Addressing the Need for Strict Meta-Modeling in Practice - A Case Study of AUTOSAR

Darko Durisic[1], Miroslaw Staron[2], Matthias Tichy[3] and Jörgen Hansson[4]
[1]*Volvo Car Group, Gothenburg, Sweden*
[2]*Chalmers | University of Gothenburg, Sweden*
[3]*University of Ulm, Germany*
[4]*University of Skövde, Sweden*
darko.durisic@volvocars.com, miroslaw.staron@cse.gu.se, matthias.tichy@uni-ulm.de, jorgen.hansson@his.se

Keywords: Strict meta-modeling principle, domain-specific meta-modeling, deep meta-modeling, AUTOSAR

Abstract: Meta-modeling has been a topic of interest in the modeling community for many years, yielding substantial number of papers describing its theoretical concepts. Many of them are aiming to solve the problem of traditional UML based domain-specific meta-modeling related to its non-compliance to the strict meta-modeling principle, such as the deep meta-modeling approach. In this paper, we show the practical use of meta-models in the automotive development process based on AUTOSAR and visualize places in the AUTOSAR meta-model which are broken according to the strict meta-modeling principle. We then explain how the AUTOSAR meta-modeling environment can be re-worked in order to comply to this principle by applying three individual approaches, each one combined with the concept of Orthogonal Classification Architecture: UML extension, prototypical pattern and deep instantiation. Finally we discuss the applicability of these approaches in practice and contrast the identified issues with the actual problems faced by the automotive meta-modeling practitioners. Our objective is to bridge the current gap between the theoretical and practical concerns in meta-modeling.

1 INTRODUCTION

The use of meta-models (Saeki and Kaiya, 2007) in the development of large software systems has been in focus of the modeling research community for more than a decade. Object Management Group (OMG) provides a standardized framework for developing models as instances of domain-specific meta-models, which are instances of UML, which in turn is an instance of MOF (Meta-Object Facility) (MOF, 2004). This forms a commonly referred 4-layer meta-modeling hierarchy of MOF. OMG also defines a standard for the exchange of models between domain-specific modeling tools based on XML known as XMI (XML Metadata Interchange) (XMI, 2011).

Numerous papers have pointed out that the framework of OMG fails to comply to one of the basic meta-modeling principles known as the strict meta-modeling (Atkinson and Kühne, 2002). According to this principle, each model element on one meta-modeling layer is an instance of exactly one element on the layer above, with no other possible relationships between the layers. In case of UML, instances of domain-specific classifiers are at the same time instances of the UML meta-class ”Object” (the problem known as dual classification (Atkinson and Kühne, 2001)). Additionally, the use of UML stereotypes for giving additional semantics to the domain-specific meta-model elements and/or their instances presumes that the language definition of stereotypes and their instances reside on the same layer as UML. For this reason, UML based meta-modeling fails to comply to the strict meta-modeling principle and therefore can be classified as ”loose” meta-modeling.

A number of proposal have been made to introduce strictness into the UML based meta-modeling. Probably the most discussed approach in the literature is the one proposed by Atkinson and Kühne known as the deep meta-modeling, which relies on the concepts of Orthogonal Classification Architecture (OCA) and deep instantiation (Atkinson and Kuhne, 2003). OCA is able to express the distinction between the linguistic and ontological (semantic) instantiations in the meta-modeling hierarchy, thus solving the dual classification problem. Deep instantiation on the other hand enables the existence of more than two ontological layers which can be used to give additional semantics to the meta-classes, instead of stereotypes.

Despite this, the practical consequences of loose meta-modeling remain unclear to the majority of practitioners. Additionally, there is a lack of industrial applications of the strict meta-modeling approaches, such as deep meta-modeling, and the analysis of their benefits/drawbacks in comparison to the UML approach. We believe that these studies are needed to bridge the current gap between the work of meta-modeling researchers and the problems faced by the meta-modeling practitioners.

In this paper, we explore the use of domain-specific meta-modeling in the context of automotive model driven development where, with the introduction of AUTOSAR (AUTomotive Open System ARchitecture) (AUT, 2003), UML-based meta-modeling is taken as a basis for specifying the architecture of automotive systems. The goal of this paper is to address the following research questions:

- **Q1** What are the consequences of UML based loose meta-modeling in the automotive domain?
- **Q2** What are the drawbacks of approaches for assuring strictness of the AUTOSAR meta-model?
- **Q3** What are the practical meta-modeling concerns of the automotive modeling practitioners?

