widetilde{\widehat{sm}}$

- $X \in closure_{\sqsubseteq_{\widehat{m}}}(\widetilde{\widehat{sm}}) \Rightarrow X \in \widetilde{\widehat{m}}$
- $X \in \widetilde{\widehat{m}} \Rightarrow \exists\, x \in \widetilde{m} \mid X = meta(x)$ by definition of $\widehat{m}$
- $X \in closure_{\sqsubseteq_{\widehat{m}}}(\widetilde{\widehat{sm}}) \Rightarrow \exists Y \in \widetilde{\widehat{sm}} \mid Y \sqsubseteq_{\widehat{m}} X$
- $Y \in \widetilde{\widehat{sm}} \Rightarrow \exists\, y \in \widetilde{sm} \mid Y = meta(y)$ by definition of $\widehat{sm}$
- $y \in \widetilde{sm} \Rightarrow y \in \widetilde{m}$, sm being a submodel of m
- $y \in \widetilde{m} \Rightarrow meta(y) = Y \in \widetilde{\widehat{m}}$ by definition of $\widehat{m}$
- $x, y \in \widetilde{m}$ and $Y \sqsubseteq_{\widehat{m}} X \Rightarrow y \sqsubseteq_m x$, m being well formed w.r.t $\widehat{m}$
- $y \in \widetilde{sm}$ and $y \sqsubseteq_m x \Rightarrow x \in closure_{\sqsubseteq_m}(\widetilde{sm})$
- but sm is invariant in m so that $closure_{\sqsubseteq_m}(\widetilde{sm}) = \widetilde{sm}$ and then $x \in \widetilde{sm}$
- so the contradiction: $x \in \widetilde{sm} \Rightarrow meta(x) = X \in \widetilde{\widehat{sm}}$ (by definition of $\widehat{sm}$).

□

Proof of 2

2.1 $Gr(\sqsubseteq_{\widehat{sm}}) \subseteq Gr(\sqsubseteq_{\widehat{m}})/\widetilde{\widehat{sm}}$.
Indeed, sm being a submodel of m, $\widehat{sm}$ is a submodel of $\widehat{m}$ (Property 1) so that $Gr(\sqsubseteq_{\widehat{sm}}) \subseteq Gr(\sqsubseteq_{\widehat{m}})$ with $\widetilde{\widehat{sm}} \subseteq \widetilde{\widehat{m}}$. Then $Gr(\sqsubseteq_{\widehat{sm}})/\widetilde{\widehat{sm}} \subseteq Gr(\sqsubseteq_{\widehat{m}})/\widetilde{\widehat{sm}}$. But $Gr(\sqsubseteq_{\widehat{sm}})/\widetilde{\widehat{sm}} = Gr(\sqsubseteq_{\widehat{sm}})$, then the result.

2.2 $Gr(\sqsubseteq_{\widehat{m}})/\widetilde{\widehat{sm}} \subseteq Gr(\sqsubseteq_{\widehat{sm}})$ which is equivalent to show that: $\forall X, Y \in \widetilde{\widehat{sm}}, X \sqsubseteq_{\widehat{m}} Y \Rightarrow X \sqsubseteq_{\widehat{sm}} Y$.

By contradiction, suppose $\exists X, Y \in \widetilde{\widehat{sm}} \mid X \sqsubseteq_{\widehat{m}} Y$ but not $X \sqsubseteq_{\widehat{sm}} Y$

- $X, Y \in \widetilde{\widehat{sm}} \Rightarrow \exists\, x, y \in \widetilde{sm} \mid X = meta(x), Y = meta(y)$
- sm being a submodel of m, $\widetilde{sm} \subseteq \widetilde{m}$ so that $x, y \in \widetilde{m}$
- sm being a submodel of m, $\widehat{sm}$ is a submodel of $\widehat{m}$ (Property 1) so that $\widetilde{\widehat{sm}} \subseteq \widetilde{\widehat{m}}$ and then $X, Y \in \widetilde{\widehat{m}}$
- $x, y \in \widetilde{m}, X = meta(x), Y = meta(y) \in \widetilde{\widehat{m}}, X \sqsubseteq_{\widehat{m}} Y \Rightarrow x \sqsubseteq_m y$, m being well formed w.r.t $\widehat{m}$
- sm being an invariant submodel of m and $x, y \in \widetilde{sm}$, $x \sqsubseteq_m y \Rightarrow x \sqsubseteq_{sm} y$
- but, by definition of $\widehat{sm}$, $x, y \in \widetilde{sm}, x \sqsubseteq_{sm} y \Rightarrow meta(x) = X \sqsubseteq_{\widehat{sm}} meta(y) = Y$

then the contradiction.

□

Examples See some samples in Figure 10:

- $m2$ is an invariant submodel of $m3$ which is well formed w.r.t its proper metamodel and indeed $\widehat{m2}$ is invariant in $\widehat{m3}$
- $\widehat{m2}$ is not invariant in $\widehat{m4}$ and indeed $m2$ is invariant in $m4$ but $m4$ is not well formed w.r.t $\widehat{m4}$
- $\widehat{m2}$ is not invariant in $\widehat{m5}$ and indeed $m5$ is well formed w.r.t $\widehat{m5}$ but $m2$ is not invariant in $m5$.

More, it appears that under the same conditions submodels are necessarily well formed w.r.t their proper metamodel (Figure 29).

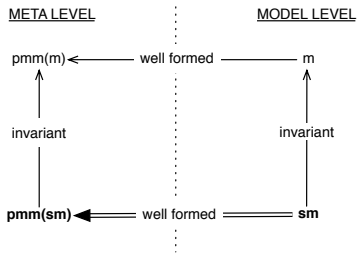

Fig. 29 Sketch of Property 13

Before establishing the property, take the following examples:

- $m2$ is indeed well formed w.r.t its proper metamodel
- $m4$ is not well formed w.r.t its proper metamodel ($\widehat{m4}$), indeed it is not invariant in $m5$ which is well formed w.r.t $\widehat{m5}$ ($= \widehat{m4}$).

Property 13 Let sm an invariant submodel of a model m, m is well formed w.r.t its proper metamodel ($\widehat{m}$) $\Rightarrow$ sm is well formed w.r.t its proper metamodel ($\widehat{sm}$).

Proof
sm is well formed w.r.t $\widehat{sm}$ iff:

1. $meta(\widetilde{sm})$ is closed in $\widehat{sm}$. This is trivial since, by definition of $\widehat{sm}$, $meta(\widetilde{sm}) = \widetilde{\widehat{sm}}$.
2. $\forall\, x \in \widetilde{sm}, \forall\, Y \in \widetilde{\widehat{sm}}$
   $meta(x) \sqsubseteq_{\widehat{sm}} Y \Rightarrow \exists\, y \in \widetilde{sm} \mid meta(y) = Y, x \sqsubseteq_{sm} y$.
   Indeed:
   - sm being a submodel of m, $\widehat{sm}$ is a submodel of $\widehat{m}$ (Property 1) so that $\widetilde{\widehat{sm}} \subseteq \widetilde{\widehat{m}}$ and then: $meta(x), Y \in \widetilde{\widehat{sm}} \Rightarrow meta(x), Y \in \widetilde{\widehat{m}}$.
   - Thanks to Property 12 (conditions being the same), $\widehat{sm}$ is an invariant submodel of $\widehat{m}$ so that: $meta(x), Y \in \widetilde{\widehat{m}}, meta(x) \sqsubseteq_{\widehat{sm}} Y$ $\Rightarrow meta(x) \sqsubseteq_{\widehat{m}} Y$
   - $x \in \widetilde{sm} \subseteq \widetilde{m}$ (sm is a submodel of m) $\Rightarrow x \in \widetilde{m}$
   - $x \in \widetilde{m}, Y \in \widetilde{\widehat{m}}, meta(x) \sqsubseteq_{\widehat{m}} Y$ $\Rightarrow \exists\, y \in \widetilde{m} \mid meta(y) = Y, x \sqsubseteq_m y$ m being given as well formed w.r.t $\widehat{m}$
   - $x \in \widetilde{sm}, y \in \widetilde{m}, x \sqsubseteq_m y \Rightarrow y \in closure_{\sqsubseteq_m}(\widetilde{sm})$
   - But sm is an invariant submodel of m so that:
     - $closure_{\sqsubseteq_m}(\widetilde{sm}) = \widetilde{sm}$ and then $y \in \widetilde{sm}$
     - $x, y \in \widetilde{sm}, x \sqsubseteq_m y \Rightarrow x \sqsubseteq_{sm} y$.

