Available online at www.sciencedirect.com

ScienceDirect

IFAC PapersOnline CONFERENCE PAPER ARCHIVE

IFAC PapersOnLine 59-25 (2025) 149–154

Conceptual Profiles in Object-Oriented Metamodeling of Industrial Assets

Marcel Auer * **Michael Riester** ** **Moritz Dorn** *
Alexander Fay *** **Mike Barth** *

* *Karlsruhe Institute of Technology, Germany (e-mail: marcel.auer@kit.edu, moritz.dorn@kit.edu, mike.barth@kit.edu)*
** *Endress+Hauser Digital Solutions, Freiburg, Germany (e-mail: michael.riester@endress.com)*
*** *Ruhr University Bochum, Germany (e-mail: alexander.fay@rub.de)*

Abstract: The distinction between information models of device types and device instances poses a significant challenge in digitally representing industrial assets. This paper analyzes the modeling relationships between types and instances of industrial field devices, e.g. mass flow sensors, using metamodel theory, highlighting conceptual gaps in current modeling methods. By delving into object-oriented metamodeling, the implicit and explicit relationships across varying abstraction layers are clarified, illustrated with examples from the process industry. As a conceptual solution, an extended modeling framework is proposed, separating structural and conceptual-semantic aspects. This approach allows for distinct profiles of type and instance-related information and offers a categorical approach for the categorization of updates of asset information.

Keywords: Meta-Models for Automation Systems; Information Modeling; Asset Administration Shell

1. INTRODUCTION

Within the automation engineering domain of process plants, information models provide an effective mechanism for the seamless exchange of information among the various disciplines involved in the process (Grüner et al., 2024). Standardized information models, such as the Asset Administration Shell (AAS), facilitate concurrent access to the data stored within the AAS repository for component manufacturers, service providers, integrators, and plant operators (Wagner et al., 2017). The Information Model (IM), which functions as a virtual representation of a real asset, aims to enhance transparency and efficiency during plant development, monitoring, and maintenance. IMs are formulated on the foundation of metamodels, which describe the structural design, permissible elements, and their interrelationships. In this modeling context, adopting the object-oriented approach prevalent in engineering, a distinction is made between the types and instances of industrial assets. Therefore, a differentiation is established between Type-IM and Instance-IM within the information exchange during an engineering process (Tauchnitz et al., 2024). However, the definition of type or instance-oriented information lacks sufficient precision and exhibits variations in terms of domain, role, or use case, which complicates the standardization of this procedure.

Field devices for process automation, such as mass flow sensors, are custom configured in limited quantities and manufactured according to specified configurations. This leads to a high degree of complexity, with potentially millions of distinct configurations (field device types) depending on the kind of device and the manufacturer's catalog. Manufacturers of these field devices can provide integrators or plant operators with all relevant information on the characteristics of a particular field device type prior to its production. When this Type-IM is made available, both conceptual and practical challenges emerge, which are elaborated in Sec. 2.

Due to its analogous structure to natural hierarchies, encompassing inheritance and polymorphism, object orientation is a prevalent modeling paradigm within engineering sciences (Maffezzoni et al., 1999). It underpins numerous industrial software solutions for the development and management of technical systems. The classification of devices and system components according to their functionality and properties, such as the type of an ultrasonic flow sensor (which includes configuration parameters such as process connection type, nominal diameter, calibrated measurement range, and connectivity), is standard practice. These field device types can be modeled using templates, analogous to classes in object-oriented software development. The term "template" is intentionally chosen over the term "class" to clearly distinguish these structures since classes in object-oriented programming provide data encapsulation and executable functions. Therefore, "template" maintains both linguistic and conceptual accuracy for our domain-specific modeling approach.

