Orthogonal Meta-Modeling

Katharina Görlach and Frank Leymann

Institute of Architecture of Application Systems
University of Stuttgart
Germany

{goerlach, leymann}@iaas.uni-stuttgart.de

BibTeX:

```
@article{GoeLey2014,
  author    = {Katharina G\"{o}rlach and Frank Leymann},
  title     = {Orthogonal Meta-Modeling},
  journal   = {Journal of Systems Integration},
  year      = {2014},
  volume    = {5},
  number    = {2},
  pages     = {3--17},
  issn      = {1804-2724},
  publisher = {Czech Society for Systems Integration (CSSI)}
}
```

ISSN: 1804-2724

Orthogonal Meta-Modeling

Katharina Görlach, Frank Leymann

Institute of Architecture of Application Systems University of Stuttgart, Germany

goerlach@iaas.uni-stuttgart.de, leymann@iaas.uni-stuttgart.de

Abstract: *This article introduces meta-modeling hierarchies additional to the conventional meta-modeling hierarchy in a model-driven architecture. Additional hierarchies are introduced orthogonal to the conventional meta-modeling hierarchy for an appropriate correlation of information on combined hierarchies. In particular, orthogonal meta-modeling enables the grouping of models on the same conventional meta-modeling layer based on additional semantic dependencies. For the enhancement of conventional meta-modeling this paper discusses the creation of orthogonal meta-modeling hierarchies, the specification of semantic dependencies in meta-modeling hierarchies, semantic instances as well as the inheritance of semantic dependencies in meta-modeling hierarchies in general. Furthermore, the paper outlines the impact of orthogonal semantic meta-modeling on automated model transformation.*

Key words: Model-Driven Architecture, Meta-Modeling, Model Transformation, Views, Semantics

1. Introduction

Model-driven software development allows developers to focus on the functionality of a software system without taking care about implementation details in the design phase. Furthermore, model-driven software development enables code generation, i.e. models can be automatically transformed to implementation artefacts. The introduction of meta-modeling allows to structure multiple models on different layers and specify particular instance-of dependencies between models (Mukerji, J. & Miller, J. 2001). Functionality-oriented models of software systems can be enhanced by adding semantic information that is typically provided by ontologies. The combination of functionality-oriented models and ontologies enables a semantic-driven reasoning on models and improves the ability for automation while developing software systems.

The approach at hand aims for an explicit specification of some semantic information, i.e. dependencies on meta-modeling layers in contrast to specify semantic information exclusively in ontologies. That means, the approach at hand emphasizes dependencies between functionality-oriented models in contrast to the enhancement of models by ontologies. In detail, semantic dependencies between models on meta-modeling layers are introduced next to the conventional instance-of dependency in meta-modeling layers. Enriching meta-modeling layers by semantic information allows to improve the ability for automation, e.g. model transformation and code generation can be improved by additionally considering semantic information. Furthermore, capabilities for grouping models in layer-based architectures are improved and the expressiveness for modeling layer-based architectures is enhanced by adding semantic dependencies to the meta-modeling concept.

The approach at hand introduces semantic meta-modeling hierarchies orthogonal to the conventional meta-modeling hierarchy and to each other. The orthogonal relation of the hierarchies ensures an appropriate correlation of contained information. In particular, models on the same conventional modeling layer can be grouped corresponding to particular semantic information. A semantic meta-modeling hierarchy is determined by a fixed instance-of dependency covering especially semantic information of related models. That means, semantic meta-modeling hierarchies are introduced similar to the conventional meta-modeling hierarchy that is determined by the conventional instance-of dependency. The conventional instance-of dependency represents the inheritance of syntax information on horizontal meta-modeling layers, i.e. allows the creation of syntactic instances on an underlying layer. In contrast, the approach at hand allows further instance-of dependencies representing the inheritance of semantic aspects, i.e. create semantic instances.

The following section 2 introduces a semantic instance-of dependency enabling a semantic meta-modeling orthogonal to the conventional meta-modeling. Afterwards, section 3 discusses orthogonal meta-modeling in general by studying different kinds of semantic instance-of dependencies for meta-modeling hierarchies, further semantic dependencies in the hierarchy as well as the inheritance of semantic dependencies. Section 4 outlines the ability to improve automated model transformation based on orthogonal meta-modeling. Finally, Section 5 presents a summary and related work.

2. Semantic Meta-Modeling

The introduction of semantic layers to the conventional meta-modeling hierarchy allows to specify semantic dependencies between models additionally to the conventional instance-of dependency. The approach at hand introduces semantic layers orthogonal to conventional meta-modeling layers. That means, models that are placed on the same conventional meta-modeling layer can be grouped based on their semantic dependencies. Figure 1 shows the enhancement of conventional meta-modeling by semantic dependencies in the field of service composition. In particular, Figure 1 shows two conventional, i.e. horizontal meta-modeling layers L_0 and L_1 that separate the object layer and the model layer. Introduced vertical semantic layers S_0, S_1, S_2, and S_3 on the horizontal layer L_1 allow to further separation of models on layer L_1. For example, the model on layer S_0 represents a *semantic meta-model* for the models on S_1.

A semantic meta-model represents a knowledge base that can be specialized in *semantic instances*. In general, a semantic instance holds the same or less information as the related meta-model. For example, the semantic meta-model G on S_0 in Figure 1 represents the knowledge base for the models on S_1-S_3. A model on S_1 inherits semantic information from the meta-model G but specifies the inherited information by a specific executable programming language, e.g. BPEL (OASIS 2007). That means, the vertical meta-modeling layer S_1 groups models representing views on the knowledge base on S_0 each using its own modeling paradigm. In particular, the models of S_1 are different implementations of the same phenomenon specified by the semantic meta-model on S_0. The semantic instance on S_1 can be semantic meta-models for models on layer S_2. For example, a model on S_2 can create a view on its semantic meta-model by excluding information in order to focus on a specific aspect. In the following the individual vertical meta-modeling layers are discussed in detail.

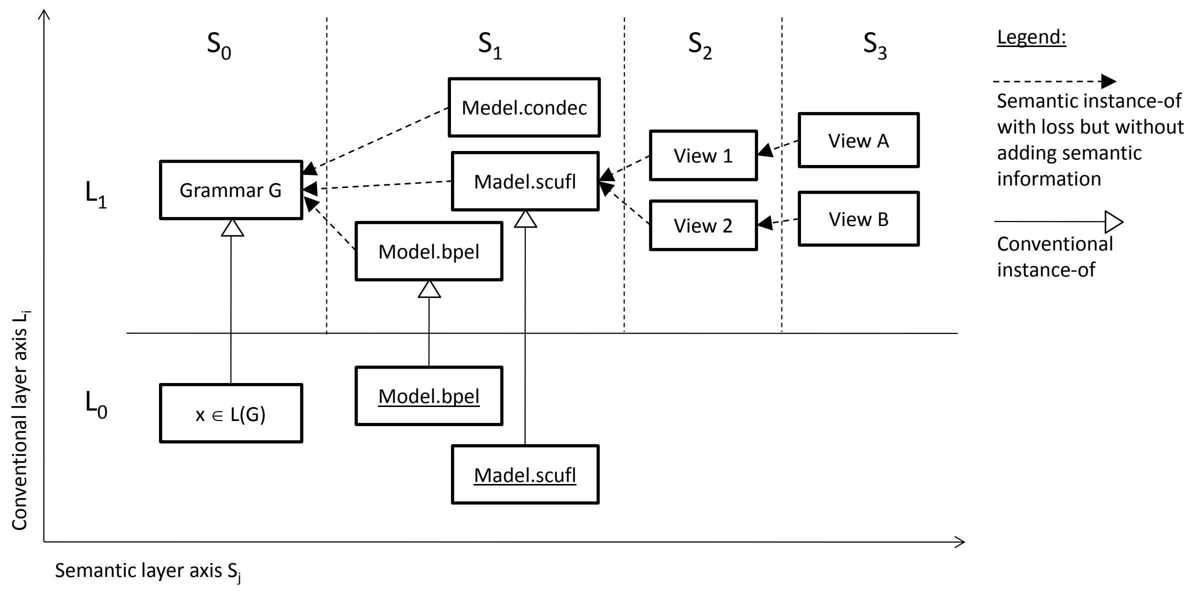


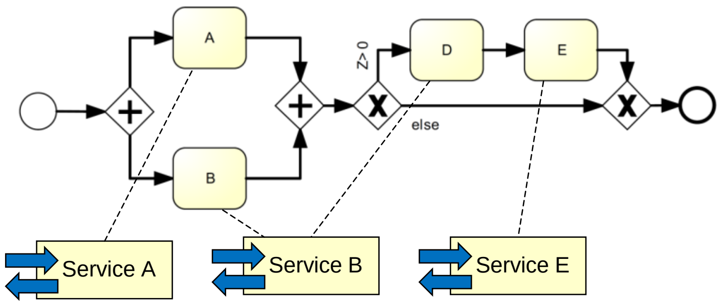
Figure 1: Orthogonal semantic meta-modeling