   This ends the proof.

□

Finally, an immediate consequence is that such submodels of a model are also guaranteed to be well formed w.r.t the proper metamodel of this overall model, mainly thanks to major Property 8 (see Figure 30).

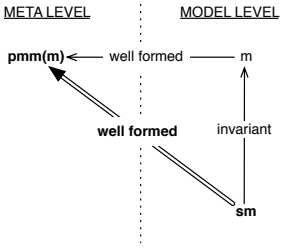

Fig. 30 Sketch of Property 14

Property 14 Let sm an invariant submodel of a model m, m is well formed w.r.t its proper metamodel ($\widehat{m}$) $\Rightarrow$ sm is well formed w.r.t the proper metamodel of m ($\widehat{m}$).

Proof
Due to Property 8, for sm to be well formed w.r.t $\widehat{m}$, its proper metamodel ($\widehat{sm}$) must be an invariant submodel of $\widehat{m}$ and sm must be well formed w.r.t it. Both features are verified thanks respectively to previously established Properties 12 and 13 which share the same conditions.

□

As an intermediate conclusion, Figure 31 synthesizes the results.

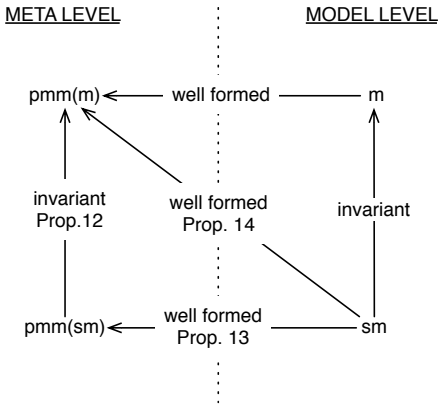

Fig. 31 Invariant submodels and their well-formedness

From submodel invariance to submodel well-formedness w.r.t a reference metamodel

Now consider any metamodel, say an overall metamodel of a project, and a well formed member model of this metamodel. Preceding properties allow to obtain major Property 15, as initially announced: invariant submodels of the model are themselves well formed w.r.t this metamodel.

Property 15 Let m a well formed model of a metamodel M, sm is an invariant submodel of $m \Rightarrow sm$ is well formed w.r.t M.

Proof

- Thanks to Property 8, m being well formed w.r.t M, its proper metamodel $\widehat{m}$ is an invariant submodel of M and m is well formed w.r.t $\widehat{m}$.
- sm being an invariant submodel of m which is well formed w.r.t $\widehat{m}$, it is well formed w.r.t $\widehat{m}$, thanks to Property 14.
- sm being well formed w.r.t $\widehat{m}$ which is an invariant submodel of M, sm is well formed w.r.t M thanks to Property 9.

□

By way of invariance transitivity applied at the meta level, another proof is the following which shows the consistency of the results, as sketched in Figure 32:

- Thanks to Property 8, m being well formed w.r.t M, its proper metamodel $\widehat{m}$ is an invariant submetamodel of M and m is well formed w.r.t $\widehat{m}$.
- sm being an invariant submodel of m which is well formed w.r.t $\widehat{m}$, the proper metamodel of sm ($\widehat{sm}$) is an invariant submodel of $\widehat{m}$ (Property 12) and sm is well formed w.r.t $\widehat{sm}$ (Property 13).
- $\widehat{sm}$ being an invariant submodel of $\widehat{m}$ and $\widehat{m}$ being an invariant submodel of M, $\widehat{sm}$ is an invariant submodel of M, thanks to invariance transitivity.
- sm being well formed w.r.t $\widehat{sm}$ which is invariant in M, sm is well formed w.r.t M thanks to Property 8.

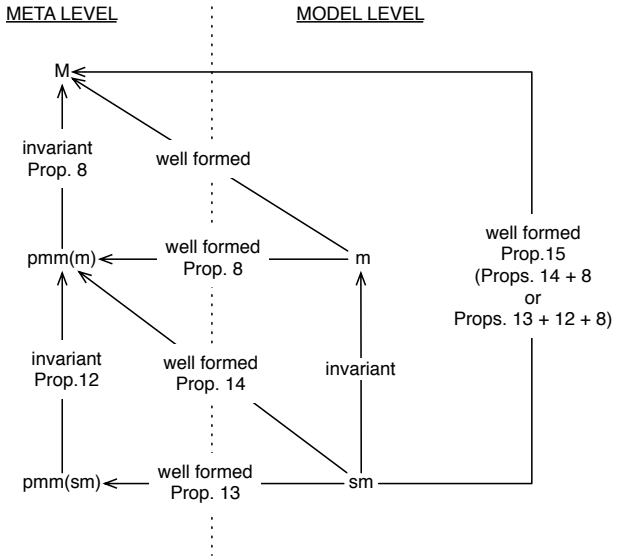

Fig. 32 Invariant submodels and their well-formedness

Finally, thanks once more to invariance transitivity at the model level this time, well-formedness applies recursively to enclosed invariant submodels as stated by the following corollary of Property 15.

Property 16 (Corollary of Property 15) Let m a well formed member model of a metamodel M, sm an invariant submodel of m and ssm a submodel of sm, ssm is an invariant submodel of $sm \Rightarrow ssm$ is well formed w.r.t M.

Proof

- Thanks to invariance transitivity: ssm being invariant in sm which is invariant in $m \Rightarrow ssm$ is invariant in m
- Thanks to preceding Property 15: ssm being invariant in m which is a well formed member model of $M \Rightarrow ssm$ is well formed w.r.t M.

□

This corollary appears to be evident at a theoretical level. But, far beyond this consideration, it is very effective at a practical engineering level when transitivity applies:

- Modularity: consider a modeling space determined by a metamodel M and m a well formed member model of M. Assuming the invariance of some of its submodels, say sm, was verified once and for all, only testing the invariance in sm of any of its proper submodels is consequently sufficient to check their well-formedness (without referring neither to m nor to the general metamodel M).

– This applies notably when modeling in the large and the need for partitioning projects into smaller manageable ones. Verifying the well-formedness of a subproject model once and for all allows to reduce checking for the well-formedness of any of its model parts to only testing its invariance in this subproject model.
– As far as computing efficiency is concerned, it is prominent. Indeed, checking submodel invariance (which only relies on set inclusion tests, what's more, circumscribed locally only to the subproject model under consideration) is more efficient than checking full well-formedness w.r.t the overall metamodel[9].

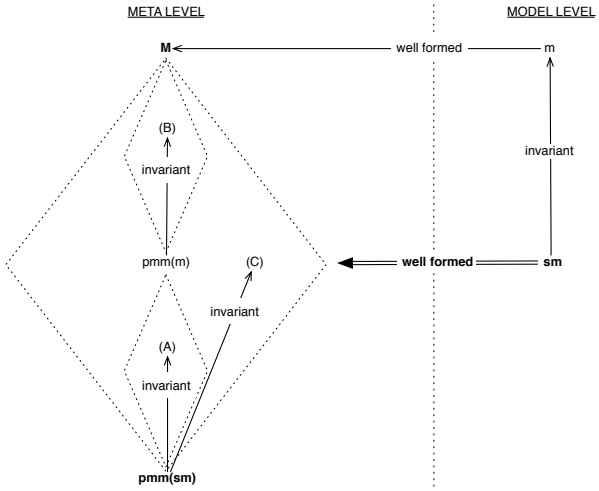

Fig. 33 Submetamodel well-formedness hierarchy

As a conclusion, in line with the characterization of the submetamodel membership hierarchy of submodels (Figure 16), a synopsis of the structure of the submetamodel well-formedness hierarchy of invariant submodels (say sm) of a well formed model (say m) w.r.t a reference metamodel (say M) is depicted in Figure 33.

sm being an invariant submodel of m, it is well formed w.r.t its proper metamodel $pmm(m)$ (Property 14). Then, as a member of any submetamodel of $pmm(m)$ which enclose its proper metamodel (Properties 3 and 4), sm is also well formed w.r.t them (Property 10). $pmm(sm)$ is the smallest one and is invariant in them all (zone A in the figure). Schematically, it is the application of Figure 27 to model sm relatively to metamodel $pmm(m)$.