In software development, an instance corresponds to an object created precisely according to the blueprint of its class, with identical methods and attributes that remain unchanged during the program's runtime (Meyer, 1997). However, for industrial field devices, factors such as firmware updates challenge this assumption, since an

2405-8963
Peer review under responsibility of International Federation of Automatic Control.
10.1016/j.ifacol.2025.11.940

instance of a field device may acquire or lose functionality through the update. Such alterations undermine the theoretical uniformity of the field device type and necessitate subsequent adaptation of the Type-IM, affecting all instances of this type that have been manufactured and distributed. Conversely, it is also plausible for an instance of a field device to transition to a different type following modifications to the instance, such as a change of the communications module. Therefore, the modeling of industrial assets must allow the decoupling of instances from their original type.

Against the backdrop of this introductory explanation of the problem, the subsequent sections explore in detail the relationship between Asset-Type and Asset-Instance. Based on the findings, specific proposals are presented to optimize industrial asset modeling. The conclusion delineates categorical approaches to update asset information, illustrated with practical examples.

2. ASSET-TYPE AND ASSET-INSTANCE IN OBJECT-ORIENTED METAMODELING

In recent years, various expert committees, such as the Industrial Digital Twin Association (IDTA) and the German Society for Measurement and Automation Technology (VDI-GMA), have engaged in recurrent discussions regarding the relationship between Asset-Type and Asset-Instance. The interpretation of this relationship significantly impacts modeling, tool design, and the formulation of lifecycle-wide processes when applied to information models, particularly when alterations to the Type-IM are considered. In particular, the discussion surrounding Asset-Type and Asset-Instance bears similarities to the debates on strict metamodelling in software engineering with UML models (Durisic et al., 2016). This alignment is not coincidental, as both discussions revolve around the fundamental question of how to model type instantiation in a manner that balances flexibility and accuracy.

The fundamental question concerns the differentiation of a strict versus a loose relationship between Asset-Type and Asset-Instance. In the context of a strict relationship, a device instance remains bound to its type throughout its entire lifecycle. Modifications to the Type-IM, such as subsequent functional or structural updates, would therefore present conceptual challenges. This rigidity stems from the fundamental principles of object-oriented programming, where altering a class definition at runtime is not possible; it necessitates recompilation and, consequently, re-instantiation of all objects based on that class (Meyer, 1997). However, such a process is impractical in the context of industrial systems and their information models, where continuous operation and minimal disruption are paramount. Re-instantiation of numerous field device instances due to a Type-IM update would be prohibitively expensive and potentially unsafe. Therefore, a flexible approach to managing the relationship between Asset-Type and Asset-Instance is crucial for accommodating evolving system requirements and maintaining operational integrity. Conversely, in a loose relationship, the instance merely references its type, enabling the possibility of adopting a new type through a reference modification.

In practical application, however, the Type-IM is often used as a predefined stereotype, which is copied wholly or partially during the production of a field device instance and augmented with instance-specific data like serial numbers and production dates. From this point on, Type-IM ceases to have further relevance for Instance-IM, as the type-specific information becomes redundantly available. Consequently, updating the type information necessitates replacing the Instance-IM and integrating it with the existing lifecycle information of the Asset-Instance.

The Meta Object Facility (MOF), as specified by the Object Management Group, constitutes a foundational framework for metamodeling, enabling the specification of metamodels across multiple hierarchical layers (Object Management Group, 2016). As will be shown in the following sections, the MOF framework is particularly relevant to the discussion of Asset-Type and Asset-Instance, as it provides a clear and concise way to model the relationships between these two concepts. The MOF framework is characterized as a metadata management framework that includes a minimum of two, but more typically four, hierarchical layers for the specification of metamodels. These hierarchical layers are categorized into the meta-metamodel layer (M3), the metamodel layer (M2), the model layer (M1), and the data or information layer (M0). Figure 1 illustrates the metamodeling framework for industrial assets, such as sensors, spanning the four MOF layers (M0-M3), and details the progression from model abstraction to physical reality. This representation systematically illustrates how abstract modeling concepts are progressively transformed across multiple layers into concrete digital representations of physical field devices. The example presented is based on the AAS-Metamodel, as it facilitates the modeling of both a type and an instance of a field device within the identical framework, thereby providing a clearer explanation of the subject matter.