S_0: Semantic Knowledge Base

The starting layer L_0 of a vertical meta-modeling hierarchy collects models that represent most comprehensive semantic knowledge bases for models on other semantic layers. That means, a model on layer L_0 specifies the most general information about a phenomenon that can be semantically specialized on other vertical layers. In general, all modeling languages can be used to specify a semantic knowledge base. However, this section briefly introduces a meta-model that is especially designed for the unification of specification languages for service compositions and therefore suitable to model semantic knowledge bases concerning service compositions. In particular, the unified model is intended to be the engine-internal representation of a service composition that is used for the execution of the particular service composition. Therefore, an engine is not required to support multiple modeling languages and the support of multiple modeling languages in an execution environment does not require the use of multiple engines, respectively. The unified model is based on formal grammars, i.e. a service composition is intended to be executed based on a grammar-based specification (Görlach, K. et al. 2013). The same service composition modeled by using a highly developed specification language like BPEL (OASIS 2007), BPMN (OMG 2011), Scufl (Oinn, T. et al. 2006), and ConDec (Pesic, M. 2008) is considered to be a view on the particular service composition grammar (cf. Figure 1). Therefore, a service (composition) grammar is considered to hold the most general information about the logic of a single service composition. A model of the same service composition based on high specification language represent a semantic instance of the service grammar as the semantic information of the service grammar is specialized corresponding to the particular specification language. If the expressiveness of the particular specification language is not sufficient some logic, i.e. semantic information is lost while specialization. For example, alternative execution paths cannot be specified in models exclusively using data flow-based modeling constructs. For enabling alternative paths in data flow-based models the meta-model needs to be extended by particular control flow-based modeling constructs (cf. Oinn, T. et al. (2006)). In general, most languages for service compositions provide mechanisms for extensibility that can be used to avoid the loss of information while specialization (cf. Kopp, O. et al. (2011)). However, grammar-based models of service compositions specify a general dependency between service invocations covering the execution order without specializing the semantics of this dependency. The general dependency in grammar-based models can be implemented, i.e. specialized for example by data flow and/or control flow in semantic instances.

Figure 2 illustrates a grammar-based model of an imperative, i.e. control-flow.based service composition by means of a BPMN-based service composition. In general, service calls and other relevant information are represented by non-terminals and/or terminals in the grammar-based model. For example, the activation of a service call is represented by a non-terminal B whereas the finishing of a service call is represented by a terminal b. A production rule $B \rightarrow b$ represents the relationship between the activation and the finishing of the service call. If the production rule is applied at runtime the particular service call is executed between reading the left-hand-side of the production rule and writing the right-hand-side of the production rule. For enabling service orientation in formal grammars the non-terminals are extended by a type. The introduced non-terminal type is correlated with a web service operation. That means, two non-terminals B and D of the same type represent two different calls of the same web service operation. Conventional non-terminals without type represent helper symbols ensuring the right order of activities. For example, the helper non-terminal H in Figure 2 is used for the synchronization of parallel execution paths.

Figure 3 illustrates a grammar-based model of a declarative service composition by means of a ConDec-based service composition. The constraints in the declarative service composition specify dependencies between service calls, i.e. activities. Typically, multiple activities are allowed to be executed at a specific point in time and a (human) user is expected to select a specific activity for execution. After the execution of the selected activity a new set of activities that are allowed to be executed next needs to be calculated based on all given constraints. The service composition in Figure 3 specifies three service calls A, B, and C as well as two constraints covering the service calls A and B. The constraint response(A,B) specifies that the call B must be executed in future when A is executed at least once. Additionally, the constraint precedence(A,B) specifies that the call A needs to be executed when B begins to execute. In between all other service calls C are allowed to be executed. At the very beginning of the service composition in Figure 3 the service call B is not allowed to be executed as the precedence constraint does not permit the execution of B when A was not executed before. Therefore, the start symbol S_1 in the grammar-based model allows only the activation of the service calls A and C. Additionally, the symbol ε is allowed to be activated indicating the finishing of the service composition. The index for the start symbol needs to be introduced for covering different sets of service calls that are allowed to be executed at the same point in time. The index for service calls A, B, and C is introduced in order to cover different effects of service call executions at different points in time. For example, the execution of the service call A at the very beginning of the service composition requires the execution of a following service call B. In order to ensure the following service call B the service composition switches to another index (cf. rule (4)) indicating the need for the execution of B by disallowing the finishing of the service composition. In the following, further calls A are allowed but have no impact on the index. In contrast, a following execution of the service call B satisfies the response constraint in the service composition and allows to switch to another index (cf. rule (10)). For more information about the grammar-based meta-model for service compositions please see (Görlach, K. et al. 2013) and (Görlach. K. 2013).



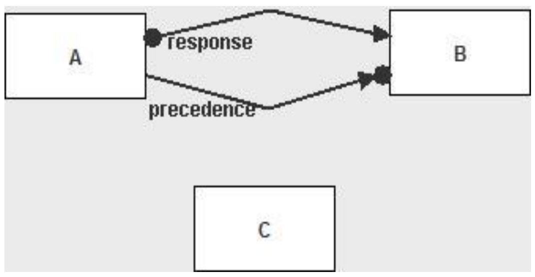
(1) S → ABH
(2) A → a
(3) B → b
(4) abH → abC
(5) C → T
(6) C → F
(7) T → D
(8) D → dE
(9) E → e
(10) F → ε

```
<nonTerminal>  <name> H </name>  </nonTerminal>


<nonTerminal>  <name> A </name>
  <type> Service A Operation A1 </type>
  <input>   <reference>Y </reference>  </input>
  <output>   <reference>Z </reference>  </output>
</nonTerminal>
<nonTerminalType   name="Service A Operation A1">
  <wsa:EndpointReference>  <wsa:Address>
        http://localhost:9763/services/ServiceA
  <wsa:Address>  </wsa:EndpointReference>
…<operation> A1 </operation>
</nonTerminalType>


<nonTerminal>  <name> C </name>
  <type> XPathSolver </type>
  <input>  <reference>Z </reference>
              <value> Z>0 </value>  </input>
   <relations>
     <relation>  <outputValue> True </outputValue>
                  <nonTerminalREF> T </nonTerminalREF>
     </relation>
     <relation>   <outputValue> False </outputValue>
                   <nonTerminalREF> F </nonTerminalREF>
     </relation>
   </relations>
</nonTerminal>
```

Figure 2: An imperative, i.e. control-flow-based service composition and its grammar-based representation



(1-3)	$S_1 \rightarrow A_1 \mid C_1 \mid \varepsilon$
(4)	$A_1 \rightarrow a\ S_2$
(5)	$C_1 \rightarrow c\ S_1$
(6-8)	$S_2 \rightarrow A_2 \mid B_2 \mid C_2$
(9)	$A_2 \rightarrow a\ S_2$
(18)	$C_3 \rightarrow c\ S_3$
(10)	$B_2 \rightarrow b\ S_3$
(11)	$C_2 \rightarrow c\ S_2$
(12-15)	$S_3 \rightarrow A_3 \mid B_3 \mid C_3 \mid \varepsilon$
(16)	$A_3 \rightarrow a\ S_2$
(17)	$B_3 \rightarrow b\ S_3$
(18)	$C_3 \rightarrow c\ S_3$

Figure 3: A declarative, i.e. constraint-based service composition and its grammar-based representation

S_1: Models with Different Underlying Paradigms

The intention of the semantic layer S_1 in Figure 1 is to group models with different underlying modeling paradigms. Assuming a grammar-based knowledge base on layer S_0 covering a single service composition the layer S_1 collects implementations of the particular service composition based on highly developed specification languages for service compositions with different underlying paradigms. For example, BPEL can be used to specify a control-flow-based model whereas ConDec can be used to specify a constraint-based model of the same service composition. Both models inherit semantic information from the same model as they specify the same service composition.