Under the conditions, the proper metamodel of sm is invariant in $pmm(m)$ (Property 12) and transitively in any submetamodel of M (including M itself) in which $pmm(m)$ is invariant (zone B). So that sm being well formed w.r.t $pmm(sm)$ (Property 13), it is also well formed w.r.t these submetamodels (Property 8). Metamodels of zone A are submodels of $pmm(m)$ and then of metamodels of zone B. But remind (discussion about Property 10) that they are not necessarily invariant in $pmm(m)$. A fortiori, there is no reason for them to be invariant in metamodels of zone B. To be convinced, see Figure 34: sm is well formed w.r.t submetamodel SMA of $pmm(m)$ (in which $pmm(sm)$ is invariant), sm is well formed w.r.t submetamodel SMB of M in which $pmm(m)$ is invariant. Though SMA is not invariant in SMB (the dependency between *Operation* and *Class* metaconstructs constrained by SMB is lacking in SMA).

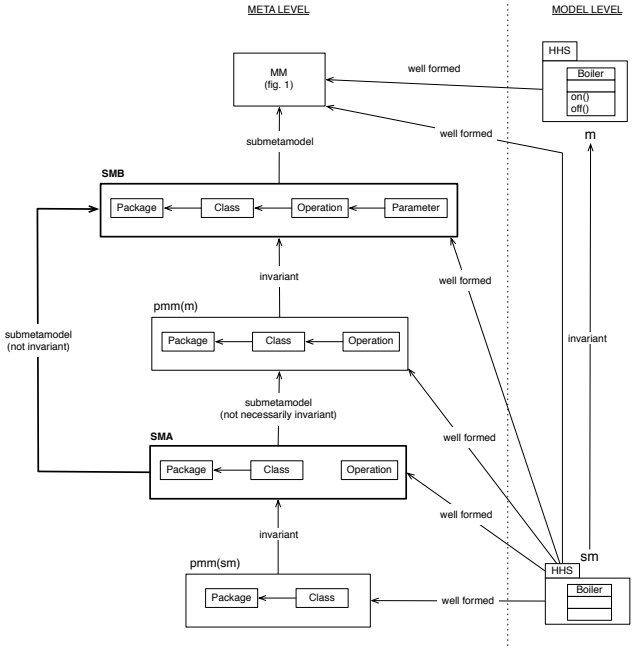

Fig. 34 Submetamodels of zone A of Fig. 33 compared to those of zone B.

Finally, as an invariant submodel of m which is well formed w.r.t M, sm is well formed w.r.t M (Property 15) and schema of Figure 27 applies once more, this time at the global level of M: sm is well formed w.r.t all submetamodels of M which enclose its proper metamodel. $pmm(sm)$ is the smallest one and is invariant in all of them. Note that this includes preceding ones (zones A and B) but not exclusively. Indeed, sm may be well formed w.r.t submetamodels of M which are not in A or B (zone C). An example is given in Figure 35 in reference with overall metamodel MM of Figure 8. Model m is well formed w.r.t MM, sm is invariant in m and take submetamodel SM of MM. sm is well formed w.r.t SM ($pmm(sm)$ is invariant in SM) which is neither a submetamodel of $pmm(m)$ nor encloses it: m is not not well formed w.r.t SM (even more it is not member of SM).

[9] See quantitatitave evaluation of the results in Section 6.3.

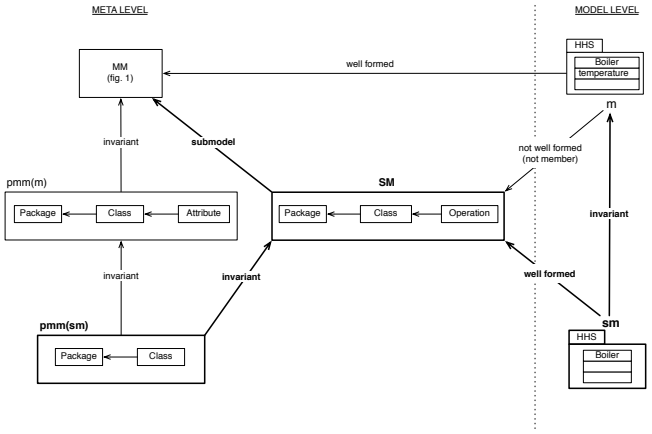

Fig. 35 *sm* well formed w.r.t SM outside the sphere of $pmm(m)$

6 Application and technology

This section presents an application of the previous results to provide reusable technology in EMF. It is followed by the description of a model search tool that profits from the offered facilities.

6.1 Sub (meta) model engine

In [12], we presented an extensible submodel engine that operates on EMF models at the model level, being well formed or not. This engine provides a set of core functionalities to determine submodel relationships, ranging from simple inclusion of element sets to invariant submodels. In the following, we will not present this engine again but the interested reader can find a detailed description of its extensible plugin-based architecture and typical scenarios and tools in the afore reference. Here, we will focus on the specific functionalities and capacities which deal with meta level constructs and relationships considered in the present paper: (sub) metamodels, proper metamodel, membership and well-formedness.

Concerning the representation capacities, the engine enables representing EMF metamodels in addition to models using the same unified formalism, that is as a set of model elements and a partial order derived from their dependency constraints. The engine also ensures consistency of the typing relationships (under function *meta*) between models and metamodels represented this way. At the operational level, thanks to the unified representation, all submodel functionalities provided by the initial engine for models are also available for metamodels, so that they can be compared under the same submodel inclusion qualities models were.

The engine offers customization capacities for adapting the representation scheme of EMF models (being meta or not) in the unified formalism. This is done through plugin-based extensions of the engine whose architecture is shown in Figure 36. The chosen representation can be more or less rich depending on the complexity of models or the need to concentrate on some specific parts while omitting others. In case studies, we use the customization capacities when applying the formalism to Ecore and UML metamodels.

Operations are offered to :

- determine if a model is a member of a metamodel;
- determine if a model is a well formed model of a metamodel;
- compute the proper metamodel of a model;

They complete those dedicated to submodels in the basic engine. Functionalities at the model and metamodel levels are fully compatible with each other so that they can be combined to apply all the properties defined in the paper. Thanks to transitivity, these functionalities can be iterated efficiently to build powerful functionalities dealing with hierarchies of models and metamodels as studied in Sections 4.3 and 5.4.

All these functionalities are available as a service offered by the engine to other plugins so that they can be easily integrated and reused into modeling tools which are compliant with the Eclipse architecture and its EMF framework. Kinds of applications or tools that can profit from this service are for example transformation engine, rich content-driven model querying and completion, submodel recommendation engines and engines for automatic classification of models in large-scale repositories. In the next section 6.2, we present an application of the previous technology to improve the task of searching submodels and enclosing models in repositories.

6.2 Application to model searching

On top of the previous engine, we have built a tool for searching EMF models stored in CDO repositories. The tool performs content-based model search through inclusion properties and adopts query by example approach. The latter means that queries are provided as models or model fragments[10]. Given a query, the tool is able to retrieve models from the repository that are either included by this model or contain it. Figure 37 presents a snapshot of the tool integrated in Eclipse for searching models. The upper view shows the content of an edited model where the user may select some elements to constitute its query. The lower view shows the result, that is the set of models satisfying the query. In addition, to present ranked resulting models, the view visualizes their inclusion relationship as shown in the figure. This graphical representation allows to visualize the network of resulting models in order to quickly focus on the most

[10] Thanks to the formalism, query model can be full well formed models but also unspecified model fragments such as simple sets of model ingredients or partial models resulting from incremental or intermediate design.