The M3 layer constitutes the MOF meta-metamodel, which articulates the foundational language and grammatical constructs necessary for formulating metamodels, thereby establishing the abstract foundation for all subordinate layers. The AAS metamodel is located in the metamodel layer (M2), where it delineates the structural framework of the administration shell and articulates its essential components (IDTA, 2025). In the model layer (M1), the application-specific concretization is provided through the AAS template of the modeled field device, in this case a mass flow sensor, which determines the structure for the respective IM. Various submodel templates are incorporated into the AAS template for demonstration purposes. The definition of a specific submodel template is depicted on the right in M1. The information layer (M0) contains serialized information objects that can be exchanged between and interpreted by engineering partners. Therefore, the administration shells, instantiated from the AAS template (Asset-Type AAS and Asset-Instance AAS), are located in M0.

The relationships denoted by solid red arrows as "instanceOf" signify instantiations within the context of strict metamodeling. Each object instantiated in this manner possesses a class (meta-object) at the M(n+1) layer, which defines the object's structure. It is important to note that both the Asset-Type AAS and the Asset-Instance AAS

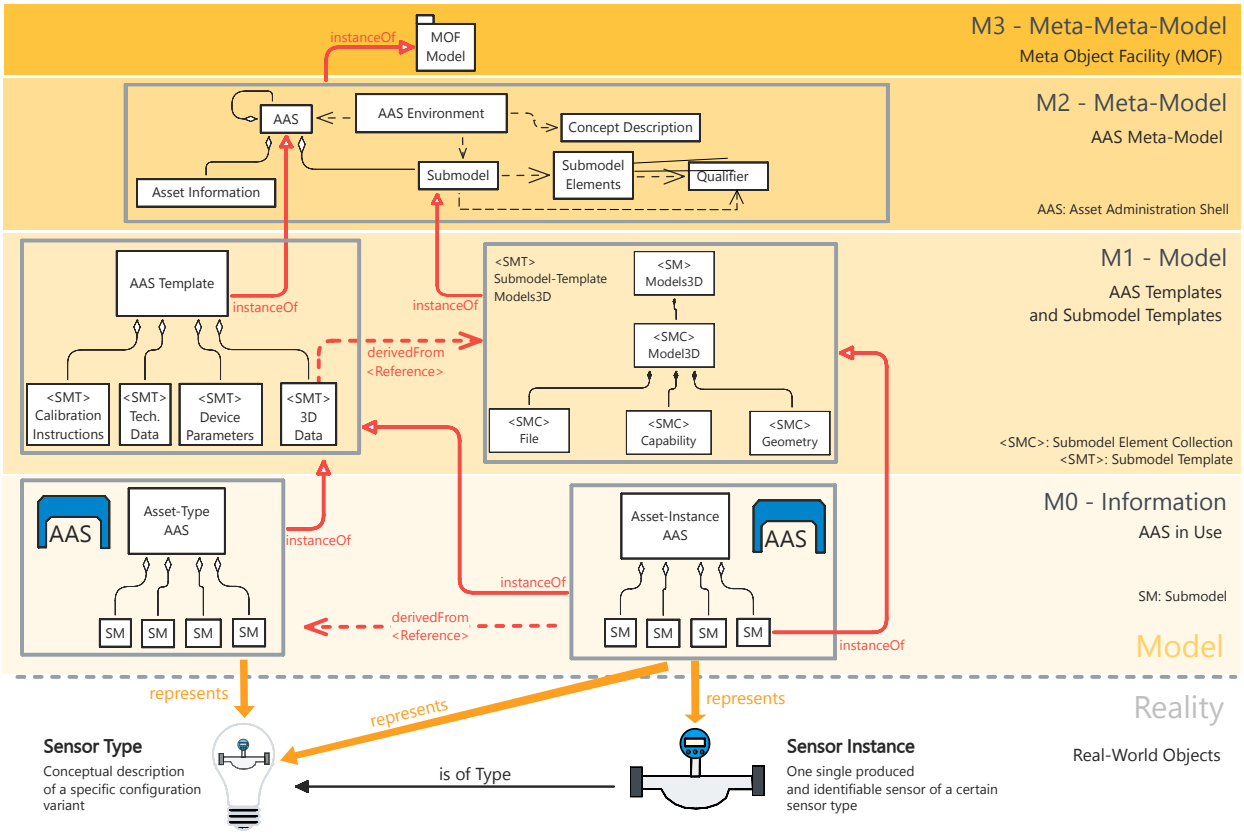

Fig. 1. Metamodel classification for Asset-Types and Asset-Instances exemplified through the Asset Administration Shell metamodel

were instantiated from their respective AAS-Template at the M1 layer. Consequently, there is no instantiation in the sense of object orientation between these two Asset Administration Shells; instead, as defined in the AAS Metamodel, a “derivedFrom” reference exists. This reference clarifies that an Instance-AAS is conceptually derived from the corresponding Type-AAS, though it is not a direct instantiation from a metamodeling viewpoint. The aforementioned copying and extension of the Asset-Type AAS to the Asset-Instance AAS is facilitated through modeling, by defining all instance-related information within the AAS template as optional. This information can be incorporated later within predefined structures. Therefore, it would be methodologically incorrect to equate predefined stereotypes field device type representation with a modeling meta-layer (Atkinson and Kühne, 2000). This distinction is universally essential for the comprehension of the model hierarchy and extends beyond the context of the Asset Administration Shell, applying to all modeling implementations in which the instantiation process itself is subject to modeling.