The semantic instance-of dependency separates the vertical layers S_0 and S_1. In general, a model on layer S_1 is called semantic instance of a semantic meta-model on layer S_0. A semantic instance inherits information from its semantic meta-model. However, the semantic instance specializes the inherited information by possibly excluding parts of information. That means, the semantic meta-model holds general semantic information whereas the semantic instance is specialized by restricting the general information. The inheritance of semantic information from meta-model to the instance is different to the conventional inheritance where all information of the super model is inherited and further information is added for specialization. In contrast, the inheritance of semantic information may cover a sub-set of information, i.e. the semantic instance inherits the same or less semantic information from the semantic meta-model but is not allowed to add new semantic information in general. However, a semantic instance is allowed to hold additional syntactic information as the syntactic aspect is not covered by the semantic inheritance. The character of the semantic inheritance leads to the fact that different models on layer S_1 can have different sets of possible runs although they are related to the same semantic meta-model on S_0. Furthermore, the set of possible runs of the meta-model may vary from the particular sets of semantic instances. In summary, the semantic dependencies on L_1 have impact on the dependencies on L_0 (cf. Section 3).

S_2: Views in General

Next to the semantic inheritance from layer S_0 to layer S_1 further layers for the specialization of semantic information can exist. The layer S_2 in Figure 1 groups views on a single model on layer S_1, i.e. the layer S_2 collects semantic instances of service composition implementations. A view allows to focus on specific aspects of a service composition implementation while excluding irrelevant information. Views on service compositions are very popular. Basically, views are defined by operations whose application generates a view. The particular objective of a view determines required operations and their application order. For example, Liu and Shen (2003) define views based on abstraction operations that exclusively include the loss of semantic information. However, other approaches exist where semantic information is lost as well as added (cf. Schumm, D. et al. (2010)).

Similar to different operations for creating views different semantic instance-of dependencies are thinkable representing the inheritance of different information. Using different semantic instance-of dependencies in the same vertical, i.e. semantic meta-modeling hierarchy enables an extensive enrichment of horizontal, i.e. conventional meta-modeling layers by semantic information. However, *the approach at hand proposes to use a single semantic instance-of dependency in one semantic meta-modeling hierarchy*. If multiple semantic inheritance dependencies need to be considered multiple semantic hierarchies should be introduced orthogonal to the conventional meta-modeling hierarchy and to each other. By adding an orthogonal meta-modeling hierarchy for each semantic instance-of dependency the capabilities to group models are improved significantly.

S_3: ... Final Layers

Similar to conventional meta-modeling *an infinite number of layers for semantic meta-models and semantic instances can be defined*. However, there always exists a starting point. In semantic meta-modeling with a semantic instance-of dependency covering abstraction the most general layer creates the starting layer S_0. Although an infinite number of layers can exists in the corresponding hierarchy a final layer can be determined as abstraction eventually finishes with the “empty” model.

In the field of service compositions scenarios of a service composition model are suitable to determine a final layer. Figure 4 illustrates the nature of interpreting scenarios as semantic views on a process model: The vertical semantic layer S_2 is “fold” to the horizontal conventional layer L_0. In particular, both layers represent the same kind of information and both layers represent a starting/final layer of their particular hierarchy.

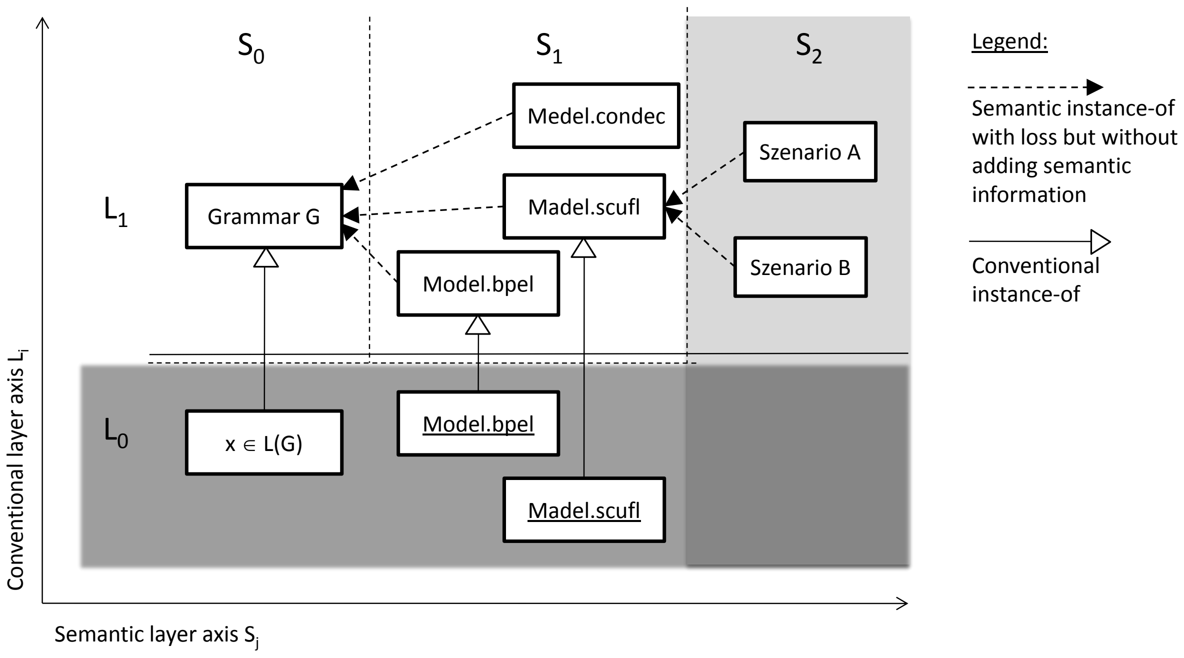
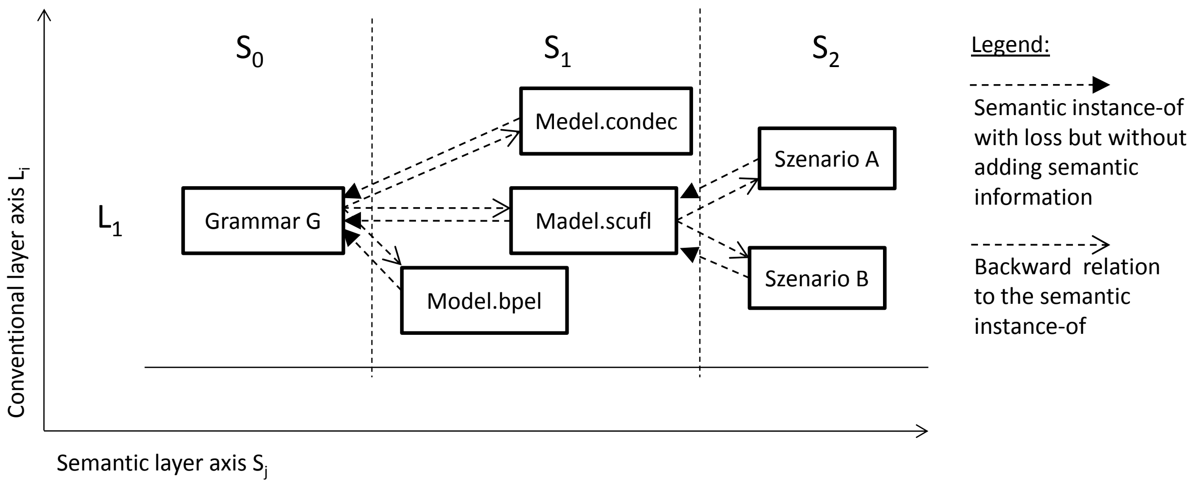


Figure 4: Scenarios creating views on service compositions

3. Semantic Dependencies on Vertical and Horizontal Layers

The previous section introduced the semantic instance-of dependency with loss but without adding semantic information. The backward relation of semantic instance-of dependencies collects semantic information of semantic instances in a single semantic meta-model on a lower semantic layer. For instance, Figure 5 shows a semantic instance-of dependency between the semantic meta-model Madel.scufl on S_1 and the semantic instances scenario A and scenario B on S_2. The backward relation reflects the combination of both scenarios A and B in the model Madel.scufl. That means, the backward relation can be used for the generation of a semantic meta-model if the instance-of dependency is assumed to be complete. The model generation based on the backward relation of a complete instance-of dependency is for example used in Petri net synthesis where a Petri net model is generated based on a given set of all possible transition sequences (cf. Badouel, E., Darondeau, P (2011) or Reisig, W. (2010)).