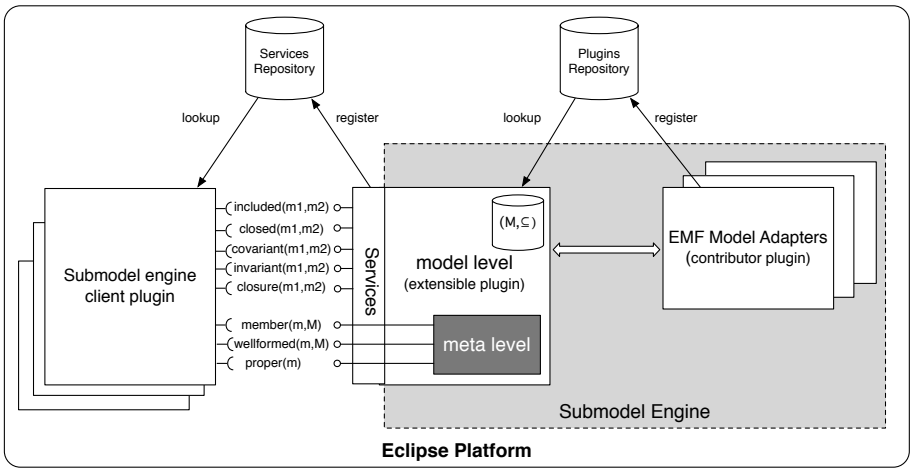

Fig. 36 Engine Architecture

interesting ones regarding the query (most specific submodels and least general enclosing ones).

Thanks to the facilities offered by this technology, our tool provides several kinds of model search which are not generally supported by other model search engines. They are summarized in Table 1. Each search in this table is presented as a function which takes one or two input parameters corresponding to a query model or a query metamodel and returns a set of models or metamodels from the repository for result. Table 1 also gives a formal definition of each function and a summary of its purpose.

The basic model searches offered by the tool corresponds to S_{sup} and S_{sub} functions in Table 1. These two searches are dedicated to model searching at one model level. They are exclusively based on submodel inclusion properties. The first one consists in searching models in the repositories that include a given query model. It is the search illustrated in Figure 37. Presented results in left part of the figure are models that include the model part selected by the user. The second model search is the symmetric of the first one. Given a model, it allows to search models in the repository that constitute its parts. Such searching may be interesting to verify that candidate parts to be registered are not already in the repository. In both cases, the search is performed by inclusion comparison between the query model and candidate models stored in the repository. Regarding these two kinds of searches, it is worthwhile to note that thanks to our formalism, their capacities are also provided at the meta level so that it is possible to search metamodels using metamodel fragments as queries.

In addition to preceding searches, the tool also supports two other ones relating to the modeling and metamodeling levels. In Table 1, S_{meta} and S_{wfm} functions represent these searches. The first enables searching metamodels for which the query model is well formed. This kind of search can be useful for determining the compatibility of models (used as query) with respect to some metamodel existing in the repository (or some compliant tools) or for organizing them according to their compatibility with some metamodels. As indicated by the formal definition, this search relies on invariance property of the proper metamodel of the query to verify its well-formedness w.r.t. metamodels. Second model search works symmetrically, offering metamodel-based search. It allows searching models in the repository which are well formed with respect to a given query (sub)-metamodel. Such capacities are helpful in practice. For instance, consider the search of class models in a repository containing UML models of any kinds as offered by this search.

Last couple of model searches in Table 1 aims to combine the capacities of model searches based on inclusion at both modeling and metamodeling level. They correspond to S_{sup_meta} and S_{sup_proper} functions in Table 1. First function accepts an input query model and an input metamodel as parameters. It allows to search models that include a query model while respecting the structure of a given metamodel. Such kind of search can be helpful to find models in the repository that can complete the contents of an edited model but without compromising its well-formedness. Second function retrieves models containing the query model and having compatible proper metamodels regarding inclusion. The latter search can be of interest for finding models which are richer than the query one while preserving some metamodel compatibility.

At the end of the search process, the tool orders resulting models or metamodels. This ordering relies on inclusion properties that may exist between these models. Results satisfying the query model are consequently organized in a net of relationships and exhibit characteristic ones such as the simplest or the richest submodel as illustrated previously through Figure 37.

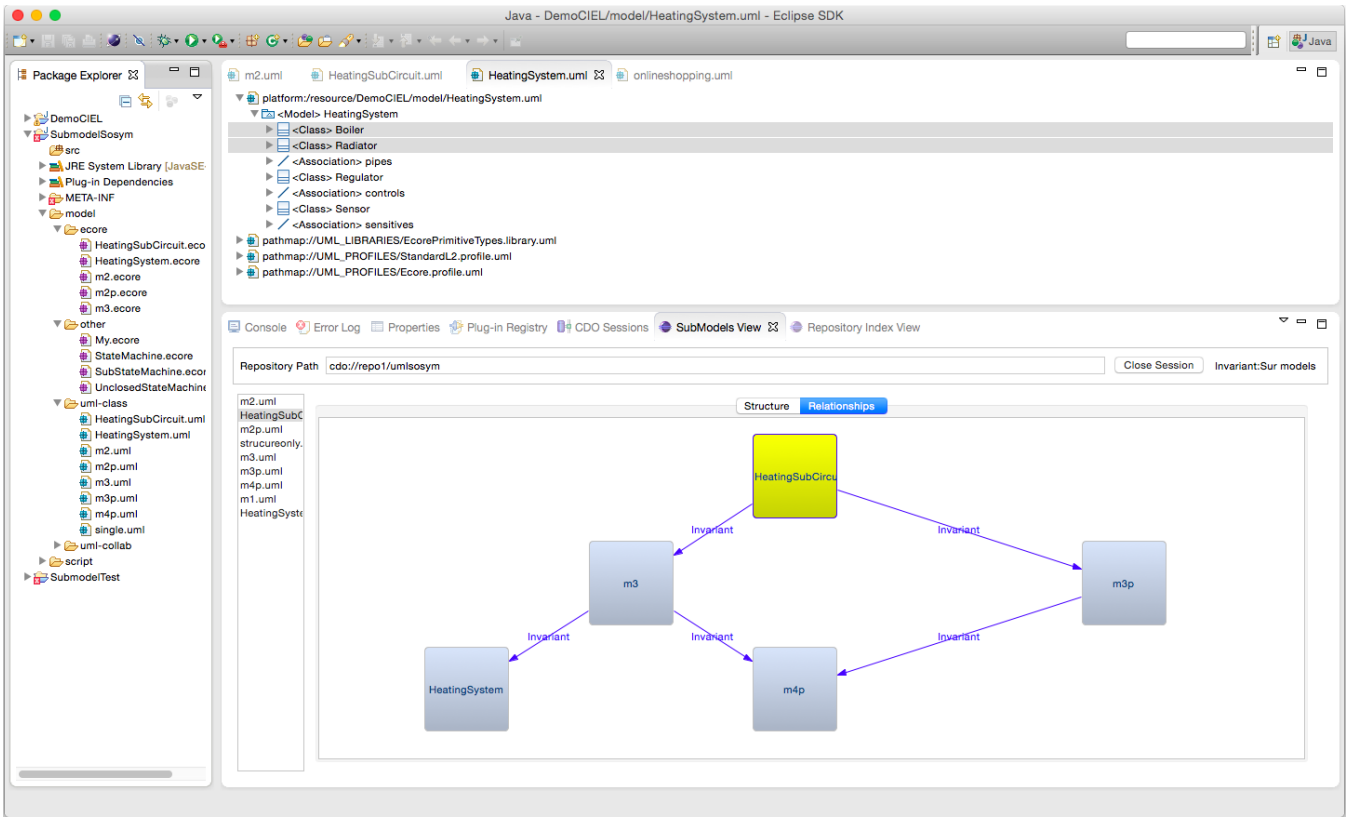

Fig. 37 Model searching in Eclipse

Search function	Formal definition	Purpose
S_{sup}	$S_{sup}(q) = \{m \mid q \text{ inv in } m\}$	Search models m that includes query model q in invariant way
S_{sub}	$S_{sub}(q) = \{m \mid m \text{ inv in } q\}$	Search models m that are invariantly included in query model q
S_{meta}	$S_{meta}(q) = \{M \mid \widehat{q} \text{ inv in } M \wedge\ q \text{ wfm \% } \widehat{q}\ \}$	Search metamodels M whom query model q is well formed
S_{wfm}	$S_{wfm}(Q) = \{m \mid \widehat{m} \text{ inv in } Q \wedge m \text{ wfm \% } \widehat{m}\}$	Search models m that are well formed w.r.t. metamodel Q
S_{sup_meta}	$S_{sup\_meta}(q, M) = \{m \mid q \text{ inv in } m \wedge\ \widehat{m} \text{ inv in } M \wedge m \text{ wfm \% } \widehat{m}\}$	Search models m containing query model q and being well formed with respect to M
S_{sup_proper}	$S_{sup\_proper}(q) = \{m \mid q \text{ inv in} m \wedge \widehat{q} \text{ inv in } \widehat{m} \wedge m \text{ wfm \% } \widehat{m}\}$	Search model m containing query model q and well formed with respect to proper metamodel of q

