Making a Case for Multi-level Reference Modeling – A Comparison of Conventional and Multi-level Language Architectures for Reference Modeling Challenges

Sybren de Kinderen
Universität Duisburg-Essen

Monika Kaczmarek-Heß
Universität Duisburg-Essen

Making a Case for Multi-level Reference Modeling – a Comparison of Conventional and Multi-level Language Architectures for Reference Modeling Challenges

Sybren de Kinderen and Monika Kaczmarek-Heß

Research Group Information Systems and Enterprise Modelling,
Institute for Computer Science and Business Information Systems (ICB),
Faculty of Business Administration and Economics,
University of Duisburg-Essen, Essen, Germany
{sybren.dekinderen,monika.kaczmarek-hess}@uni-due.de

Abstract. As a continuation of our earlier work, in this paper we focus on the suitability of multi-level modeling for the creation and use of reference models. Specifically, we first discuss known challenges of reference modeling. Then, using the UML (for conventional meta modeling) and the FMML^x (for multi-level modeling) as language architectures of choice, we show how conventional meta modeling contributes to challenges of reference modeling, and how the added flexibility and expressiveness of multi-level modeling can address these. We use an excerpt of NISTIR 7628, a well-established reference model for smart grid cyber security, as an illustrative scenario.

Keywords: reference modeling, multi-level modeling, comparison

1 Introduction

Reference models, being conceptual models, abstract away from one specific organization, and instead focus on characteristics common to many organizations, within or across one or more industries and/or application domains [1], [2]. Reference models are created to provide so-called best practices for particular domains/scenarios [3], and as such hold several promises, e.g., (1) fostering reuse, cf. [1], [4], meaning that instead developing models from scratch, one can capitalize on already encoded expertise, and (2) fostering a shared domain understanding, cf. [1], [5], by providing a common semantic reference system [1] to stakeholders.

Those promises express themselves in reference modeling still being a topic of active research, e.g., [6-9], and in particular, in the design and adoption of reference models for various domains, cf. [1]. A relatively recent example of such a domain is the electricity sector, where reference models such as NISTIR 7628 have been proposed, being a logical reference model for smart grid cyber security [10], [11]. Nevertheless, there are several challenges associated with, both, the design and use of a reference model. As we discuss in Section 2.1, these challenges include finding a balance between *generality* and *specificity*, supporting *variability*, and consistent *adaptation* of a

16th International Conference on Wirtschaftsinformatik,
March 2021, Essen, Germany

reference model. As we discuss in our earlier work [12], while these challenges have been already reported a while ago, they still play a pertinent role in recent reference models, like the mentioned NISTIR 7628.

In this paper, we argue that the mentioned reference modeling challenges are partially related to the characteristics of the modeling languages used to create and disseminate reference models. Especially, we show how these challenges arise for reference models that rely on conventional meta modeling (next to the afore- mentioned NISTIR 7628, these include, e.g., UML-CI for critical infrastructure modeling [13], and E-MEMO for e-commerce scenarios [14]). Conventional meta modeling, of which UML class diagrams [15] are a prominent exemplar, has not been natively designed with classification levels in mind [16], and (in keeping with its basis in object-orientation) maintains a strict separation between types and instances. As such, as we show in Section 3.2, conventional meta modeling does not *naturally* lends itself well to expressing domain hierarchies with different levels of classification, while this is very much of importance to reference modeling. While mechanisms such as generalization/specialization, and specific to the UML, redefinition and default values, can be partly used, still redundancies and inconsistencies of reference models remain an issue. Also, in keeping with [16], using conventional meta modeling leads to accidental complexity of reference models, in the sense that their complexity increases not due to the complexity of the domain one is modeling, but rather due to the underlying language (architecture) that is being used. Especially, this expresses itself in the use of multiple abstraction mechanisms where one should suffice, like with the use of generalization/specialization within the abstraction level meant for language specification (and so, one is "overloading the level" [16]), or the use of redefinition plus default values, where in principle instantiation can suffice.

Additionally, we argue that the application of a relatively novel language architecture, namely a multi-level language architecture, contributes to addressing these challenges. Multi-level modeling is an emerging trend that accounts for multiple, i.e., more than one, levels of classification within one single body of model content [17], [18]. As we explain in more detail in Section 4.1, a multi-level modeling language architecture offers expressiveness and flexibility that naturally fit with the idea of reference models, by capitalizing on mechanisms such as a relaxed type/instance dichotomy, or deferred instantiation [18].