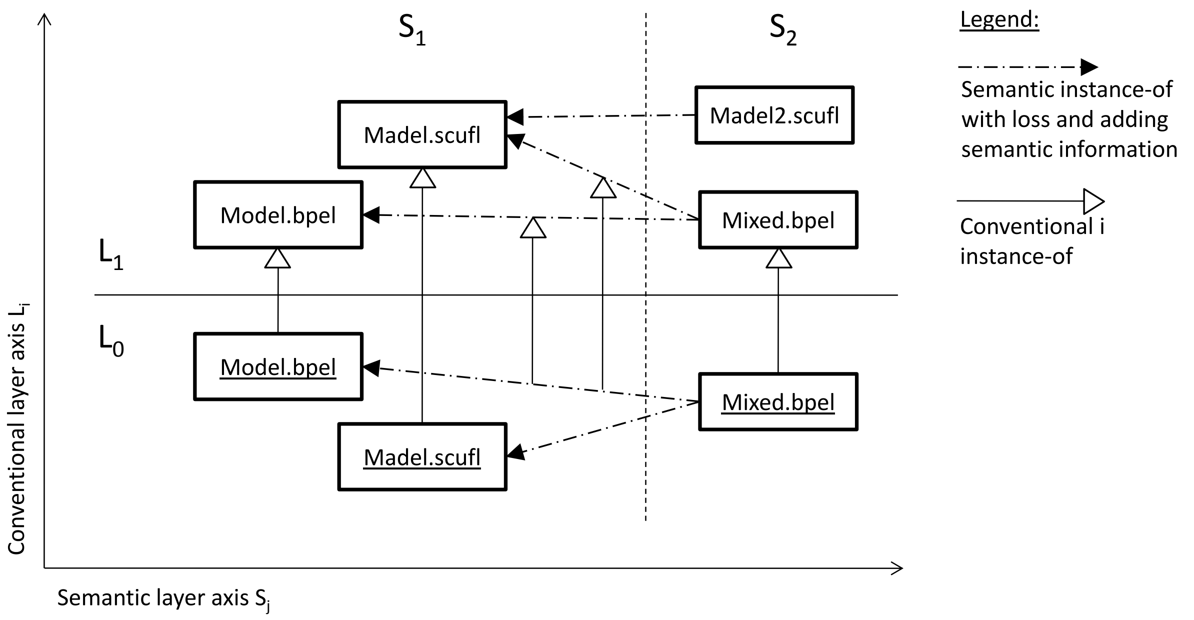
Next to the semantic instance-of dependency with loss but without adding information other dependencies can be used to create orthogonal meta-modeling layers. For example, a semantic instance-of dependency with loss and adding semantic information can be used similar to view operations allowing to add information while abstraction (cf. Schumm, D. et al. (2010)). Figure 6 shows an orthogonal meta-modeling hierarchy created by a semantic instance-of dependency with loss and adding information. Obviously, the different instance-of dependency leads to another grouping. In particular, the vertical layer S_2 in Figure 6 groups models that lose and add semantic information semantic information of models on layer S_1. For example, the semantic instance Madel2.scufl on S_2 may inherit some information from its semantic meta-model and add some logic, e.g. for fault handling. As *different instance-of dependencies lead to different groupings* and therefore to different vertical hierarchies the approach at hand proposes to introduce different vertical meta-modeling hierarchies if different instance-of dependencies are considered.



S_0
S_1
S_2
Legend:
Semantic instance-of with loss but without adding semantic information
Medel.condec
Szenario A
L_1
Grammar G
Madel.scufl
Szenario B
Model.bpel
Backward relation to the semantic instance-of
Conventional layer axis L_i
Semantic layer axis S_j

Figure 5: Semantic instance-of dependeny with loss but without adding semantic information and its backward relation

Furthermore, multiple inheritance allows to mix semantic information. For example, the model Mixed.bpel on vertical layer S_2 in Figure 6 represents a mix of the logic of both semantic meta-models corresponding to the particular instance-of dependency. Considering multiple inheritance the instance-of dependency with loss and adding information offers maximum flexibility, i.e. the semantic information of all meta-models can be combined in the semantic instance. That means, the semantic instances is allowed to specify information that is not included in one or more semantic meta-models. In contrast, the instance-of dependency with loss but without adding information restricts the mixed model to include only semantic information that is specified by all meta-models.



S_1
S_2
Legend:
Semantic instance-of with loss and adding semantic information
Madel.scufl
Madel2.scufl
Model.bpel
Mixed.bpel
Conventional i instance-of
L_1
L_0
Model.bpel
Mixed.bpel
Madel.scufl
Conventional layer axis L_i
Semantic layer axis S_j

Figure 6: Semantic instance-of dependency with loss and adding semantic information

Semantic instance-of dependencies of vertical layers can be conventionally inherited to lower horizontal layers. For example, the semantic inheritance between Model.bpel and Mixed.bpel on L_1 in Figure 6 are conventionally inherited by particular objects on L_0. That means, semantic dependencies should be specified on the highest possible horizontal layer whereas dependencies on lower layers can be derived. However, the semantics of a semantic dependency determines the limitations to the layer specifying the particular semantic dependency that can be conventionally inherited on lower horizontal layers. Figure 7 illustrates limitations to the inheritance of semantic dependencies between different horizontal layers. In particular, the layer L_2 cannot specify the same semantic instance-of dependency as specified on the layers L_1. Instead, another semantic instance-of dependency is needed on layer L_2 for instance representing specialization considering expressiveness. However, the nature of the additional semantic instance-of dependency on L_2 would be different to the nature of the semantic instance-of dependencies on layer L_1 and L_0. Therefore, the conventional instance-of dependency could not be used for inheriting possible semantic inheritance dependencies from layer L_2 to L_1 analogous to the conventional inheritance between layer L_1 and L_0. Furthermore, the additional instance-of dependency would requires to introduce an additional hierarchy orthogonal to all other hierarchies (cf. Figure 11).

Instead of a semantic instance-of dependency a meta-model mapping is specified on layer L_2 in Figure 7. The meta-model mapping represents a transformation relation that can be used for transformations of corresponding syntactic instances. That means, the transformation relation is a semantic dependency next to the semantic instance-of dependency that can be conventionally inherited to underlying horizontal layers. Figure 7 exemplarily illustrates the conventional inheritance of the transformation relation between grammars and BPEL from L_2 to layer L_1. However, the transformation relation can be further conventionally inherited from corresponding models on layer L_1 to particular models on layer L_0.