We achieve this by visualizing places in the AUTOSAR meta-model which are not compliant to the strict meta-modeling principle and re-working the AUTOSAR meta-modeling environment according to the rule of strictness using three distinct approaches: UML extension, prototyping pattern and deep instantiation. We combine each approach with the OCA representation of the layers and discuss their benefits and drawbacks in practice. Finally, we explain why the issues of loose meta-modeling are not the primary concern of the automotive practitioners and contrast them with the actual problems they face in their work.

The rest of this paper is structured as follows: Section 2 describes the role and hierarchy of the AUTOSAR meta-model in the automotive modeling environment. Section 3 applies three approaches for assuring compliance of the AUTOSAR meta-model to the strict meta-modeling principle. Section 4 discusses the benefits and drawbacks of these approaches in practice. Finally we conclude in Section 5 with some guidelines for future meta-modeling research.

2 Automotive Modeling

Automotive software system is a distributed system with typically more than 100 Electronic Control Units (ECUs) today responsible for executing allocated vehicle functions. Apart from its distributed organization, the development of the automotive software systems is also distributed as ECUs are usually developed by a hierarchy of suppliers, e.g. application software, middleware and hardware suppliers. In order to facilitate this distributed development, AUTOSAR standard was introduced as a joint partnership of car manufacturers and their suppliers.

The goal of AUTOSAR is to standardize the architecture of the automotive systems and their development methodology. Figure 1 shows a common AUTOSAR based model-driven development process until the generation of the ECU source code.

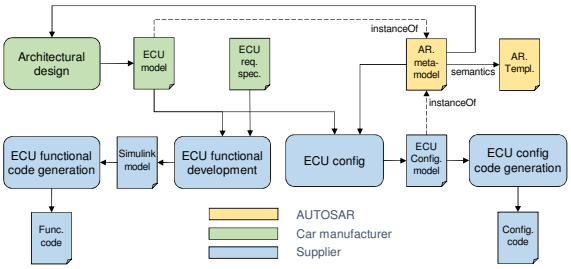

Figure 1: AUTOSAR based software development process

The development starts with the *Architectural design*, which includes the definition of the ECUs and allocation of the software components onto them. As ECU development is usually done by different suppliers, the complete architectural model is divided into a number of *ECU model*s, each one delivered to the chosen supplier. Because different car manufacturers and suppliers may use different modeling tools, assuring their interoperability is quite challenging.

To facilitate this interoperability, *AUTOSAR meta-model* is defined as a standardized domain-specific modeling language for the architectural *ECU model*s, i.e. an *ECU model* represents an instances of the *AUTOSAR meta-model* that specifies its syntax. The semantics is specified in the specifications called *AUTOSAR templates* (Gouriet, 2010). Therefore the *AUTOSAR meta-model* serves as a basis for the development of the automotive modeling tool-chain.

Figure 2 shows an example of the AUTOSAR meta-model which is used to specify the allocation of the software components onto ECUs, and a corresponding model allocating the *EnginePowerUnit* component onto the *EngineControlModule* ECU (Durisic et al., 2014). The AUTOSAR models are expressed in XML and they are validated by the XML schema which is generated from the AUTOSAR meta-model using a set of defined transformation.

The *ECU model* and the *ECU requirements specification* defining the behavior of the allocated software components serve as basis for the *ECU functional development*. This is usually done in Simulink

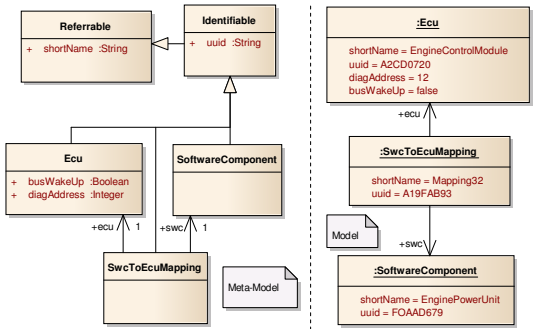

Figure 2: AUTOSAR meta-model example

as it supports automated *ECU functional code generation* from the *Simulink model*s (Liu et al., 2013).

Apart from being input to the functional development, the *ECU model*s also serve as basis for the *ECU configuration*, i.e. configuration of the middleware (e.g. ECU communication, diagnostics). One ECU can be configured using a number of parameters. The values of certain parameters can be automatically generated from the attributes of the *ECU model*, e.g. signal transmission period. Other parameters, e.g. related to the operating system, need to be configured in the ECU configuration tools (Lee and Han, 2009). Based on the complete *ECU configuration model* instantiating the *AUTOSAR meta-model*, *Configuration code* (C-structs) can be automatically generated in the *ECU configuration code generation* phase.