As such, the purpose of this paper is to make a case for multi-level reference modeling, with a focus on comparing a reference model as created with conventional meta modeling, with the same reference model as created through multi-level modeling. To this end we compare a reference model as created with conventional meta modeling (using UML), with the same reference model created using a multi-level language architecture (using Flexible Meta Modeling and Execution Language (FMML^x) [18]). We perform this comparison in the light of a set of well-established reference model challenges, as reported in [12]. As a running scenario, we use an excerpt of a smart cyber security reference model by [10], [11], and, building on our previous work, the multi-level cyber security reference model.

As already mentioned, this paper is a continuation of earlier work. In [12], we discussed typical challenges regarding the creation and use of reference models, and

showed how multi-level modeling, as a language architecture, can help address these. However, for the presented challenges a systematic comparison of multi-level modeling to conventional meta-modeling is missing. This leads to unresolved issues like the possibility of using subtyping, or power types, to address reference model challenges while using conventional meta modeling. The paper at hand is meant to address this gap.

The paper is structured as follows. In Section 2 we provide a background on reference models, recap reference model challenges from earlier work, and introduce the smart grid cyber security reference model that is used for illustration purposes for the remainder of the paper. In Section 3 we subsequently discuss the extent to which conventional meta modeling can address reference modeling challenges and discuss its limitations. Subsequently, in Section 4, we introduce multi-level modeling, and show how it can be used to overcome the limitations of conventional meta modeling. Section 5 concludes with the final remarks.

2 Reference Modeling

2.1 Reference Models and Challenges in Their Creation and Usage

Although a common definition of a reference model has not been established yet, cf. [3], [6-8], it is usually understood as a special type of an information model. From the variety of reference model definitions, cf. [1], [5], [19], [20], in this paper we adopt the definition from Thomas [19, p. 1]: "[r]eference models are reusable representations of abstract know-how for a given application domain". This definition emphasizes (i) abstraction of a reference model, in the sense of moving beyond one particular application context [1], as well as (ii) a reference model targeting a class of problems in a given domain [2].

To create a reference model a modeling language is used, which provides a set of constructs and rules that dictate how modeling concepts can be combined. Here, typically traditional modeling languages such as Entity Relationship Model (ERM) [21], Unified Modeling Language (UML) [15], Event-Driven Process Chain (EPC) [22], or Business Process Model and Notation (BPMN) [23], cf. [20], are either directly used, or extended with additional constructs to increase their expressiveness [24], [25].

Reference models come with a variety of prospective uses [6-9]. By capitalizing upon the domain knowledge encoded in the reference model, reference models may serve as a blueprint, e.g., for designing an information system [26], or for business process management [5]. As such, one avoids the resource-intensive task of designing a domain model from scratch. Thus, reference models are seen to promote knowledge sharing, communication, and reference implementations [27].

While these are attractive prospects, as we point out in our earlier work [12], reference modeling comes with a set of challenges, which limit its full potential. In the following, we summarize a subset of challenges as relevant in the light of this paper.

Challenge 1: Expressing both general and specific domain information, i.e., addressing a conflict between reuse and productivity.

To ensure coverage of a class of problems, and thus, be applicable beyond a specific context/organization, a reference model should offer general concepts [26], [28]. At the same time, to meet the goals of a particular setting (e.g., for implementation purposes), and thus, to increase model productivity, a reference model should also provide specific concepts [28]. Therefore, a reference model needs to be detailed enough to be usable for an organization [28]. Unfortunately, current modeling languages provide only a limited set of mechanisms for expressing both generic and specific concepts [18], [24]. Especially for this paper, this holds for reference models based upon the UML [18]. As we detail in Section 3.2, while mechanisms like generalization/specialization can be used, they offer only a limited means for expressing both the generic and specific within a reference model.

Challenge 2: Expressing variability while avoiding redundancy, i.e., providing flexibility to users of reference models.