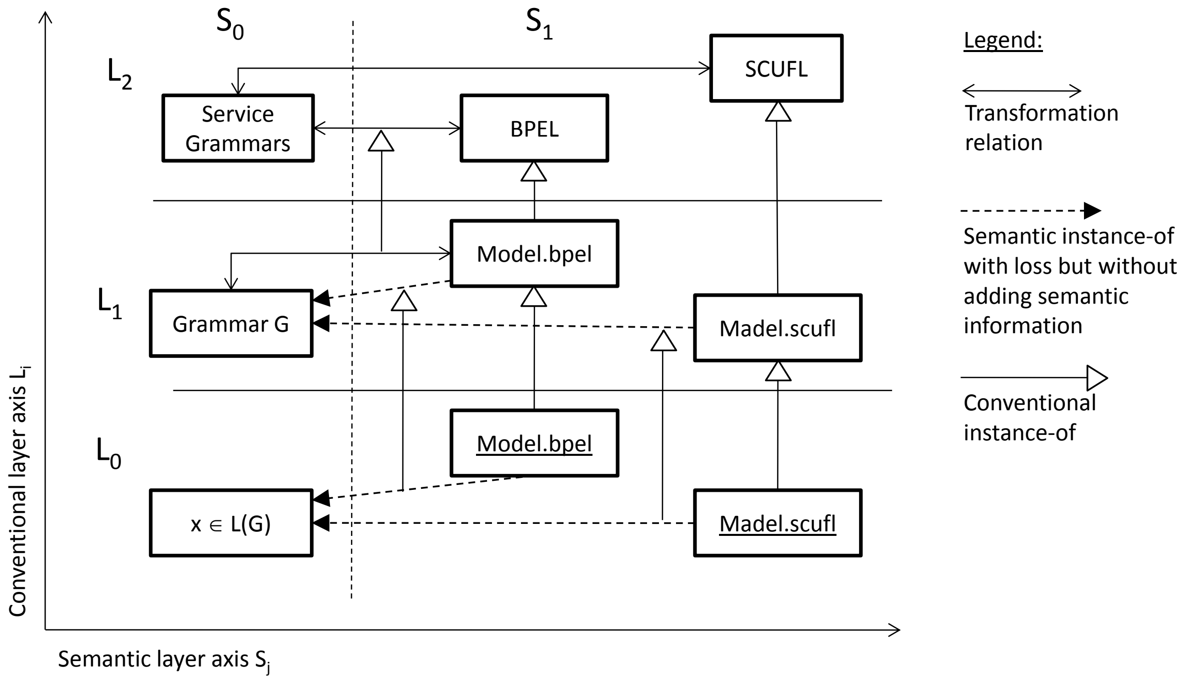


Figure 7: Conventional, i.e. horizontal inheritance of semantic dependencies

Further semantic dependencies next to the instance-of dependency allow to enrich the information specified on vertical meta-modeling layers. For example, Figure 7 already introduced the transformation relation that crosses the border of vertical layers but does not cross the border of horizontal layers. Furthermore, Figure 8 introduces a "contained in" relation representing a semantic dependency next to the semantic instance-of dependency. The semantic instance-of dependency creates the vertical hierarchy whereas the relation "contained in" does not cross the border of its particular vertical layer. However, *semantic dependencies, i.e. relations that do not cross the border of vertical layers may be semantically inherited to an adjacent vertical layer* if corresponding models are semantically inherited. That means, the inherited relations reflect the modified relations between modified models. In this case the same semantic inheritance for models and relations is used as the modifications in models and relations correlate. Additionally, *semantic dependencies, i.e. relations that* *do not cross the border of horizontal layers may be conventionally inherited to an underlying horizontal layer* if corresponding models are conventionally inherited (cf. Transformation relation in Figure 7).

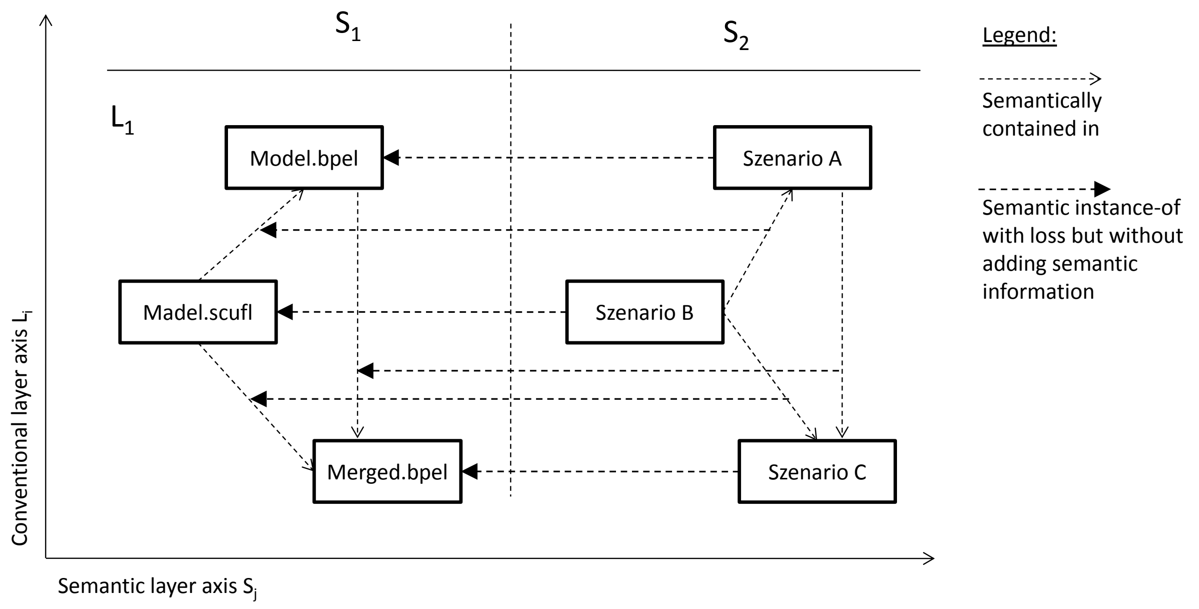

S1
S2
Legend:
Semantically contained in
Semantic instance-of with loss but without adding semantic information
L1
Model.bpel
Szenario A
Madel.scufl
Szenario B
Merged.bpel
Szenario C
Conventional layer axis Li
Semantic layer axis Sj

Figure 8: Semantic, i.e. vertical inheritance of semantic dependencies

In summary, semantic dependencies, i.e. relations that do not cross the borders of vertical and horizontal layers can be inherited to vertical as well as horizontal layers if corresponding models are inherited. For example, Figure 9 shows the dependency “contained in” initially specified on L_1+S_1. This dependency is coventionally inherited on L_0+S_1 and afterwards semantically inherited on L_0+S_2. That means, semantic dependencies should be specified on the highest horizontal and lowest vertical layer for fully exploiting the inheritance of dependencies in corresponding meta-modeling hierarchies.

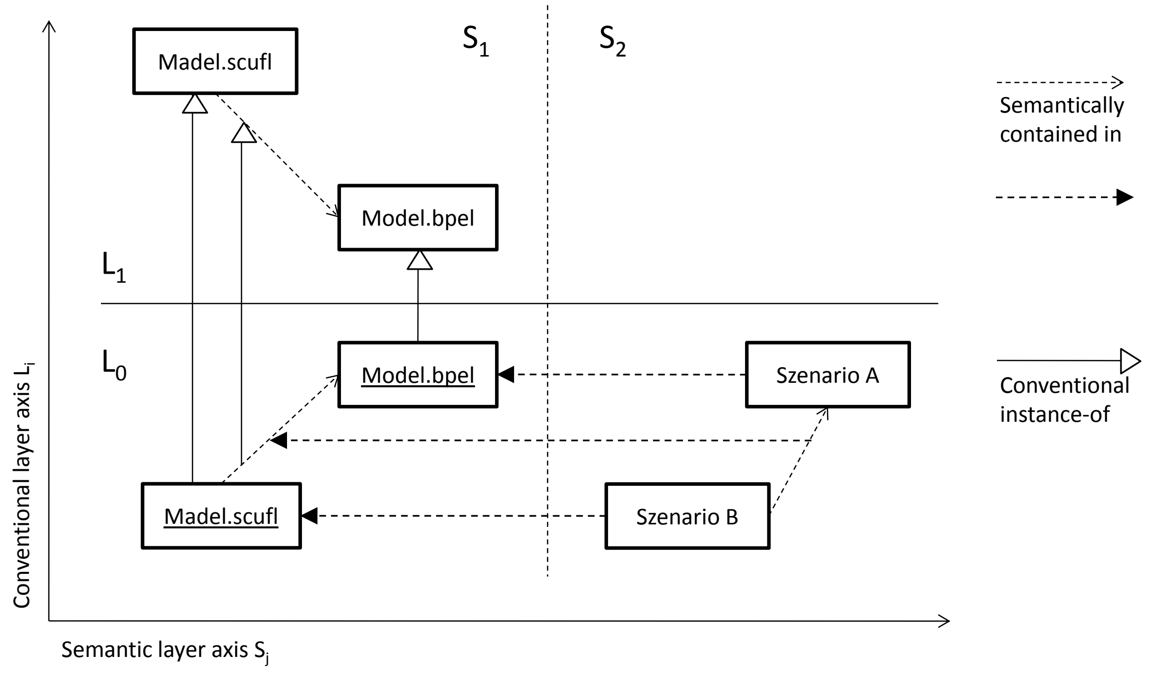


Figure 9: Conventional and semantic inheritance of semantic dependencies

4. Impact of Orthogonal Meta-Modeling on Model Transformation

For model transformation information about the semantics of the source model as well as the syntax of the source and the target model is required in general. The MDA pattern (Mukerji, J. & Miller, J. 2001) specifies the need for inputting a platform model (PM) for the transformation from a platform-independent model (PIM) to a platform-specific model (PSM). The PIM provides required information about the semantics and the syntax of the source model. Additionally, the PM provides required information about the syntax of the target model, i.e. the PM determines the meta-model mapping that is required for the particular transformation.