Table 1 Kinds of model search provided by our tool

Implementation uses an inverted-index structure to filter candidate models as they can be numerous in a large repository. This inverted-index is filled from the content of stored models, metamodels including proper ones[11]. It is composed of three hashtables following the structure of models based on our formalism. There is one hashtable for indexing elements of models and two hashtables for indexing their constraints. Figure 38 shows the two kinds of hashtable. In the hashtable dedicated to model elements, each hash key corresponds to a compact representation of a model element based on its name and metatype while the associated value is a list of location (URI) corresponding to models in the CDO repository that contains the element. From this hashtable, it is easy and fast to find stored models that include a set of elements contained in a query model by intersection. The two hashtables dedicated to constraints are both struc-

[11] Concerning proper metamodels, they are stored and indexed at the same time than their related model. An additional bi-directional mapping table is also maintained between model and their proper metamodel for accelerating checking required by several searches

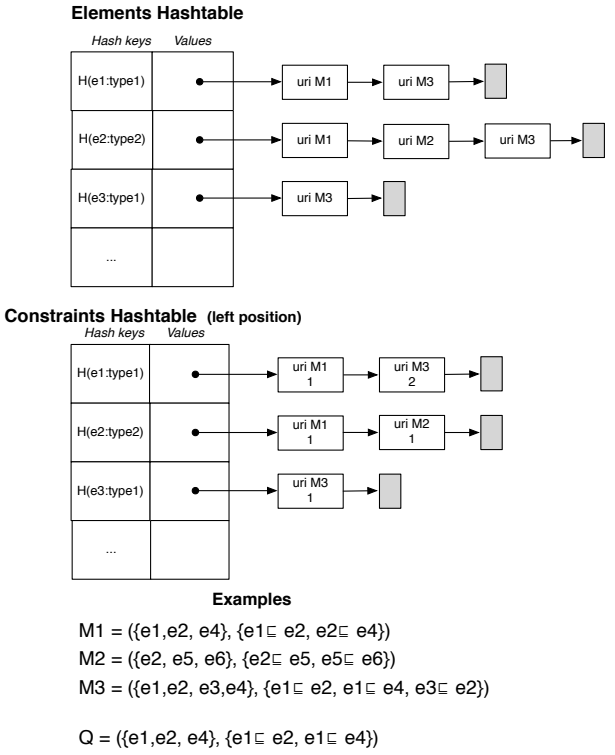

Fig. 38 Hashtables forming the inverted-index used for model searching

tured the same way. In these hashtables, keys are compact representation of model elements as explained before but structure of the associated values differs. In that case, values in the list are couple containing a location of a model in the repository plus a number. The latter is equal to the number of times the element identified by the key appears at the left position in all constraints (right position for the second hashtable) of the corresponding model. Purpose of memorizing the number of constraints for an element is to quickly detect models that can not be candidate for invariant inclusion due to a too weak number of constraints on that element. For illustrating the principle, assume a query model Q where one of its element $e1$ depends on two others elements $(e2, e4)$ and the hashtable of Figure 38. Because model M1 in the repository has element $e1$ but not all constraints on $e1$, it can not be included and so can be ignored. On the contrary model M3 satisfies the condition. In the search process, the purpose of the inverted-index is to quickly detect models in the repository that must be retained or ignored on the basis of their content before going further in the process. Then, after this filtering step, selected models are retrieved from the repository and compared with the query model regarding inclusion or well-formedness relationships depending on cases. For the comparisons, the tool exploits some results exhibited in previous section.

6.3 Quantitative evaluation

Search functions of the preceding section make intensive use of invariance and well-formedness tests with the concept of proper metamodel (see Table 1). This section performs time analysis on the execution of these tests. Properties of this paper are applied and enhance running time[12]. A quantification of time benefits is showed.

For this evaluation, we use models of our GenMyModel industrial partner [13]. GenMyModel is a web collaborative modelling tool which supports the edition of either standard models such as UML (class, sequence, activity and use case diagrams) and BPMN (Business Process Management and Notation diagrams) or proprietary models such as flowchart and database diagrams. It also supports the collaborative edition of metamodels (EMF Ecore diagrams). GenMyModel counts more than 150 000 community users distributed in 140 countries.

The evaluation process involves the following steps. First one is the selection of models (UML models, BPMN models and Ecore metamodels) which own more than 20 model elements. Second step consists in the evaluation of metamodel membership and well-formedness checking for each of these models.

A	B	C	D	E	F
Kind of model	Nb. of model m	Avg. size of $\widetilde{m}$	Metamodel M	Size of $\widetilde{M}$	Size of $\sqsubseteq_M$
Class	1791	174	UML	1622	2214
UseCase	500	116			
Sequence	397	98			
Activity	336	80			
BPMN	29	132	BPMN2	831	1074
Ecore	16	70	ECORE	164	146

Table 2 Kinds of evaluated models

Table 2 shows the different kinds of the 3069 models that were selected. Most of them are UML models. Column C gives the average number of model elements for each kind of model, columns E and F give metamodel size (number of constructs and constraints).

Table 3 shows evaluation results. For the well-formedness test, two methods are applied. The first one checks if model m is well formed w.r.t its whole metamodel M while the second one directly applies Property 8: checking if its proper metamodel $\widehat{m}$ is invariant in M and m is well formed w.r.t $\widehat{m}$.

Column B of Table 3 gives the number of models which pass the tests. All models are members of their corresponding metamodel while some models are not well formed. For each model kind, Column C (resp. Column D) gives the average time (in milliseconds) required

[12] on Intel I7, 2.10 GHz × 4 core.

[13] `http://www.genmymodel.com`

A	B	C	D	E
Model Kind	Nb. of member (wfm)	Meth. 1 Avg. time wfm(ms)	Meth. 2 Avg. time wfm(ms)	gain (%)
Class	1791 (1714)	67.15	28.70	57.26
UseCase	500 (432)	37.57	9.08	75.83
Sequence	397 (77)	8.10	3.64	54.99
Activity	336 (322)	31.69	7.57	76.11
BPMN	29 (24)	16.10	8.10	49.68
Ecore	16 (16)	4.25	3.38	20.59
All	**3069(2585)**	**50.00**	**19.62**	**60.76**

Table 3 Membership and well-formedness tests

for the well-formedness test following method 1 (resp. method 2). Column E gives the percent of time benefit of method 2 over method 1. Last row of the table summarizes this study by giving results for all models of any kind. It appears that method 2 is near 61 percent faster than method 1.

A	B	C	D	E	F	G
Model Kind	Nb of m	Avg. size of $\widetilde{\widehat{m}}$	Avg. size of $\sqsubseteq_{\widehat{m}}$	Avg. time of $\widehat{m}$	$\widehat{m}$ inv M	m wfm wrt $\widehat{m}$
Class	1791	103	137	0.16	0.68	27.86
UseCase	500	97	129	0.11	0.82	8.15
Sequence	397	108	145	0.06	1.36	2.22
Activity	336	96	122	0.09	0.58	6.91
BPMN	29	91	109	0.03	0.34	7.72
Ecore	16	63	70	0.06	0.06	3.25

Table 4 Focus on method 2

Table 4 focuses on method 2. Column E gives running time for building of the proper metamodel and Column F gives running time for checking invariance between $\widehat{m}$ and M. These two operations have low running time costs in accordance with their linear complexity[14]. Column G gives the running time for the well-formedness test of m w.r.t. $\widehat{m}$ and it clearly appears that it is the most time-consuming part. This explains why method 2 offers significative time benefit compared to method 1 (Table 3). It is due to the reduced size of the involved metamodel (proper one vs. whole metamodel)[15]. It is particularly true for large metamodels (see difference of gain betweeen the large UML metamodel and the simpler Ecore metamodel). The benefit is confirmed when considering metamodel partition into smaller submetamodels thanks to Property 9 (such as partitioning of UML in diagrams of Class, UseCase, Sequence ...).