The final ECU software is obtained by compiling and linking the *Functional code* generated from the *Simulink model*s, the *Configuration code* generated from the *ECU model*s and completed in the ECU configuration tools and the actual implementation of the AUTOSAR middleware (commonly known as AUTOSAR basic software). The basic software is entirely specified by AUTOSAR and it is usually implemented outside of the model driven approach. The language mostly used in the automotive domain is C.

Based on the described process, we can conclude that the AUTOSAR meta-model plays an important role in the architectural design of the system and the configuration of the ECU basic software.

2.1 AUTOSAR Meta-Model Hierarchy

MOF (Meta-Object Facility) (MOF, 2004), an accepted standard for model-driven engineering, defines the following 4-layer meta-modeling hierarchy:

1. **M3**: MOF meta-meta-model
2. **M2**: UML meta-model
3. **M1**: Application model
4. **M0**: Application data

These layers are connected by the instantiation mechanism, i.e., each layer represents an instance of the layer above, except for the top layer which is considered to be an instance of itself. According to the strict meta-modeling principle (Atkinson, 1998), each model element on one layer is an instance of exactly one element on the layer above.

The meta-modeling hierarchy of MOF exhibits the property known as dual classification (Atkinson and Kühne, 2002). This is a consequence of the two notions of the *instanceOf* relationship - linguistic, defining the language (e.g. *EngineControlModule* is a UML *Object*), and ontological, defining the semantics (e.g. *EngineControlModule* is an *ECU*) (Atkinson and Kuhne, 2003). This means that the objects on the *M1* layer are both ontological instances of the *M1* classes and linguistic instances of the *M2* meta-class *Object*, thus breaking the strict modeling principle.

To resolve this problem, MOF considers the strict meta-modeling principle only for the linguistic notion of the *instanceOf* relationship. AUTOSAR, however, attempts to visualize both linguistic and ontological layers in the meta-modeling hierarchy thus defining the following 5 meta-modeling layers (AUT, 2014):

1. **ARM4**: MOF meta-meta-model
2. **ARM3**: UML meta-model and AR profile
3. **ARM2**: AUTOSAR meta-model
4. **ARM1**: AUTOSAR model
5. **ARM0**: AUTOSAR run-time objects

By examining the semantics of the layers, we can see that the *ARM1* (containing objects) is an ontological instance of the *ARM2* (containing classes) and both the *ARM1* and *ARM2* are linguistic instances of the *ARM3* (containing meta-classes *Class* and *Object*). Therefore the illustration of the AUTOSAR layers breaks the strict meta-modeling principle as the *ARM1* objects directly instantiate (linguistically) the *ARM3* meta-class *Object*. On the *ARM3* layer, AUTOSAR also extends the UML meta-model with its own AUTOSAR Template Profile (ATP). An simplified example of the AUTOSAR meta-layers and the ATP profile is presented in Figure 3.

AUTOSAR defines a number of *umlStereotype*s in the *atpProfile* and we show one of them, *atpVariation*, applicable to *umlClass*es (indicates that a class may have different variants), with the corresponding *atpBindingTime* tagged value (indicates when the variant shall be bound). All stereotypes are branding both classifiers and instances (Atkinson et al., 2002).

Figure 3 also shows three types of the non-compliance of the AUTOSAR meta-modeling hierarchy with respect to the strict meta-modeling principle (indicated by the numbers [1], [2] and [3]):