Transformations from PIM (PSM) to PIM (PSM) allowing a model refinement or refactoring also use mappings similar to transformations from PIM to PSM. However, a transformation from PIM to PSM always requires mappings between two conventional meta-models whereas a transformation from PIM (PSM) to PIM (PSM) does not need a mapping between two conventional meta-models in case the same language is used for the input and output model. In summary, different kinds of mappings are used for transformations in the model-driven architecture: At first, a transformation from PIM to PSM uses a *syntactic mapping* that covers the dependency between a syntactic meta-model and its syntactic instance. Secondly, a transformation from PIM (PSM) to PIM (PSM) uses a *semantic mapping* that covers the dependency between a semantic meta-model and its semantic instance. In case the language also needs to be changed by the transformation from PIM (PSM) to PIM (PSM) both kinds of mappings, i.e. a semantic mapping as well as a syntactic mapping need to be used.

Mens T. and van Gorp P. (2006) used the term endogenous for transformations that are based on a single syntactic meta-model and the term exogenous for transformations that are based on different syntactic meta-models, i.e. for transformations between different languages. The approach at hand relates exogenous transformations with syntactic mappings whereas endogenous do not require syntactic mappings necessarily. Furthermore, Mens T. and van Gorp P. (2006) used the term vertical for transformations where the input model is located on another abstraction level than the output model. In contrast, horizontal transformations cover models on the same abstraction level. However, a vertical transformation requires a semantic mapping and exactly covers the dependency between models on different vertical layers, i.e. semantic layers introduced by the approach at hand. In summary, vertical transformations require semantic mappings whereas horizontal transformations do not necessarily. Table 1 summarizes the required mappings for the orthogonal dimensions of transformations introduced by Mens T. and van Gorp P. (2006). The endogenous, horizontal transformation does not require a specific kind of mapping but is allowed to be based on syntactic or semantic information.

Table 1: Orthogonal dimensions of model transformation with required mappings

	Horizontal	**vertical**
endogenous	syntactic OR semantic mapping	semantic mapping
exogenous	syntactic mapping	semantic AND syntactic mapping

Conventional meta-modelling allows the separation of syntactic meta-models and their syntactic instances. Therefore, conventional meta-modeling enables to cover syntactic mappings that are always defined between models on the same conventional, i.e. horizontal meta-modeling layer. A syntactic mapping can be used for transformations between syntactic instances. For example, Figure 10 shows a syntactic mapping between the meta-models Service Grammars and BPEL on L_2. Based on the syntactic mapping the exogenous, horizontal transformation between syntactic instances on L1, i.e. from Grammar G to Model1.bpel can be realized by using the conventional inheritance.

Similarly, orthogonal meta-modeling allows the separation of semantic meta-models and their semantic instances enabling the specification of semantic mappings. In particular, semantic mappings can be determined based on semantic instance-of dependencies between different vertical layers. For example, Figure 10 shows a semantic instance-of dependency between the vertical layers S_0 and S_1. *The backward relation of the semantic instance-of dependency specifies a semantic mapping.* Based on the semantic mapping an endogenous, vertical transformation from the semantic meta-model Model1.bpel on S_0 to a semantic instance Model2.bpel on S_1 can be realized. Furthermore, a semantic mapping specified by the backward relation of a complete instance-of dependency can be used to realize a vertical transformation from a semantic instance to a semantic meta-model (cf. Figure 5).

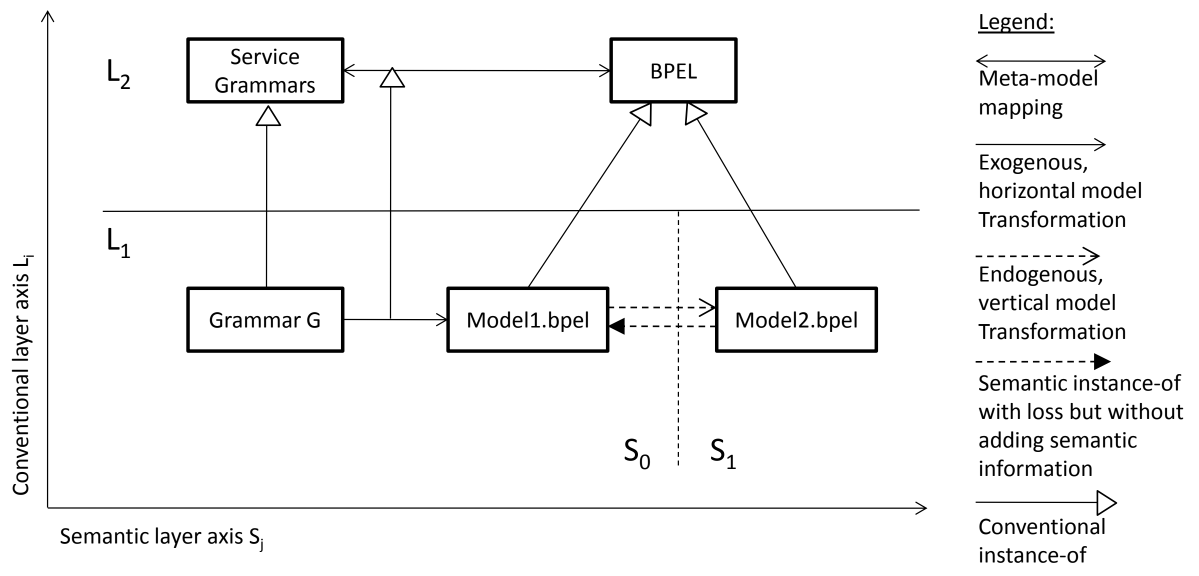

Legend:
Meta-model mapping
Exogenous, horizontal model Transformation
Endogenous, vertical model Transformation
Semantic instance-of with loss but without adding semantic information
Conventional instance-of
Service Grammars
BPEL
L2
L1
Grammar G
Model1.bpel
Model2.bpel
S0
S1
Conventional layer axis Li
Semantic layer axis Sj

Figure 10: Exogenous, horizontal model transformation from Grammar G to Model1.bpel and endogenous, vertical model transformation from Model1.bpel to Model2.bpel

Figure 11 illustrates an exogenous, vertical transformation from a Grammar G on L_1+S_0 to a BPEL-based model on L_1+S_1. The exogenous, vertical transformation requires information about the syntax of source and target model as well as information about the semantics of the source and target model. The syntax-related information is provided by the syntactic mapping between particular meta-models on the next higher horizontal layer. The semantics-related information is provided by the semantic mapping specified by the backward relation of the particular semantic instance-of dependency. In order to combine the syntax-related information and the semantics-related information for the exogenous, vertical model transformation an additional semantic instance-of dependency and therefore an additional orthogonal semantic hierarchy needs to be introduced. The additional hierarchy enables to (semantically) inherit the syntactic mapping, i.e. the meta-model mapping on L_1 from the semantic layer T_0 to the layer T_1. As already discussed at hand of Figure 7 the conventional inheritance cannot be used for the combination as the exogenous, vertical model transformation is not a syntactic instance of the meta-model mapping. Furthermore, the semantic inheritance between models on the layers S_0 and S_1 cannot be used as it does not correlate with the syntactic mapping (cf. Figure 8). Instead, an additional semantic instance-of dependency needs to be introduced allowing the semantic inheritance of the meta-model mapping, i.e. allowing the inheritance of dependencies between syntactic meta-models.