A major conclusion of this experimental study is that the invariance test offers low running time cost in comparison to well-formedness checking, as expected. As a consequence, all stated properties which exploit invariance to establish the well-formedness are much more efficient. In particular, this confirms the importance of Property 15 as a major result: once you know that a model m (of, say, an overall project) is well formed w.r.t. a reference metamodel M, you can check that any submodel sm of m (corresponding to a sub-project) is well formed w.r.t. M with the only low-time consuming test that sm is invariant in m. As already said in the preceding sections, this is particularily prominent when modeling in the large with the need to partition (meta)models into smaller manageable ones.

7 Related works

Although submodels and submetamodels underlie many MDE practices, these notions have not been widely studied by themselves otherwise than through specific applications. This often leads to loose definitions or implicit meaning which are prone to misunderstanding and make difficult the systematic control of methods and tools which manipulate them [12, 49]. Furthermore, much of existing works on submodels and submetamodels only address one dimension of their duality, submodels compared to a reference metamodel or alternatively models compared to submetamodels, but rarely the two: submodels compared to submetamodels. Finally, submodels and submetamodels are not so often defined the same way as it is the case here, allowing to apply same submodel concepts and properties whatever the level for the benefit of their symmetrical and homogeneous treatment. See [12] for an extended synthesis of related works on the submodel dimension.

In [29], an algorithm to systematically decompose a model into a submodel lattice with reference to a metamodel is proposed. The major objective is the acceptation by tools which treat models conforming to a specific metamodel of submodels also. It is a major breakthrough in model comprehension by tools through submodels. But, conversely, they do not consider the problem of metamodel decomposition into submetamodels and subsequent decomposition of models. There is a good reason for this. Envisaged tools are those tightly wired to a reference metamodel which, somehow, defines them. It

[14] More precisely $O(e_m + c_m)$ and $O(c_{\widehat{m}} \times c_M)$, with e_x = number of model elements of x, c_x = number of constraints of x.

[15] More precisely $O(e_m^2 \times c_M)$ for method 1 compared to $O(e_m^2 \times c_{\widehat{m}})$ for method 2.

would be interesting to generalize the approach by also considering the systematic decomposition of a reference metamodel. Consequent decomposition of its member models into submodels could then be treated by tools based on a metamodel but also dealing with its submetamodels.

Fragmenta [3] is a formalization of fragmented models. Its goal is to enable the construction of model fragments that can be processed in isolation and composed to make bigger fragments. In Fragmenta, a model is a collection of fragments organized hierarchically by means of clusters which are fragment containers. Model fragments group related elements with their links. Elements of a fragment may reference elements from other fragments of the same model through proxies according to fragmentation strategies specified in the metamodel. Such strategies allow to express the decomposition or extension of a particular kind of element in a fragment using those of other fragments. Composition of model fragments is based on the union of their elements and the merge of their proxies. Fragmenta also defines the typing between models and metamodels at the level of fragments. The typing of a model relies on the typing of all elements and links included in its fragments with those of corresponding fragments in the metamodel plus the compliance to fragmentation strategies. Fragmenta formalization lies on typed graphs and category theory. Models and their included clusters and fragments are defined as three layered graphs and several morphisms between them. Composition is based on the colimit construction of category theory. Model typing builds up on typing morphisms between constituents of layered graphs representing the model and metamodel. So, Fragmenta also covers both modeling and metamodeling level but makes some distinction between models and metamodels in the formalization contrarily to the present work. One key difference between both works comes from the fact that Fragmenta is a design approach aiming to support the modularization of models by metamodel-defined fragmentation. In Fragmenta, inclusion relationships between units (fragments, clusters, models) are established at design time in conformance with their corresponding metamodel. This mainly contrasts with our grounding which aims to relate (sub)models and (sub)metamodels only on the basis of their inclusion properties, regardless of their origin.

The present work is also related to model extraction which is an operation producing a submodel of an overall input model. This operation underlies many MDE practices such as view or aspect-oriented modeling, model comprehension [35,38] model management and more generally structuring of large projects [12,43,46]. In existing works, there are many variations on the mechanism used for determining submodel elements. It ranges from simple mechanism such as the enumeration of submodel elements to more sophisticated ones such as logical expression over models [28] or dedicated modeling language [10]. Depending on the work, the nature of produced submodels also varies. It is not always mandatory that produced submodels conform to the metamodel of the input model [26]. Some works have investigated submodel production due to metamodel decomposition and pruning [4,47,50] or submetamodel selection [11] so that they directly compare with the present contribution. In these works, produced submodels are related to corresponding submetamodels through conformance but their respective inclusion properties are not considered to structure and relate them. As so, it is only a particular case of the general question of the relation between submodels and submetamodels studied here. Moreover, propositions in existing works often depend on particular technological choices, such as the UML standard, or specific modeling environments. The present formalism and its properties could contribute to generalize and better characterize the technics by overcoming technological specificities.

Model compatibility is another topic which is related to the present work. This topic is concerned with model substitutability in operations and tools and calls for the comparison of models with a hierarchy of related metamodels. In the works on Model Typing and SubTyping [23, 52], strategies for model substitutability through metamodel inclusion is stated. Substitutability is defined by a matching relationship between two metamodels which is based on the inclusion of their elements. According to this matching relationship, a metamodel is stated as a supertype of another when all its elements is included in the latter. Typical operations that use model typing are model transformations [19, 48, 57]. They use metamodels to type and filter input models and submetamodels for genericity in their processing. Model inheritance presented in [36] is also a work that addresses model compatibility for tools through inheritance between model types. Several forms of inheritance relying on inclusion of model type intension (metamodel and its constraints) and model type extension (model instances) are proposed to provide various level of model compatibility.

Regarding these works, some proximity exists with ours. We also consider metamodel inclusion for characterizing model membership and well-formedness which are forms of model compatibility. The main aspect that differs in our work is due to submodels. In our case, submodels can also be related to metamodels and submetamodels to state their membership and well-formedness. In terms of model compatibility, capacities on submodels offered in our work allow to determine model compatibility for submodels as well partial compatibility of models regarding its submodels. To our knowledge, no existing work deals precisely with similar partial compatibility except [23]. In this work, partial compatibility for models is ensured through partial subtyping between metamodels.

Another field where our work is of interest is model searching in repositories as illustrated in Section 6. In ex-

isting works, two main categories of model search exist: text-based model search and content-based model one. In the first category, models are indexed and queried as documents containing text or keywords extracted from their elements. For this kind of search, the model structure and information contained in the metamodel are not taking into account when performing indexing and searching. On the contrary, the second category of work indexes and queries models using their structural content. Existing works in this category mainly differ on the way they represent model content to index and query. Moogle [42] is a search engine that constructs indexes and performs model searching using types and attributes of model elements in addition to their names. Some works like [13, 33, 53] rely on logical expressions for that purpose. Finally, there are works [9, 24] which adopt query by example search and graph representation for indexing and searching models. In these works, queries are provided as models or model fragments. Resulting models are retrieved using graph-matching techniques between queries and models.

Present work compares to content-based searching with two main specificities. First, existing works consider neither the uniform search of models and metamodels nor the search of unspecified fragments as permitted by our formalism. Indeed, they generally assume that the searched models and query model fragments are all well formed with respect to their metamodel. A second difference is related to inclusion properties. In general, existing works only test inclusion between the querying model and the stored ones. By comparison, the present work considers a larger range of inclusion modalities. As presented in Section 6, resulting properties are exploited to support the search of submodels included in a query model, the ordering of the results and improve efficiency thanks to transitivity. To our knowledge, there are no existing works that rely on inclusion properties to provide similar capacities. Beyond that comparison, it is interesting to observe that no work considers the search of models with respect to a metamodel space although such capacities are helpful in practice.