Partially overlapping with the call for specific concepts, reference models should account for variability. This means that a reference model should provide coverage of a range of specific requirements/constraints [26], e.g., to adapt a reference model to the processes of a specific industry, as done, e.g., in [24]. Thus, it is required to distinguish between those parts of the system that are invariant within the group of intended users, and other parts that may need individual adaptation. At the same time, redundancy in a reference model should typically be avoided (see, e.g., [24], who in their configurable reference modeling approach speak of "mutually exclusive alternatives"). Unfortunately, only a limited set of mechanisms is provided that can deal with both variability and redundancy. These mechanisms either extend an existing modeling language (like EPCs, as done in [24]), or, as we show specifically in section 3.2, rely on mechanisms like the mentioned generalization/specialization, or instantiation [20].

Challenge 3: Supporting adaptation of reference models, while ensuring compliance and integrity of the system.

The reference modeling language as well as resulting reference models need to offer flexibility. By this we mean that reference model adaptation and extension should be possible, since reference models cannot contain all individual requirements of all potential users [28] (cf. also Challenge 2). While adapting a reference model to the needs of a specific organization (e.g., for implementation purposes), and vice versa, when adapting a reference model based upon its specific application, one should ensure consistency of the adaptation to the reference model. In the simplest case, this implies copying a reference model and adjusting it to the context at hand. However, in that case redundancies may arise, and potential inconsistencies as well [2]. One can envision adding extensions to the reference model, like in [29], but this can be cumbersome and importantly: such extensions are often designed for a one-way adaptation only. For instance, [29] is designed to ensure that an organization-specific model complies with the reference model, but it is not designed to check adaptations of the reference model itself.

2.2 A Reference Model for Smart Grid Cyber Security

The NIST reference model for cyber security, as encoded in NISTIR 7628 [10], offers concepts, cyber security requirements, and guidelines specific to the electricity sector. It follows that the NISTIR 7628 elements are specific for the energy sector in terms of, e.g., considered actors, and types of IT infrastructure. For example, it distinguishes between different equipment types like a smart meter or a customer gateway (also referred to as a home area network gateway in NISTIR 7628 [10, p. 18]). A customer gateway, being relevant for our running example in Sections 3.2 and 4.2, is an (embedded) piece of equipment on the customer side, which acts as a communication interface towards other parts of the smart grid (like the service provider), and which can take care of computationally intensive tasks, like encrypting sensitive metering data prior to transmission.

The NISTIR 7628 has been widely touted for providing guidance on cyber security concerns in smart grid projects, cf. e.g., [30-33], but its adoption and maintenance is partially hampered by the above-mentioned challenges. In particular, [32] points out a lack of systematicy in relating the generic security requirements and guidelines to the concerns of specific smart grid projects, stating that this relation has to be established in an ad-hoc manner. While by no means we want to claim that these challenges are fully due to an underlying language architecture, further in the paper we explain why a language architecture based on conventional meta modeling does not provide a satisfactory solution, and we illustrate the potential that multi-level modeling has in addressing them. It is important to note that at the core of this paper stands a comparison between conventional meta modeling and multi-level modeling. As such, we use the NISTIR 7628 reference model only in as far as it illustrates this comparison, for which a relatively small subset of the larger model, presented in [12], suffices.

3 Challenges of Reference Modeling with Conventional Meta Modeling

3.1 Conventional Meta Modeling

As stated in Section 2.1, different languages, and potentially their accompanying language architectures, can underlie a reference model. In this paper, we focus on conventional meta modeling. We do so since, for reference models emphasizing a static perspective on an organizational action system (as opposed to a dynamic perspective, as done in, e.g., [5], [9], [25]), conventional meta modeling is often an underlying language (architecture) of choice, as among others visible in (i) a reference architecture for NISTIR 7628 [33], (ii) UML-CI, a reference model for critical infrastructure modeling [13], or (iii) E-MEMO [14], a family of reference models for e-commerce development.

Conventional meta modeling refers to language architectures that are based on the Meta Object Facility (MOF [34]). In MOF, one defines the abstract syntax of a language in terms of a meta model on the M_2 level, in terms of defining the key concepts of a

language, their attributes and relations. Subsequently, this meta model can be instantiated into models, which reside on the M_1 level. In line with these two classification levels conventional meta modeling is also referred to as two-level meta modeling [35].

Conventional meta modeling exhibits a fundamental distinction between meta model elements residing on the M_2 level and model elements residing on the M_1 level. Instantiation is the only allowed, one-way, relation between these two levels, to instantiate a model element from a meta model element (but not vice versa). This distinction, also referred to as a type-instance dichotomy [16] is inherited from the object-oriented paradigm underlying conventional meta modeling, which makes a strict separation between classes and objects [16], [17].