In summary, next to the separation of models *orthogonal semantic meta-modeling allows to determine the nature of vertical model transformations*: Vertical transformations maintaining a language (i.e. endogenous, vertical transformations) use a single semantic instance-of dependency in order to determine the transformation mapping covering semantics-related information but no syntax-related information. In contrast, vertical transformations between different languages (i.e. exogenous, vertical transformations) use two different semantic instance-of dependencies for inheriting from a semantic meta-model and from a syntactic mapping (conventional meta-model mapping) in order to determine the transformation mapping combining semantic and syntactic information. In particular, a single orthogonal meta-modeling hierarchy is sufficient for endogenous, vertical transformations whereas exogenous, vertical transformation require two orthogonal meta-modeling hierarchies.

Orthogonal meta-modeling covering semantic instance-of dependencies refines the MDA pattern by a *semantics model* (SM) providing the particular semantics-related information for the transformation. Similar to the PM the SM represents input information for the model transformation. The PM represents syntax-related input for the transformation, i.e. determines the particular meta-model mapping realizing the syntactic mapping. The SM represents semantics-related input for the transformation, i.e. determines the particular semantic instance-of dependency determining the semantic mapping. That means, the SM specifies the modification operation (e.g. semantic instance-of dependency with loss but without adding semantic information) that is intended to be applied on the model content.

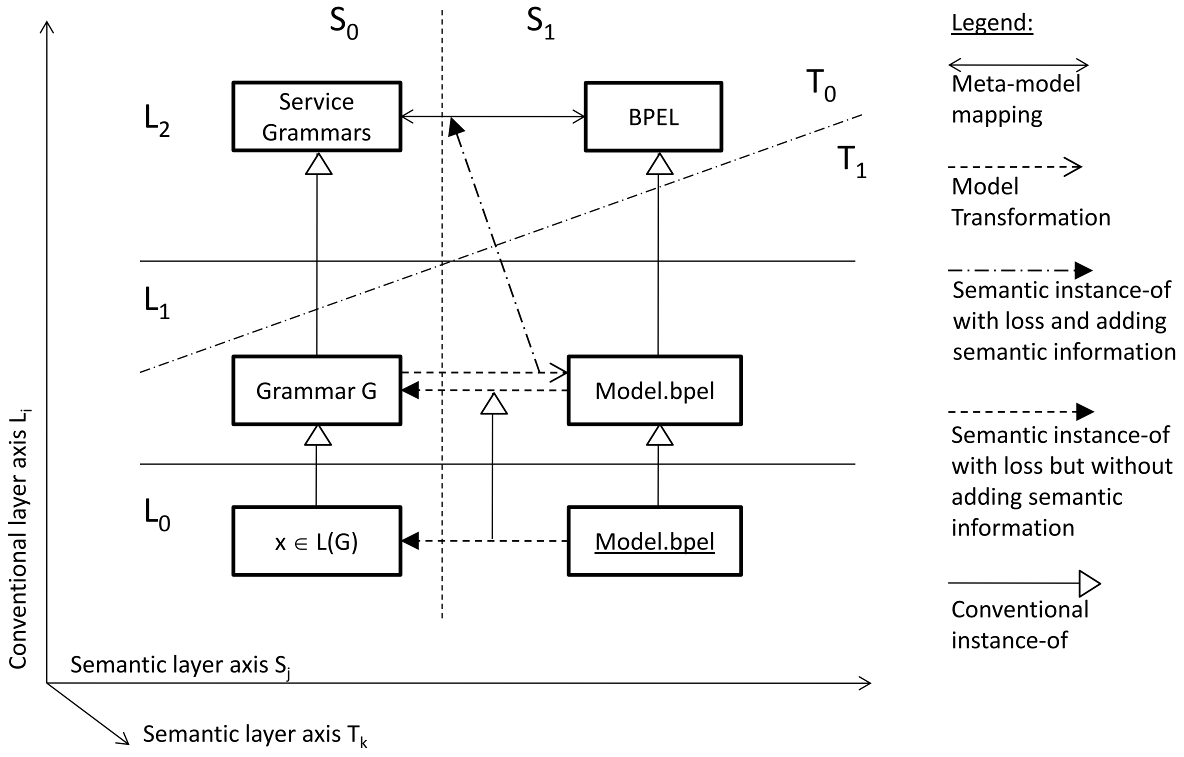


Figure 11: Exogenous, vertical model transformation from Grammar G to Model.bpel

Figure 12 illustrates the refinement of the MDA pattern. A transformation implementing a language transformation without modifications to the model content requires the input of a PM as shown in Figure 12(a). A transformation implementing automated modifications of the model content without changing the language require the input of an SM as shown in Figure 12(b). Finally, a transformation implementing a language transformation in combination with automated modifications to the model content require the input of PM and SM as shown in Figure 12(c). As shown in Figure 12(c) the transformation from a PIM to a PSM may require modifications of the model content. For instance, attributes or operations that are required for a realization on the particular platform need to be added or some model content needs to be deleted if the particular platform does not support appropriate mechanisms. In that cases the SM provides information how to derive attribute values or operation implementations for the added model content or information about model content depending on model content that needs to be deleted. That means, the extension of the MDA pattern by a SM improves the ability to automate particular transformations. However, transformations from PIM to PSM requiring no modifications of model content use the SM determining the identity relation allowing the (semantic) inheritance of the unmodified model content in the PIM.

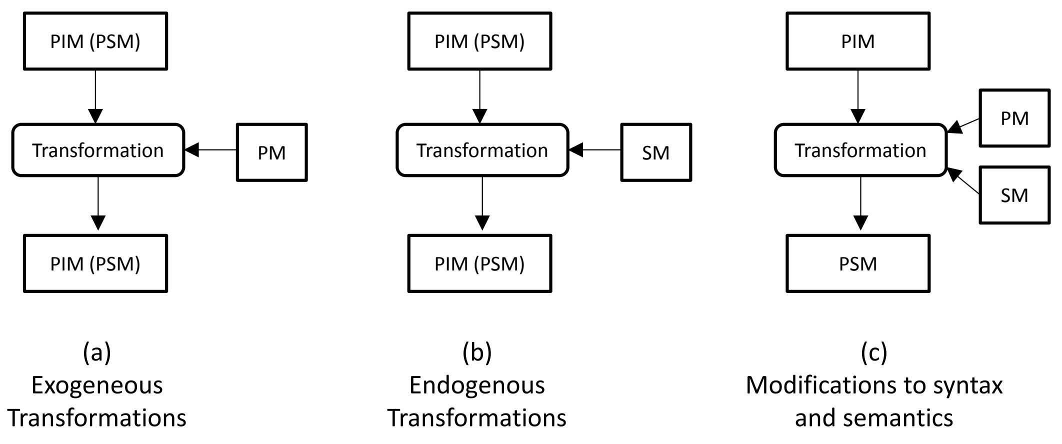


Figure 12: MDA pattern extended by a semantic model (SM)

5. Summary

The approach at hand introduced orthogonal meta-modeling for enriching conventional meta-modeling with semantic dependencies. A specific semantic instance-of dependency (with loss but without adding semantic information) is used to introduce a vertical hierarchy additionally to the conventional horizontal meta-modeling hierarchy. A semantic instance-of dependency enables the grouping of models on the same conventional horizontal layer based on semantic information. That means, the orthogonal meta-modeling allows to structure models on conventional meta-modeling layers on additional semantic information. Next to the semantic instance-of dependency additional semantic dependencies can be specified in the meta-modeling hierarchy. These additional semantic dependencies can be inherited on adjacent meta-modeling layers if they do no cross the border of the particular hierarchy.

The introduced semantic instance-of dependency enables the improvement of automated model transformations. In particular, the backward relation of a semantic instance-of dependency leads to a semantic mapping that is suitable to influence a model transformation. In general, a semantic meta-model can be transformed to a semantic instance by using the semantic mapping specifying which model content from the semantic meta-model is inherited in the semantic instance. Furthermore, the backward relation of the semantic instance-of dependency can be used for the generation of a semantic meta-model if the instance-of dependency is assumed to be complete. That means, the model content of all semantic instances is combined in the content of the semantic meta-model. However, some model content may be lost in a transformation round trip. That means, the transformation from a semantic meta-model M to some semantic instances and a following transformation from the semantic instances back to a semantic meta-model M’ does not ensure the equality of the semantic meta-models M and M’. Instead some model content is lost if this model content is abstracted in all semantic instances.