Through the proper metamodel notion, our work is related to metamodel induction also studied elsewhere. [25] relies on metamodel induction to address the issue of model instances in repositories with unknown or lost metamodel, notably due to metamodel evolution. The authors adapt algorithms from the field of grammar inference in order to cope with the issue. Metamodel induction is also exploited in [41] for supporting the design of metamodel by experts who have the knowledge of domain concepts but not necessarily the technical skills required by MDE platforms. They propose an interactive approach where experts sketch some model fragments via graphical editors and a metamodel is induced iteratively from these fragments. At the end, the resulting metamodel is compiled for MDE platforms like EMF or MetaDepth [39]. The "promotion" operator described in [55] is also dedicated to metamodel induction. This operation consists in mapping a model into a new metamodel that in turn is used to build models that conform to it. In the present work, the purpose of metamodel induction is quite different. Its intent is the characterization of submodels at the metalevel and the comparison of induced submetamodels with other metamodels. We did not find works that provide similar form of metamodel induction. At a more general level, submetamodel induction is related to the symmetrical operation of model projection resulting in submodels due to the selection of metamodels. We are currently studying how these operations compare and combine with model extraction studied in [12] whatever the model level.

8 Conclusion

The concept of sub (meta) model is central in MDE, by offering structuring qualities to model spaces and related operations. Concerned practices are numerous and involve (but not exhaustively) : composition methodologies and model reuse, decomposition for (meta) model comprehension, practical engineering activities such as model editing, versioning and sharing and more generally model management as offered by repositories.

In the present paper we concentrate on the special form of submodels which are submetamodels with their specific role for model space structuring. Retaining the formalism of [12] offers homogeneous solutions to the two main following issues:

- Submetamodels being defined the same way submodels are, stated submodel inclusion properties proved powerful enough to structure model spaces at their own meta level.
- Assuming the preceding, it was demonstrated that submetamodel structuring at the meta level relates to submodel structuring at the model level and reciprocally due to modeling level interdependency.

After pointing out that the problem is two-way between submodels and submetamodels, it was systematically and symmetrically examined under the logical relationships of model membership and model well-formedness which relate them. This was facilitated by the isolation of the original concept of *proper metamodel* of a model which makes the bridge between submodel and submetamodel properties. The uniform formalization offers algebraic grounding to characterize model spaces for better comprehension and control of related practices.

We particularly study its application to model spaces as supported by repositories. Submetamodels bring qualities for organizing repositories through partitioning and facilitating metamodel guided operations. We are studying systematic arrangement of repositories thanks to the comparison of submodel and submetamodel hierarchies established in the paper. As far as querying is concerned,

it was shown how this work contributes to content-based model searching. Functionalities for model searching relative to submetamodels were also presented. They were eased by indexing submodels by submetamodels using proper metamodels as characteristic keys identifying models at the meta level. Finally, experiment on a big base of models showed quantified benefits of the stated properties.

Another area where submodels and submetamodels are determinant is substitutability in model operations and tools. As said in the related works, this concerns model (sub) typing which is a major issue of MDE. This calls for comparison of models with hierarchies of metamodels, viewed as hierarchies of model types. The present formalism and its results on submodel and submetamodel hierarchies may contribute to this need. We ourselves use this formalism to formalize the application of templates to model hierarchies with the idea that template formal parameters determine a "model type" which governs substitutability of candidate models [56].

More generally, the study aimed at better characterization of model spaces and related practices as identified in [22,30]. The growing needs for the management of big model spaces in repositories, moreover populated with larger and larger models, cause problems of scalability in model management [7,18,34]. All of this calls for systematic investigation of the concept of model part and its related notion of submodel to address the complexity. We hope that the present paper could help in better comprehension and control of related phenomena.

References

1. UML. Home Page. http://www.omg.org/technology/uml, 2001.
2. M. Alanen and I. Porres. Difference and Union of Models. In *Proceedings of 6th International Conference on the Unified Modeling Language, Modeling Languages and Applications (UML'03)*, volume 2863 of *LNCS*, pages 2–17. Springer, 2003.
3. N. Amálio, J. de Lara, and E. Guerra. Fragmenta: A Theory of Fragmentation for MDE. In *Proceedings of ACM/IEEE 18th International Conference on Model Driven Engineering Languages and Systems*, October 2015.
4. J. H. Bae, K. Lee, and H. Seok Chae. Modularization of the UML Metamodel Using Model Slicing. In *Proceedings of 5th International Conference on Information Technology: New Generations (ITNG'08)*, pages 1253–1254. IEEE Computer Society, 2008.
5. E. Baniassad and S. Clarke. Theme: An Approach for Aspect-Oriented Analysis and Design. In *Proceedings of 26th International Conference on Software Engineering (ICSE '04)*, pages 158–167. IEEE Computer Society, 2004.
6. M. Barbero and J. Bézivin. Structured Libraries of Models. In *Proceedings of 1st International Workshop on Towers of Models (TOWERS'07)*, 2007.
7. A. Benelallam, A. Gómez, G. Sunyé, M. Tisi, and D. Launay. Neo4EMF, a Scalable Persistence Layer for EMF Models. In *ECMFA- European conference on Modeling Foundations and applications*, volume 8569, pages 230–241. Springer, April 2014.
8. J. Bézivin, F. Jouault, P. Rosenthal, and P. Valduriez. Modeling in the Large and Modeling in the Small. In *Model Driven Architecture*, volume 3599 of *LNCS*. Springer, 2005.
9. B. Bislimovska, A. Bozzon, M. Brambilla, and P. Fraternali. Graph-based search over web application model repositories. In *Proceeding of International Conference on Web Engineering*, volume LNCS 6757. Springer, 2011.
10. A. Blouin, B. Combemale, B. Baudry, and O. Beaudoux. Modeling Model Slicers. In *Proceedings of 14th International Conference on Model Driven Engineering Languages and Systems (MoDELS'11)*, volume 6981 of *LNCS*, pages 62–76. Springer, 2011.
11. E. Burger, J. Henss, M. Küster, S. Kruse, and L. Happe. View-based model-driven software development with ModelJoin. *Software and Systems Modeling*, pages 1–24, 2014.
12. B. Carré, G. Vanwormhoudt, and O. Caron. From subsets of model elements to submodels, a characterization of submodels and their properties. *Software and Systems Modeling*, 14:861–887, May 2015.
13. J. Chimiak-Opoka, M. Felderer, C.Lenz, and C.Lange. Querying UML Models using OCL and Prolog: A Performance Study. In *In Proceeding of Software Testing Verification and Validation Workshop*, 2008.
14. T. Clark, A. Evans, and S. Kent. Aspect-oriented metamodelling. *Computer Journal*, 46(5):566–577, 2003.
15. S. Clarke. Extending standard UML with Model Composition Semantics. In *Science of Computer Programming*, volume 44, pages 71–100. Elsevier Science, 2002.
16. J. Dingel, Z. Diskin, and A. Zito. Understanding and improving UML package merge. *Software and System Modeling*, 7(4):443–467, 2008.
17. D. D'Souza and A. Wills. *Objects, Components and Frameworks With UML: The Catalysis Approach*. Addison-Wesley, 1999.
18. J. Espinazo-Pagán and J. García Molina. Querying large models efficiently. *Information & Software Technology*, 56(6):586–622, 2014.
19. A. Etien, A. Muller, T. Legrand, and R. Paige. Localized model transformations for building large-scale transformations. *Software and Systems Modeling*, pages 1–25, 2013.
20. R. B. France, J. M. Bieman, and B. H. C. Cheng. Repository for Model Driven Development (ReMoDD). In *Proceeding of MoDELS'06 Workshops*, volume 4364 of *LNCS*, pages 311–317. Springer, 2006.
21. R.B. France, K. Dae-Kyoo, S. Ghosh, and E. Song. A UML-based pattern specification technique. *IEEE Transactions on Software Engineering*, 30(3):193–206, March 2004.
22. R.B. France and B. Rumpe. Model-Driven Development of Complex Software: A Research Roadmap. In *Proceedings of Future of Software Engineering at ICSE*, 2007.
23. C. Guy, B. Combemale, S. Derrien, J. Steel, and J.M. Jézéquel. On Model Subtyping. In *Proceedings of 8th*