As a result of the type-instance dichotomy classes cannot have a state. This is because they reside on the M_2 level, and thus, serve as language specification. Furthermore, the type-instance dichotomy leads to a separation between language specification and language application. Finally, in conventional meta modeling instantiation is only possible to directly proceeding classification levels, also referred to as "shallow instantiation" [17].

As we detail in Section 3.2, the above inherent characteristics of conventional meta modeling, i.e., classes not having a state, a separation between language specification and language application, and shallow instantiation, have a considerable impact on the creation and use of reference models.

3.2 Challenges with Conventional Meta Modeling

To showcase the challenges which arise from employing conventional meta modeling, in the following we focus on the UML, being standardized and widely used. In addition, UML is often the language of choice for contrasting conventional meta modeling with multi-level modeling, cf. e.g., [16], [17], hence it makes sense to proceed in a similar spirit for reference modeling challenges specifically.

Challenge 1: Expressing both general and specific domain information. *Rationale:* As stated in Section 2.1, we should be able to express both generic and specific domain concepts, while expressing domain information as soon as it becomes known, in order to avoid redundancy. When employing UML, we can partly deal with this challenge through a combined use of generalization/specialization, redefinition, and default values. Especially, generalization/specialization allows us to create abstraction hierarchies of concepts, whereas redefinition in combination with default values partially allows us to incorporate information in the reference model, as soon as it becomes known.

However, this would address the challenge only partially. In particular, redefinition in UML allows one to modify a data type and default value, while ensuring that the redefined element "[...] shall be consistent with the RedefinableElement it redefines" [15, p. 100]. However, while UML tracks the exact element being redefined (through a "redefinitionContext" [15, p. 100]) what exactly "consistency" entails here, and what kinds of specific checks are necessary, remains ambiguous. This has resulted in calls for clear definitions of redefinition (e.g., [36], and a recently reported open issue for

UML 2.5.1[1]), and calls for extensions, in the form of additional well-formedness rules, which enforce a consistent redefinition, cf. e.g., [37]. As a result, the inconsistent redefinition mechanism from UML may allow for violating monotonic model extensions, in the sense of catering for inconsistencies of specialized classes, which redefine attributes/association ends of their superclass. Finally, default values allow for assigning values to attributes as soon as they become known. However, this assignment happens on the type level, i.e., separate from the running data of the organization. So, any updates/modifications as it pertains to attribute values from the running organization would have to be separately mirrored in the default values. Finally, since within the UML one is creating the reference model on the M_2 level, and one uses the abstraction mechanism generalization/specialization at the same time, one is in principle using two abstraction mechanisms where one should suffice (in [16], this is also referred to as "overloading the level").

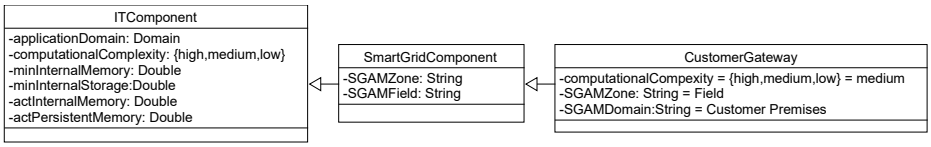

Figure 1 An excerpt from the NISTIR cyber security reference model, reconstructed with the UML

Scenario: For our scenario, we focus on an excerpt of NISTIR 7628 dealing with smart grid components. Specifically, in Figure 1 we see how generalization/specialization allows us express both general and specific concepts, starting with a general "ITComponent" whose attributes are inherited and specialized to the class "SmartGridComponent", and finally into the class "CustomerGateway". Equally, we can see how redefinition, combined with default values, allows us to assign values to attributes from the class "CustomerGateway", such as "computationalComplexity" being assigned the default value "medium", and the two SGAM-related attributes equally being assigned relevant values.

Nevertheless, for the same scenario we can also observe limitations that conventional meta modeling imposes, when it comes to expressing the general and specific at the same time. Firstly, since default values exist on the type level, an update, like the "computationalComplexity" of a "CustomerGateway" being changed to "high", which can be a reflection of a change in a class of technologies, needs to be made manually. Secondly, since (a) UML does not maintain a clear hierarchy of semantic richness among its primitive types (e.g., a Boolean type having less permissible instantiations than a type String), and (b) the question of what type of consistency should be kept remains at least partly open, one can in principle envision redefining the types of the attributes "minInternalMemory"and "minStorage" from the

[1] see https://issues.omg.org/issues/spec/UML/2.5#issue-47019. The issue has been reported on 21-7-2020. This open issue provides a minor indication that redefinition is still not well defined.

class “ITComponent” to “String” for its subclass “SmartGridComponent”. However, for the sake of maintaining monotonic model extensions, this is not desired.