Additional semantic dependencies next to the semantic instance-of dependency help to improve the expressiveness in the design of meta-modeling-based software architectures. In particular, an advanced management of dependencies between models in a layer-based architecture is enabled. For example, Figure 6 and Figure 8 show different approaches for realizing the merging of models. The design in Figure 6 uses multiple inheritance and a semantic instance-of dependency allowing to add semantic information for merging models. In contrast, the design in Figure 8 uses a semantic instance-of dependency without allowing to add semantic information. In order to specify the dependency between a merged model in the particular individual models an additional semantic dependency is introduced. However, the design in Figure 6 allows to derive a merged model based on the particular semantic instance-of dependency, i.e. the merged model can be derived by an automated transformation. Therefore, the merged model is on another vertical layer than the individual models. In contrast, the design in Figure 8 “only” specifies the particular dependency but does not support the derivation of the merged model by an automated transformation. Therefore, the merged model is on the same vertical layer as the individual models.

6. Related Work

Semantic information is typically provided by ontologies. In the field of service-based computing ontologies are very prominent to enrich the information about a web service for enabling automated semantic-based service discovery. For example, OWL-S (Martin, D. 2004) and WSMO (Roman, D. et al. 2005) are well-established languages for the specification of ontologies for web services. Service-oriented computing typically handles layer-based systems by considering modeling languages, models, and model instances. That means, meta-modeling is essential in service-oriented computing with focus on functionality-oriented models. In order to consider the combination of functionality-oriented models with semantic-oriented models, i.e. ontologies the ontology language WSMO briefly discusses the relation of specified ontologies to functionality-related models covering the OMG’s four meta-modeling layers.

Next to the automated semantic-based service discovery other alternatives for the application of semantic enrichment in service-based computing exist, e.g. automatic service composition and auto-completion of composition models (Karastoyanova, D. et al. 2009). The approach by Pahl, C. (2007) presents an ontology-based transformation for service-based software systems. This approach does not use meta-modeling but semantic enrichment that is used to guide transformations similar to the approach at hand. However, the approach hand introduces orthogonal meta-modeling layers for semantic enrichment in order to allow an appropriate correlation of conventional and additional semantic information. The correlation enables to reason on particular information and to create new information, e.g. by inheriting dependencies on adjacent meta-modeling layers.

Meta-models and ontologies are closely related in general. For example, Guizzardi (2007) discusses in detail the correlation of meta-models and ontologies specifying additional domain logic. Davis et al. (2003) also discuss the relation of meta-models and ontologies but applies meta-modeling on ontologies for an appropriate comparison of ontologies. In contrast, the approach at hand combines meta-modeling and ontologies for semantic enrichment of meta-modeling hierarchies in order to allow the grouping of models on the same conventional horziontal meta-modeling layer and an automated model transformation. For achieving the same expressiveness as ontologies additional semantic dependencies next to the hierarchy-specific instance-of dependency need to be enabled in semantic meta-modeling. The approach at hand discusses semantic dependencies as well as their inheritance on horizontal and vertical meta-modeling layers in detail.

The approach presented in (Bencomo, N. & Blair, G. 2005) already uses orthogonal semantic layers for a meta-model-based application. The approach uses orthogonal layers to separate the meta-model-based application itself from the specification of the architecture, interfaces, and interceptions. However, orthogonal meta-modeling is not discussed in general and relations between models on different horizontal and vertical layers are not covered. Furthermore, Atkinson and Kühne (2003) apply orthogonal semantic meta-modeling layers in the field of linguistic. However, the approach is restricted to a single so called ontological instance-of dependency. Further instance-of dependencies and additional semantic dependencies are not considered. In contrast, the approach at hand aims for an introduction of orthogonal meta-modeling hierarchies in general allowing the correlation of dependencies on orthogonal layers.

References

Atkinson, C. & Kühne, T., 2003. Model-driven development: A metamodeling foundation, IEEE Software 20(5): Atkinson, C. & Kühne, T., 2003. Model-driven development: A metamodeling foundation, *IEEE Software* 20(5): 36-41

Badouel, E. & Darondeau, P. 2011. Petri Net Synthesis, Springer, http://www.irisa.fr/s4/wg22/phd/RECENT/ENSynth.pdf

Bencomo, N. & Blair, G., 2005. Preening: reflection of models in the mirror a meta-modelling approach to generate reflective middleware configurations. In *Proceedings of the 2005 international conference on Satellite Events at the MoDELS (MoDELS'05)*

Davies, I. & Green, P. & Milton, S. & Rosemann, M., 2003. Using Meta Models for the Comparison of Ontologies. In *Evaluation of Modeling Methods in Systems Analysis and Design Workshop - EMMSAD'03*

Görlach, K. & Leymann, F. & Claus, V., 2013. Unified Execution of Service Compositions. *Proceedings of the 6th IEEE International Conference on Service Oriented Computing & Applications, SOCA 2013*

Görlach, K., 2013. A Generic Transofrmation of Existing Service Composition Models to a Unified Model. *Technical Report* 2013/01, University of Stuttgart, Germany, ftp://ftp.informatik.uni-stuttgart.de/pub/library/ncstrl.ustuttgart_fi/TR-2013-01/TR-2013-01.pdf

Guizzardi, G., 2007. On Ontology, ontologies, Conceptualizations, Modeling Languages, and (Meta)Models. In *Proceedings of the 2007 conference on Databases and Information Systems IV: Selected Papers from the Seventh International Baltic Conference DB&IS'2006*

Karastoyanova, D. & van Lessen, T. & Leymann, F. & Ma, Z. & Nitzsche, J. & Wetzstein, B., 2009. *Semantic Business Process Management: Applying Ontologies in BPM.* Handbook of Research on Business Process Modeling, Information Science Publishing

Kopp, O. & Görlach, K. & Karastoyanova, D. & Leymann, F. & Reiter, M. & Schumm, D. & Sonntag, M. & Strauch, S. & Unger, T. & Wieland, M. & Khalaf, R., 2011. A Classification of BPEL Extensions. In: *Journal of Systems Integration*. Vol. 2(4): 3-28

Liu, D. & Shen, M., 2003. Workflow modeling for virtual processes: an order-preserving process-view approach. *Information Systems* 28(6): 505–532

Martin, D., 2004. OWL-S: Semantic Markup for Web Services. W3C, http://www.w3.org/Submission/2004/SUBM-OWL-S-20041122/

Mukerji, J. & Miller, J., 2001. Model Driven Architecture, OMG, www.omg.org/cgi-bin/doc?ormsc/2001-07-01

OASIS, 2007. Web Services Business Process Execution Language Version 2.0. OASIS Standard; http://docs.oasis-open.org/wsbpel/2.0/wsbpel-v2.0.html

Oinn, T. & Greenwood, M. & Addis, M. & Nedim Alpdemir, M. & Ferris, J. & Glover, K. & Goble, C. & Goderis, A. & Hull, D. & Marvin, D. & Li, P. & Lord, P. & Pocock, M. & Senger, M. & Stevens, R. & Wipat, A. & Wroe, C., 2006. Taverna: lessons in creating a workflow environment for the life sciences. *Concurrency and Computation: Practice and Experience*, 18(10): 1067-1100

OMG, 2011. Business Process Model and Notation (BPMN 2.0). OMG Standard, http://www.omg.org/spec/BPMN/2.0/PDF

Pahl, C., 2007. Semantic model-driven architecting of service-based software systems. *Information and Software Technology* 49(8): 838-850

Pesic, M., 2008. *Constraint-Based Workflow Management Systems: Shifting Control to Use*rs. PhD thesis, Technische Universiteit Eindhoven

Reisig, W., 2010. *Petrinetze: Modellierungstechnik, Analysemethoden, Fallstudien.* Leitfäden der Informatik, Vieweg+Teubner

Roman, D. & Lausen, H. & Keller, U., 2005. Web Service Modeling Ontology (WSMO)“ W3C, http://www.w3.org/Submission/WSMO-primer/

Schumm, D. & Leymann, F. & Streule, A., 2010. “Process Viewing Patterns. In: *Proceedings of the 14th IEEE International EDOC Conference (EDOC 2010)*

JEL Classification: D80, M15