Reality itself is excluded from the model and consequently depicted beneath the four modeling layers. The conceptual-semantic understanding of a sensor type is illustrated on the left, while its tangible sensor instance is depicted on the right. The conceptual description of a field device represents the ontological type of the field device instance (Atkinson and Kühne, 2001). This terminology implies that the conceptual-semantic dependency introduces an additional dimension to metamodeling. While it may be posited that this dependency is implicitly included within the metamodel through references such as ’derivedFrom’, it is imperative, for maintaining consistency in the handling of Asset-Types and Asset-Instances, that these relationships are defined explicitly. An approach to explicit modeling of conceptual-semantic dependencies will be presented in the next section.

3. CONCEPTUAL PROFILES FOR ASSET-TYPE AND ASSET-INSTANCE

In Section 2, a fundamental conceptual inconsistency in the modeling of relationships between Asset-Types and Asset-Instances was identified when predefined stereotypes are equated with a modeling meta-layer. Comparable issues have been observed in the development of object-oriented software using UML models, and the implementation of strict modeling profiles was suggested as a potential solution (Atkinson and Kühne, 2000). Strict profiles are defined as a set of predefined modeling elements that cut across the levels in the four-layer meta-architecture. Based on these strict UML modeling profiles in software development, we introduce the term of conceptual modeling profiles for Asset-Types and Asset-Instances to refine the metamodeling of industrial assets. Unlike strict UML profiles, conceptual profiles do not represent a structural predefined or abstracted model, but rather take on a semantic and contextual descriptive role. Figure 2 schematically represents the separation of modeling dimensions into structural and conceptual-semantic descriptions.

The MOF layers (M0 and M1) define the structure of information models, arranged vertically, while the conceptual-

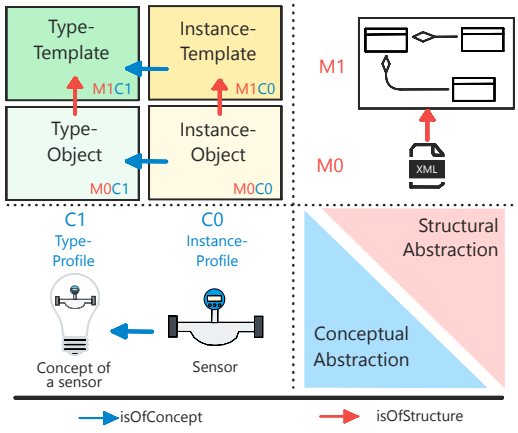

Fig. 2. Orthogonal relations in structural and conceptual-semantic descriptions of reality for the development of information models