Challenge 2: Expressing variability while avoiding redundancy. *Rationale:* UML can partially ensure variability of a reference model, so that on the type level it can be “configured” according to the needs of a (class of) scenarios. Prominently, as with Challenge 1, the combination of generalization/specialization, redefinition and default values allows us to express domain information on a level of abstraction suitable for a range of application scenarios.

However, in line with core notions of conventional meta modeling, UML only allows for instantiation to the directly proceeding classification level. As a result of this, we cannot constrain on a high level of classification at what exact proceeding level of classification domain information should be added. This in turn limits the ability to account for variability.

Scenario: Consider again Figure 1. Here variability and the avoidance of redundancy is partially supported by using generalization/specialization, e.g., to express for a range of scenarios a generic class “ITComponent” with attributes such as “computationalComplexity : high, medium, low”, “minInternalMemory: Double”, and “actInternalMemory : Double”. However, importantly, we are not able to express *when* these attributes should be assigned a value, since in UML – like in conventional meta modeling – *abstraction level is not a first class* citizen. As such, when to assign values to attributes (and equally: when to specify association ends) is arbitrary in the UML. For example, using the UML in our scenario we cannot distinguish between when to assign a value for “minInternalMemory : Double”, which for NISTIR 7628 is important for a *type* of smart grid component (e.g., a “CustomerGateway”), and when to assign a value for “actInternalMemory : Double”, which is important for a specific smart grid component (e.g., “CustomerGateway9876”).

Challenge 3: Supporting adaptation of reference models, while ensuring compliance. *Rationale:* As stated, it is desired that a reference model can be adapted, both in the sense of adaptation to a specific context, but also so that context-specific adaptations can become part of the reference model.

In the UML, one can adapt a reference model as follows. First, one can simply copy-paste the reference model and adapt it for the situation at hand, but as stated in Section 2.1especially in the absence of added consistency checks, like in [29], this can be error-prone and can lead to inconsistencies. Note that the underspecified notion of redefinition, mentioned under Challenge 1, is also relevant here, since any adaptations that are made through redefinition may violate monotonicity.

Second, one can instantiate the reference model, and, with the use of constraints (as typically expressed in OCL [38]), one can check the well-formedness of any extensions. Yet, in that case, one has to essentially “duplicate” the reference model, leading to redundancies. Finally, power types are a candidate for model adaptation. A power type can be defined as a model pattern whereby the instances of a certain class are subclasses of another class [39], and has a dedicated notation in UML [40, p. 530]. As such, a power type in principle can be used to alleviate the strict separation between type and instance, avoiding the aforementioned duplication of model elements. Yet, power types are conceptual only, and as a result natively lack mechanisms for consistency checks.

As such, if anything changes (in the power type class, or in either of the relevant subclasses), there is subsequently no means to ensure consistency.

Scenario: In the scenario, for illustration purposes, we focus on power types. **Figure 2** presents the use of this modeling pattern for our scenario. In this case, "CustomerGateway" is a subclass of "SmartGridComponentType", and at the same time "CustomerGateway" can be considered as an instance of "SmartGrid-Component", since the latter is a power type. However, as stated power types are a conceptual pattern only, meaning that consistency checks on the subclasses, which act also as instances, are lacking.

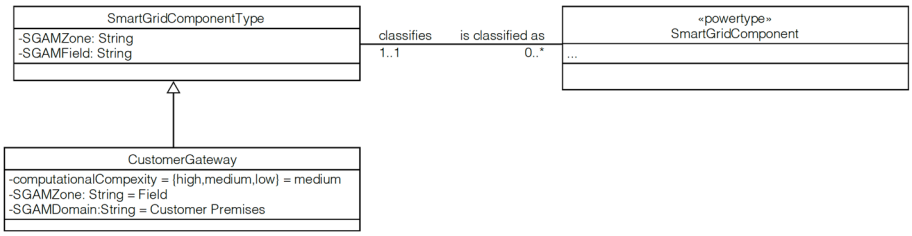

Figure 2 Using power types as a workaround for the type-instance dichotomy

4 Multi-level Reference Modeling

To alleviate the discussed limitations of conventional meta modeling, we now introduce multi-level modeling (Section 4.1), and discuss its possibilities for reference modeling using the same excerpt of the smart grid cyber security reference model (Section 4.2).