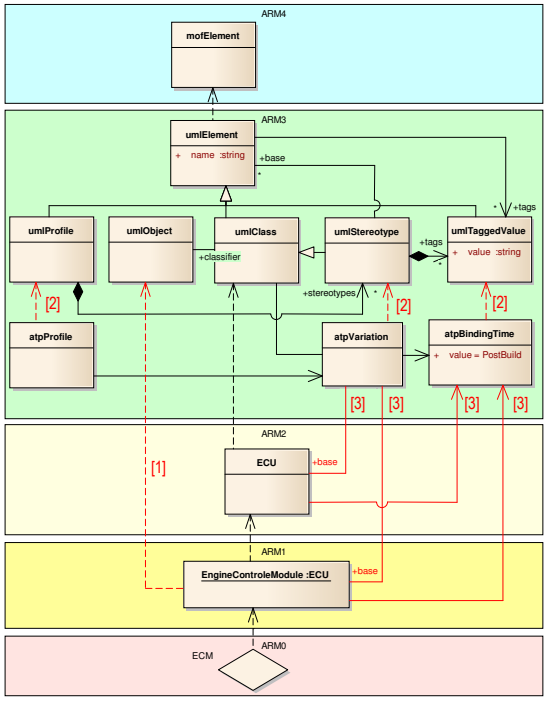

Figure 3: Example of the AUTOSAR meta-layers

[1] The *EngineControlModule* on the *ARM0* layer is a direct linguistic instance of the *umlObject* on the *ARM3* layer (dual classification).

[2] The *atpProfile*, *atpVariation* and *atpLatestBindingTime* represent ontological instances of the *umlProfile*, *umlStereotype* and *umlTaggedValue*, respectively, even though they reside on *ARM3*.

[3] The non-*instanceOf* relationships between the *atpVariation* / *atpLatestBindingTime* and the *ECU* / *EngineControlModule* cross the layer boundaries.

3 Assuring Strictness of AUTOSAR

The dual classification problem [1] can be solved using the OCA with two dimensions, where the ontological instantiation is depicted horizontally and the linguistic vertically. The problems related to the use of stereotypes [2] and [3] can be solved using different approaches and we show in this paper three of them combined with OCA: UML extension, prototypical pattern and deep instantiation (referred to as deep meta-modeling in combination with OCA).

UML Extension: In this approach, the *umlClass* on the linguistic layer *L2* is directly extended by the *atpVariableClass* with the attribute *atpBindingTime* by means of inheritance, as shown in Figure 4. Green color indicates linguistic instantiation (between *L* layers) while blue color indicates ontological instantiations (between *O* layers). The linguistic instantiation of the *atpBindingTime* attribute from the *L2* layer shall be done in a static manner, i.e. all *O0* instances of *ECU* shall have "*PostBuild*" *atpBindingTime*. The *L0* layer is omitted due to its questioned value in the meta-modeling hierarchy (Harel and Rumpe, 2004).

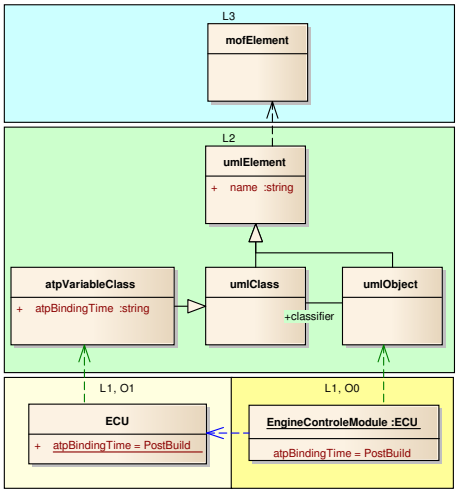

Figure 4: Example of UML extension

Prototypical Pattern: In this approach, the *umlClass* on the *L2* layer is instantiated into a number of *atpVariable* classes for different values of the *atpBindingTime*, e.g. *atpPreBuildVariableClass* and *atpPostBuildVariableClass*, as shown in Figure 5. The actual domain-specific *L1* classes, such as *ECU*, are then inherited from the right *atpVariableClass* depending on the intended value of the *atpBindingtime*.

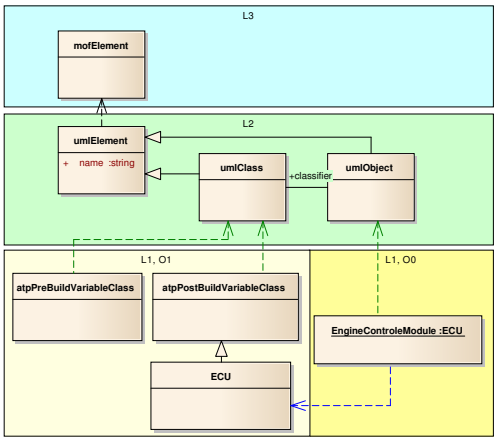

Figure 5: Example of prototypical pattern

Deep Instantiation: In this approach, the hierarchy of more than two ontological instantiations is allowed, i.e. domain-specific classes can be instantiated by other classes, not just objects as in UML.