semantic profile hierarchy extends horizontally. Objects in the and Type-Profiles are located at the M0 layer, representing fully-specified data objects exchangeable among engineering partners, depicted as XML files. Instance-Profile (C0) and Type-Profile (C1) templates at the M1 model layer define M0 object structures without detailing their values. These templates incorporate a composition of proprietary or standardized submodel templates. Thus, in M1, unique templates for Asset-Types and Asset-Instances are defined, differing in submodel quantity and structure.

Similarly to the MOF structure, there are no predefined constraints on the number of conceptual profiles that can exist within a given layer. However, higher abstract layers and higher-order profiles (M2+ and C2+) are excluded from further consideration in this article, as they remain unaffected by modifications from engineering partners.

In the context of the mass flow sensor example, the Type-Profile (C1) encapsulates the static properties that the manufacturer determines and thus delineates the static conceptual attributes of the device. The systematic separation into Type-Profile (C1) and Instance-Profile (C0) facilitates the integration and updating of asset-related information. This partitioning allows for the elimination of redundancies in the object information at the M0 layer and enables precise modifications without inducing unintended side effects.

In Fig. 3, the illustration of modeling Asset-Types and Asset-Instances discussed in Section 2 is expanded to incorporate the notion of strict conceptual-semantic profiles. Within the M1 layer, it is noted that the templates for Asset-Type AAS and Asset-Instance AAS differ in the submodels they include, as they intrinsically represent distinct entities. This allows type- and instance-related information to be mapped without redundancy and linked logically. As depicted in the figure, 3D data and dimensions are associated with the field device type, thereby obviating the need to model these attributes for each field device instance individually. In contrast, location information concerning the precise installation site within the plant is entirely independent of field device type data within the modeling framework. The new “derivedFromType” reference therefore does not describe a stereotypical predefined set of attributes, but defines the elements of the instance profile as a conceptual derivation of the type profile elements.

To illustrate how conceptual profiles can be applied in practice, their application will be explained using excerpts from the submodel templates “Device Type Parameters” within the Type-Profile (C1) and “Parameter Config” within the Instance-Profile (C0), illustrated in Fig. 3. Within the submodel template of the Type-Profile, pertinent parameters can be modeled with respect to their intended meanings and constraints, such as specified value intervals or options delineated by the field device type. Conversely, the submodel template within the Instance-Profile allows for the modeling of the specific parameter values set on the installed field device instance. Excerpts of the corresponding Type-Object and Instance-Object are shown in Fig. 4. The established value (250 cd/m²) is encompassed within the Instance-Object (on the right) and makes reference to the interval (50-500 cd/m²) of permissible values as delineated in the Type-Object (on the left) as a qualifier. All changes to the display brightness set on the device are therefore limited to the Instance-Object, but can be checked for conformity using the Type-Object provided by the manufacturer.