European Conference on Modelling Foundations and Applications (ECMFA 2012), volume 7349 of *LNCS*, pages 400–415. Springer, 2012.
24. B. Izsó, G. Szárnyas, I. Ráth, and D. Varró. Incquery-d: Incremental graph search in the cloud. In *Proceedings of the Workshop on Scalability in Model Driven Engineering*, 2013.
25. F. Javed, M. Mernik, J. Gray, and B. R. Bryant. Mars: A metamodel recovery system using grammar inference. *Information and Software Technology*, 50(910):948 – 968, 2008.
26. C. Jeanneret, M. Glinz, and B. Baudry. Estimating footprints of model operations. In *Proceedings of 33rd International Conference on Software Engineering (ICSE'11)*, pages 601–610. ACM, 2011.
27. J.M. Jézéquel. Model driven design and aspect weaving. *Software and System Modeling*, 7(2):209–218, 2008.
28. H. H. Kagdi, J. I. Maletic, and A. Sutton. Context-Free Slicing of UML Class Models. In *Proceedings of 21st IEEE International Conference on Software Maintenance (ICSM'05)*, pages 635–638. IEEE Computer Society, 2005.
29. P. Kelsen, Q. Ma, and C. Glodt. Models within Models: Taming Model Complexity Using the Sub-model Lattice. In *Proceedings of 14th International Conference on Fundamental Approaches to Software Engineering, FASE'11*, volume 6603 of *LNCS*, pages 171–185. Springer, 2011.
30. S. Kent. Model Driven Engineering. In *Proceedings of the 3rd International Conference on Integrated Formal Methods (IFM'02)*, volume 2335 of *LNCS*, pages 286–298. Springer-Verlag, May 2002.
31. J. Kienzle, W. Al Abed, F. Fleurey, J-M. Jézéquel, and J. Klein. Aspect-Oriented Design with Reusable Aspect Models. In *Transactions on Aspect-Oriented Software Development VII - A Common Case Study for Aspect-Oriented Modeling*, volume 6210 of *LNCS*, pages 272–320. Springer, 2010.
32. J. Klein, J. Kienzle, B. Morin, and J.M. Jézéquel. Aspect Model Unweaving. In *Proceedings of 12th International Conference on Model Driven Engineering Languages and Systems, MODELS'09*, volume 5795 of *LNCS*, pages 514–530. Springer, 2009.
33. D. Kolovos, R. Wei, and K. Barmpis. An approach for efficient querying of large relational datasets with OCL based languages. In *Proceedings of the Workshop on Extreme Modeling at MODELS'13*, 2013.
34. D.S Kolovos, L.M. Rose, N. Matragkas, R.F. Paige, E. Guerra, J.S. Cuadrado, J. De Lara, I. Rath, D. Varro, M. Tisi, and J. Cabot. A research roadmap towards achieving scalability in model driven engineering. In *Proceedings of the Workshop on Scalability in Model Driven Engineering*, pages 2:1–2:10, Budapest, Hungary, June 2013. ACM.
35. B. Korel, I. Singh, L. H. Tahat, and B. Vaysburg. Slicing of State-Based Models. In *Proceedings of 19th International Conference on Software Maintenance (ICSM'03)*, pages 34–43. IEEE Computer Society, 2003.
36. T. Kuhne. An Observer-Based Notion of Model Inheritance. In *Proceedings of 13th International Conference on Model Driven Engineering Languages and Systems (MoDELS'10)*, volume 6394 of *LNCS*, pages 31–45. Springer, 2010.
37. Ph. Lahire, B. Morin, G. Vanwormhoudt, A. Gaignard, O. Barais, and J-M Jézéquel. Introducing Variability into Aspect-Oriented Modeling Approaches. In *Proceedings of 10th International Conference on Model Driven Engineering Languages and Systems (MoDELS'07)*, volume 4735 of *LNCS*, pages 498–513. Springer, 2007.
38. K. Lano and S. Kolahdouz-Rahimi. Slicing Techniques for UML Models. *Journal of Object Technology*, 10:11:1–49, 2011.
39. J. Lara, E. Guerra, R. Cobos, and J. Moreno-Llorena. Extending deep meta-modelling for practical model-driven engineering. *Computer Journal*, 57(1):36–58, 2014.
40. T. Levendovszky, L. Lengyel, and T. Mészáros. Supporting domain-specific model patterns with metamodeling. *Software and Systems Modeling*, 8:501–520, 2009.
41. J. Lopez-Fernandez, J. Cuadrado, E. Guerra, and J. de Lara. Example-driven meta-model development. *Software & Systems Modeling*, pages 1–25, 2013.
42. D. Lucrédio, R. Pontin de Mattos Fortes, and J. Whittle. Moogle: a metamodel-based model search engine. *Software and System Modeling*, 11(2):183–208, 2012.
43. S. Melnik, E. Rahm, and Ph. A. Bernstein. Rondo: a programming platform for generic model management. In *Proceedings of the 2003 ACM SIGMOD international conference on Management of data*, SIGMOD '03, pages 193–204. ACM, 2003.
44. A. Muller, O. Caron, B. Carré, and G. Vanwormhoudt. On Some Properties of Parameterized Model Application. In *Proceedings of 1st European Conference on Model Driven Architecture - Foundations and Applications (ECMDA-FA'05)*, volume 3748 of *LNCS*, pages 130–144. Springer, November 2005.
45. Y. R. Reddy, S. Ghosh, R. B. France, G. Straw, J. M. Bieman, N. McEachen, E. Song, and G. Georg. Directives for Composing Aspect-Oriented Design Class Models. *Transactions on Aspect-Oriented Software Development I*, 3380:75–105, 2006.
46. T. Reiter, E. Kapsammer, W. Retschitzegger, and W. Schwinger. Model Integration Through Mega Operations. In *Proceedings of the International Workshop on Model-driven Web Engineering (MDWE'05)*, 2005.
47. S. Sen, N. Moha, B. Baudry, and J-M. Jézéquel. Metamodel Pruning. In *Proceedings of 12th International Conference on Model Driven Engineering Languages and Systems (MoDELS'09)*, volume 5795 of *LNCS*, pages 32–46. Springer, 2009.
48. S. Sen, N. Moha, V. Mahé, O. Barais, B. Baudry, and J-M. Jézéquel. Reusable model transformations. *Software and Systems Modeling*, 2010.
49. M. Siikarla, J. Peltonen, and J. Koskinen. Towards unambiguous model fragments. *Nordic Journal of Computing*, 13:180–195, 2006.
50. A. Solberg, R. France, and R. Reddy. Navigating the MetaMuddle. In *Proceedings of 4th Workshop in Software Model Engineering*, 2005.
51. P. Sriplakich, X. Blanc, and M.P. Gervais. Collaborative software engineering on large-scale models: requirements and experience in modelbus. In *Proceedings of the 2008 ACM symposium on Applied computing*, pages 674–681. ACM, 2008.
52. J. Steel and J-M. Jézéquel. On model typing. *Software and System Modeling*, 6(4):401–413, 2007.

53. H. Störrle. VMQL: A visual language for ad-hoc model querying. *Journal of Visual Languages and Computing*, 22(1):3–29, 2011.
54. G. Taentzer, C. Ermel, P. Langer, and M. Wimmer. A fundamental approach to model versioning based on graph modifications: from theory to implementation. *Software and Systems Modeling*, pages 1–34, 2012.
55. X. Thirioux, B. Combemale, X. Crégut, and P.L Garoche. A Framework to Formalise the MDE Foundations. In *International Workshop on Towers of Models (TOWERS 2007)*, pages 14–30, Zurich, Switzerland, June 2007. University of York.
56. G. Vanwormhoudt, O. Caron, and B. Carré. Aspectual templates in UML. *Software and Systems Modeling*, pages 1–29, 2015.
57. A. Vignaga, F. Jouault, M. Bastarrica, and H. Brunelire. Typing artifacts in megamodeling. *Software and Systems Modeling*, 12(1):105–119, 2013.
58. J. Whittle, P. K. Jayaraman, A. M. Elkhodary, A. Moreira, and J. Araújo. MATA: A Unified Approach for Composing UML Aspect Models Based on Graph Transformation. *Transactions on Aspect-Oriented Software Development VI*, 6:191–237, 2009.