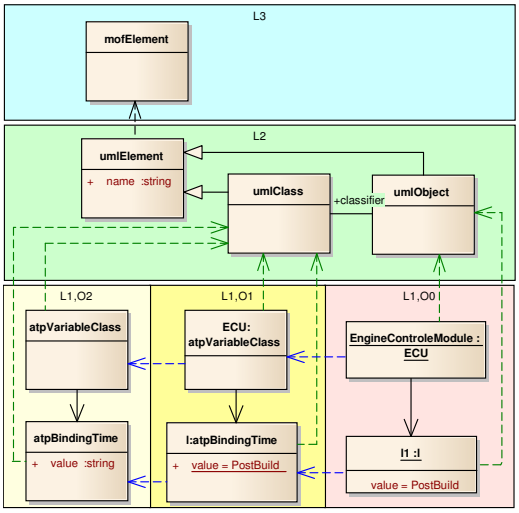

Figure 6: Example of deep instantiation

Figure 6 shows an example of the deep instantiation. *O2* classes (e.g. *atpVariableClass*) are linguistically instantiated from the *umlClass*, just like the domain-specific classes on the *O1* layer (e.g. *ECU*) ontologically instantiating the corresponding *O2* classes. The objects instantiating the *O1* classes reside on the *O0* layer. The ontological instantiation of the *O2* class attributes (e.g. *value* of the *atpBindingTime*) shall be done in a static manner, i.e. all *O0* instances shall have the same value.

4 Discussion

We showed in Section 2.1 that the AUTOSAR meta-modeling environment relies on the traditional UML based meta-modeling extended with the UML stereotypes. The consequences of this approach (Q1) described in Section 2.1 (see [1], [2] and [3]) are solved by the majority of UML based modeling tools available on the market today (e.g. Enterprise Architect used by AUTOSAR).

For solving the problem of dual classification [1], modelers are required to create instances as *Object*s (linguistic instantiation) and then classify them with the correct *Class* (ontological instantiation). The stereotype related problems [2] and [3] can be solved by providing a graphical interface to the modelers for mapping the applicable stereotypes to the actual UML elements (e.g. *atpVariation* is applicable to *Class*es).

Regarding the use of the three proposed approaches instead of stereotypes that assure strictness, the prototypical pattern can be applied in the traditional UML based tools. However it is not feasible for large domain-specific meta-models because a new top-layer meta-class is required for each pair of stereotype-tagged value creating a wide and deep inheritance hierarchy of meta-classes. For example, we need a new top layer class for *atpPostBuildVariationPoint*, *atpPreCompileVariationPoint*, etc.

UML extension and deep instantiation approaches, on the other hand, require special modeling tools. For UML extension, modelers would need to extend the UML meta-model with domain-specific classes and then create their linguistic instances (e.g. *ECU* as an instance of *atpVariableClass* rather than *umlClass*). For deep instantiation, modelers would need to define classes as ontological instances of other classes (e.g. *ECU* as an instance of *atpVariableClass*). The use of OCA in all three approaches improves the correct understanding of the meta-modeling hierarchy.

Despite the fact that there are some modeling tools today supporting these approaches (e.g. Melanee (Atkinson and Gerbig, 2012) for deep meta-modeling), they were not available on the market when the AUTOSAR meta-model was initially developed in 2003. Furthermore, they are still not widely used in the industry. The designers of large software systems are reluctant to accept new approaches and tools until they have a long successful history of application in different domains (Pagel and Brörkens, 2006). This is one of their major drawbacks (Q2).

Additionally, the majority of automotive modeling practitioners are not aware of the advanced meta-modeling concepts, such as deep meta-modeling. For example, even the formal definition of the AUTOSAR profile (as an instance of the UML profile definition) that required deeper analysis of the UML meta-model was considered too complex for the automotive modelers (Pagel and Brörkens, 2006). On the other hand, they are mostly familiar with the use of UML and UML stereotypes. Therefore there is no need for additional training of engineers which is important for companies where hundreds of modelers are responsible for the design of the system.

Finally, the approach used by AUTOSAR based on UML extended with profiles and XML as an exchange format for the AUTOSAR models is a well established approach used for many years. This makes the automotive modeling practitioners reluctant to changes it, unless a new approach can solve some of the practical problems they face today such as (Q3):

- How to estimate the impact of domain-specific meta-model evolution on the modeling tools?
- How to minimize the tooling interoperability issues after adopting a new meta-model release?
- How to assure smooth integration of the models based on different meta-model releases?