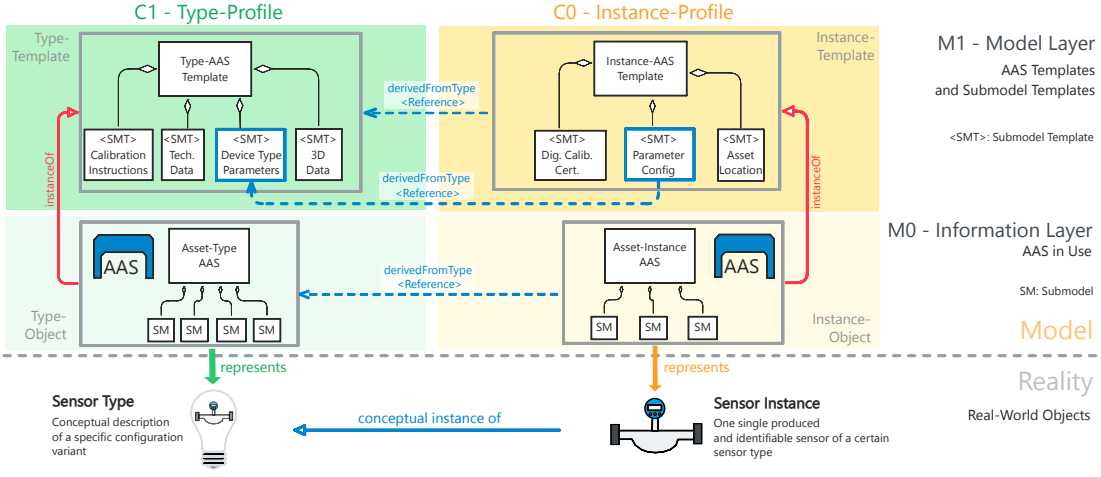

Fig. 3. Distinct conceptual profiles for both Asset-Type and Asset-Instance within the context of the Asset Administration Shell

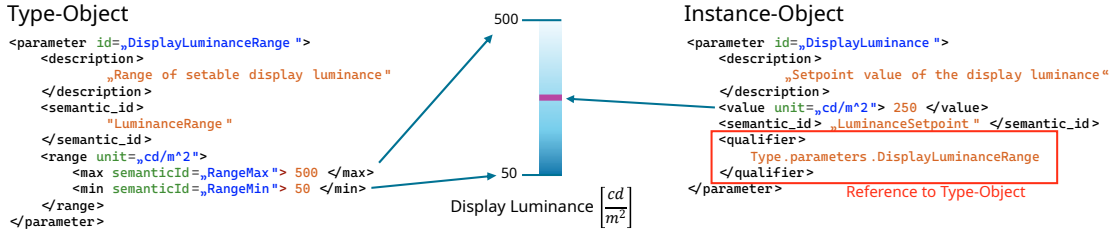

Fig. 4. Modeling of a device parameter for display brightness adjustment. Left: Representation of the adjustable value range in the Type-Object; right: Depiction of the device-set value in the Instance-Object.

4. UPDATES OF ASSET INFORMATION

The previous section outlined an approach to separate Asset-Type modeling from Asset-Instances and make implicit relationships explicit, facilitating asset information updates. It is important to note that updates to the information models are deliberate actions by the operator, who receives updates from the manufacturer or service provider beforehand. The plant operator maintains their own system for administrating and maintaining the relevant information models. Updates at the model layer (M1) or Type-Profile (C1), beyond just changes in the Instance-Object (M0C0) status, require engineering decisions within the plant. Automatic or direct model updates are intentionally avoided to prevent manufacturers from altering a certified plant's status via updates.

This approach provides the necessary flexibility to adapt to changing asset characteristics while maintaining operator control over their information system. Explicitly modeling dependencies among conceptual profiles enhances the transparency and traceability of external dependencies. This feature supports more precise change tracking and clearer classification. The resulting information models are used to manage process-critical devices and to contextualize and orchestrate the flow of information within production. Therefore, precise change tracking is vital in industrial settings with certified plants and regulatory requirements, as it aids in assessing the propagation effects of a proposed modification. Examining the unidirectional dependencies, depicted in Fig. 2, reveals that the possible propagation of a modification of the asset model depends on the highest modeling layer and the highest-order profile affected. The resulting update categories, shown in Fig. 5, are further explained in the following through practical examples. The update categories are categorized in ascending order of their potential propagation effects from a metamodeling perspective:

I. Instance-Object Update (M0C0): Modification of instance-based information at the value level. This scenario arises when a field device instance undergoes re-parameterization or when a new calibration certificate is stored or referenced (version 1.2 within the Instance-Profile). Before applying the change, the conformity with the structural requirements of the Instance-Template and the conceptual requirements of the Type-Object must be checked.

II. Instance-Template Update (M1C0): Alterations to the structure of instance modeling can lead to structural impacts on the resulting Instance-Object. In the event that a purely instance-related submodel template is altered (for example, by introducing a new iteration of a standardized submodel template), the relative version of the Instance-Profile escalates to 2.1. Consistent with the earlier scenario, a correspondingly modified Instance-Object is subsequently generated (version 2.2). Given the unidirectional dependency existing between the Instance-Profile and the Type-Profile, alterations in the former do not exert any influence on the latter.

III. Type-Object Update (M0C1): Modifications to type-related information at the value level may impart conceptual implications on the value conformity of the Instance-Object. For example, should the manufacturer curtail the maximum permissible parameter value for a particular device type through a firmware update (Type-Profile version 1.2), e.g. the maximum speed setpoint of a pump, a corresponding adjustment of the chosen setpoint in the Instance-Object (Instance-Profile version 1.2) may become requisite. It should be emphasized at this point that the categories enumerated herein represent an estimate of the potential propagation of modification. Therefore, it remains possible that the parameter value encapsulated within the Instance-Object continues to align with the specifications of the Type-Object, rendering alterations to the Instance-Object unnecessary.

IV. Type-Template Update (M1C1): Alterations to the Type-Template can induce structural effects on the generated Type-Object, conceptual effects on the Instance-Template, as well as structural and conceptual effects on the Instance-Object. For example, should a firmware update introduce a novel parameterizable functionality for an field device type, it is necessary to augment the submodel template for type-related parameters with these new attributes in the Type-Template (resulting in an increase of the Type-Profile's relative version from 1.1 to 2.1). Due to the conceptual dependency of the Instance-Template on the Type-Template, it is necessary to update the model structure of the Instance-Template (Instance-Profile version 2.1). Consequently, both Type-Object and Instance-Object are generated with the corresponding values for the new modeled functionality (attaining version 2.2 in Type-Profile and Instance-Profile).

This initial examination of the effects of modifications on the modeling of industrial assets is not exhaustive and serves as a preliminary assessment. The dependencies facilitate only an approximation of the maximum propagating impact to which they can induce or potentially lead. A differentiation between various discrete scenarios is possible,

depending on the nature of the modified information and the relevant category. The aim of systematic categorization of updates is to enable practical implementation strategies for industrial asset management systems. Plant operators shall be able to establish automated conformity checking between Type-Objects and Instance-Objects, while manufacturers provide structured update packages that clearly indicate propagation requirements.

Regarding the specified version numbers, illustrated in Fig. 5, it should be noted that these versions are not aligned with the semantic versioning standards typically applied in software development (Preston-Werner, 2013). Rather, the leading numeral conveys the relative version status of the related profile template, and the following numeral indicates the relative version status of the profile object.

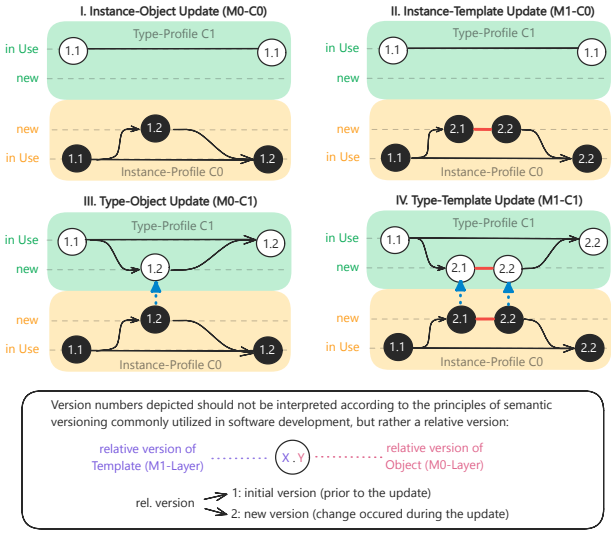

Fig. 5. Updates to the modeling of field devices categorized according to their possible propagation effects