To exemplify these problems, consider the following automotive scenario: A new AUTOSAR meta-model release supports 10 new architectural features. 5 ECUs in the system are interested in using 2 of them while other ECUs do not need any, just minor functional updates (e.g. new signals on the bus). This requires that the architectural design tools support multiple partial meta-model releases for different ECUs (or their superset, if possible), which may cause unexpected issues in the ECU configuration tools. Additionally the integration of the ECU models developed based on different meta-model releases may cause bus and/or functional incompatibilities.

It remains unclear if and how the described meta-modeling approaches support solving these practical issues and whether there are significant differences between the approaches related to them. However it is clear that the mentioned issues affect the entire modeling tool-chain which may be distributed among several companies, while the problems related to strictness can be solved by individual modeling tools.

5 Conclusions

In this paper, we recognize that the concept of domain-specific meta-modeling strictness is not the primary concern of the meta-modeling practitioners designing large software systems with distributed implementation, such as automotive systems. However we believe that the approaches assuring strictness, such as deep meta-modeling (deep instantiation combined with OCA), could be employed in the development process of such systems. We also believe that a joint work of researchers and practitioners to rework a part of a real industrial domain-specific meta-modeling environment, such as AUTOSAR, according to these approaches would be needed to evaluate their benefits and drawbacks. This would bridge the current gap between the theoretical meta-modeling discussion and their practical realization.

We also hope that this paper will lead to more research addressing the practical problems described in this paper both in the automotive domain and other domains (e.g. avionics), such as the tooling interoperability issues after the adoption of a new meta-model version (or its subset) in the development projects.

ACKNOWLEDGEMENTS

We want to thank Swedish Governmental Agency for Innovation Systems (VINNOVA), grant no. 2013-02630, and Volvo Cars for funding this research.

REFERENCES

(2003). *Automotive Open System Architecture*. AUTOSAR, www.autosar.org.

(2004). *MOF 2.0 Core Specification*. Object Management Group, www.omg.org.

(2011). *XML Metadata Interchange (XMI) Version 2.2* . Object Management Group, www.omg.org.

(2014). *AUTOSAR Generic Structure Template v4.2.1*. www.autosar.org.

Atkinson, C. (1998). Supporting and Applying the UML Conceptual Framework. In *International Workshop on The Unified Modeling Language*, pages 21–36.

Atkinson, C. and Gerbig, R. (2012). Melanie: Multi-level Modeling and Ontology Engineering Environment. In *Objects, Models, Components, Patterns*, page 7.

Atkinson, C. and Kühne, T. (2001). The Essence of Multilevel Metamodeling. In *International Conference on the UML 2000*, volume 2185, pages 19–33.

Atkinson, C. and Kühne, T. (2002). Rearchitecting the UML Infrastructure. *Transactions on Modeling and Computer Simulation Journal*, 12(4):291–321.

Atkinson, C. and Kuhne, T. (2003). Model-Driven Development: A Metamodeling Foundation. *Journal of IEEE Software*, 20(5):36–41.

Atkinson, C., Kühne, T., and Henderson-Sellers, B. (2002). Stereotypical Encounters of the Third Kind. In *International Conference on The Unified Modeling Language*, pages 100–114.

Durisic, D., Staron, M., Tichy, M., and Hansson, J. (2014). Evolution of Long-Term Industrial Meta-Models - A Case Study of AUTOSAR. In *Euromicro Conference on Software Engineering and Advanced Applications*, pages 141–148.

Gouriet, P. (2010). Involving AUTOSAR Rules for Mechatronic System Design. In *International Conference on Complex Systems Design & Management*, pages 305–316.

Harel, D. and Rumpe, B. (2004). Meaningful modeling; whats the semantics of semantics? *IEEE Computer*, 37(10):64–72.

Lee, J. C. and Han, T. M. (2009). ECU Configuration Framework Based on AUTOSAR ECU Configuration Metamodel. In *International Conference on Convergence and Hybrid Information Technology*, pages 260–263.

Liu, Y., Li, Y. Q., and Zhuang, R. K. (2013). The Application of Automatic Code Generation Technology in the Development of the Automotive Electronics Software. In *International Conference on Mechatronics and Industrial Informatics Conference*, volume 321-324, pages 1574–1577.

Pagel, M. and Brörkens, M. (2006). Definition and Generation of Data Exchange Formats in AUTOSAR. In *European Conference on Model Driven Architecture-Foundations and Applications*, pages 52–65.

Saeki, M. and Kaiya, H. (2007). On Relationships among Models, Meta Models and Ontologies. In *6th OOPSLA Workshop on Domain-Specific Modeling*.