5. CONCLUSION AND OUTLOOK

This paper highlighted conceptual and practical challenges encountered in the modeling of Asset-Types and Asset-Instances within the framework of industrial information models. The findings underscore the necessity for a distinct differentiation between structural modeling and conceptual-semantic modeling. This was achieved by the introduction of distinct conceptual-semantic profiles for Asset-Types and Asset-Instances, which expand the metamodeling layers of the MOF framework. The relationship between Type-IM and Instance-IM, which is implicitly embedded in the layers of the MOF framework, can be explicitly modeled with this approach. A notable limitation of the proposed approach is the increased initial complexity inherent in the introduction and implementation of the prescribed modeling framework. However, this complexity is mitigated by the enhanced semantic clarity and thus improving the update process. The proposed framework establishes a foundational basis for the systematic categorization of modifications to information models according to their potential propagation effects. It provides a structured approach for the consistent modeling of industrial assets, such as field devices, throughout their entire lifecycle.

As articulated in Sec. 4, a thorough assessment of propagation effects stemming from model modifications necessitates an expansion in the classification of modifications within both Type-Profile and Instance-Profile. Future research will therefore focus on assessing techniques for automated consistency checks between Type-Profiles and Instance-Profiles during updates, utilizing practical applications of the metamodeling framework presented herein. Additionally, the aim is to investigate the mapping of updates to both Type-IM and Instance-IM in the context of static versioning, thus improving change traceability and allowing complete or selective rollback to previous versions.

REFERENCES

Atkinson, C. and Kühne, T. (2000). Strict Profiles: Why and How. In A. Evans, S. Kent, and B. Selic (eds.), *≪UML≫ 2000 — The Unified Modeling Language*, 309–322. Springer, Berlin, Heidelberg. doi:10.1007/3-540-40011-7_22.

Atkinson, C. and Kühne, T. (2001). The Essence of Multilevel Metamodeling. In M. Gogolla and C. Kobryn (eds.), *≪UML≫ 2001 — The Unified Modeling Language. Modeling Languages, Concepts, and Tools*, 19–33. Springer, Berlin, Heidelberg.

Durisic, D., Staron, M., Tichy, M., and Hansson, J. (2016). Addressing the need for strict meta-modeling in practice - a case study of AUTOSAR. In *2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD)*, 317–322.

Grüner, S., Eskandani, N., Stark, K., Hoernicke, M., and Braun, R. (2024). Asset-Administration-Shell-Based Continuous Engineering in Energy Industries. In *2024 IEEE 29th International Conference on Emerging Technologies and Factory Automation (ETFA)*, 01–08. doi:10.1109/ETFA61755.2024.10711114.

IDTA (2025). Specification of the Asset Adminstration Shell - Part 1: Metamodel v3.1, Industrial Digital Twin Association.

Maffezzoni, C., Ferrarini, L., and Carpanzano, E. (1999). Object-oriented models for advanced automation engineering. *Control Engineering Practice*, 7(8), 957–968. doi:10.1016/S0967-0661(99)00074-X.

Meyer, B. (1997). *Object-Oriented Software Construction*. Prentice Hall PTR, Upper Saddle River, NJ, 2nd edition.

Object Management Group (2016). Meta Object Facility - Version 2.5.1.

Preston-Werner, T. (2013). Semantic Versioning 2.0.0. https://semver.org/.

Tauchnitz, T., Maurer, F., Barth, M., Schüller, A., and Drath, R. (2024). Engineering in der Prozessindustrie mit der Verwaltungsschale: Teil 1: Zukunftsbild und Konzeption eines durchgehenden Informationsflusses am Beispiel des Engineerings von PLT-Einrichtungen. *atp magazin*, 66(6-7), 88–99. doi:10.17560/atp.v66i6-7.2734.

Wagner, C., Grothoff, J., Epple, U., Drath, R., Malakuti, S., Grüner, S., Hoffmeister, M., and Zimermann, P. (2017). The role of the Industry 4.0 asset administration shell and the digital twin during the life cycle of a plant. In *2017 22nd IEEE International Conference on Emerging Technologies and Factory Automation (ETFA)*, 1–8. doi:10.1109/ETFA.2017.8247583.