4.1 Multi-level Modeling

Partly as a response to the limitations of conventional meta modeling [16], [17], multi-level modeling refers to modeling approaches which share the following core ideas, cf. [41]: (1) one can define an arbitrary number of classification levels in one and the same body of model. This means that one can employ as many classification levels as needed for expressing the domain knowledge at hand [16]. This is opposed to the two classification levels (M_2 and M_1) from conventional meta modeling; (2) one can defer instantiation, meaning that one can constrain the instantiation to a model element residing at a specific classification level [18]. This is opposed to shallow instantiation for conventional meta modeling, whereby one can instantiate only to the directly proceeding level; (3) one can relax the strict separation between type and instance [17], allowing one to populate and use a model with instance level data. This is again opposed to conventional meta modeling which adheres to a strict type-instance dichotomy.

Different multi-level modeling approaches exist, such as, among others, m-objects and m-relations [42], deep instantiation [16], and the Flexible Meta Modeling and Execution Language (FMML^x) [18]. As an exemplary multi-level modeling approach,

for this paper we select FMML^x to show how multi-level modeling alleviates the limitations introduced by conventional meta modeling. One of the reasons for selecting the FMML^x is that, besides the expertise of the authors, it appears to be the only approach with a meta modeling editor (XModeler [18]) that has an integrated language execution engine. For future research this allows for, among others, computational analysis of reference models.

4.2 Addressing Challenges with Multi-level Modeling

Figure 3 shows an excerpt of cyber security reference model created with FMML^x, containing the same domain information captured earlier with UML (Section 3.2). When it comes to expressing both the generic and specific information (Challenge 1), with classification levels being a first class citizen in multi-level modeling, we can naturally model the domain hierarchy, as relevant for the smart grid reference model. Similar to the use of generalization/specialization in Section 3.2, we can thus express domain concepts both on a high level of abstraction (e.g. an "ITComponent" and its attributes), and on a lower level one (e.g., for "SmartGridComponent"). However, in addition, due to having a relaxed type-instance dichotomy, multi-level modeling allows us to express naturally domain information as soon as it is known. For example, to assign a particular value to "minInternalMemory" for a "Customer Gateway". Especially of note here, is that due to the relaxed type-instance dichotomy one can keep the attribute value up-to-date with the data of the running organization. This is in contrast to using default values in UML, which one needs to update separately on the type level on the basis of instance-level data. Also one can concisely express domain information on the basis of having a relaxed type-instance dichotomy only, instead of having to rely on two mechanisms specific to UML (default values and redefinition).

When it comes to coverage of different domain scenarios while avoiding redundancy (Challenge 2), the above multi-level modeling characteristics are equally important. For example, to express characteristics of different types of "IT Component" once, thus avoiding redundancy, while covering a wide range of different domain scenarios through the ability of expressing both the generic and the specific. However, of additional importance for Challenge 2 is the ability of multi-level modeling to defer instantiation of a model element to a particular level of classification. In FMML^x deferred instantiation is expressed through intrinsicness. Intrinsicness, which in Figure **3** is depicted as a white number on a black background, expresses the classification level one instantiates the model element to (intrinsicness is depicted for attributes Figure **3**, but equally can be used for association ends). For our scenario, this intrinsicness allows us to constrain the initialization of values of attributes for "IT Component", which resides on level M_3. For example, for the abstraction hierarchy of "IT Component" we can express that "minInternalMemory : Double" shall be instantiated on level M_1, whereas "actInternalMemory : Double" is to be instantiated on level M_0. In a more general sense, this deferred instantiation through intrinsicness allows us to constrain already on a high level of abstraction when domain information becomes relevant. This in turn provides additional means for ensuring variability. Finally, when it comes to the adaptation of reference models (Challenge 3), multi-level

modeling enforces a *monotonic* model extension [18]. As a result, extensions to a reference model are consistent with the domain rules already encoded into the multi-level model on a higher level of classification. So for example, arbitrarily changing the attribute type "minInternalMemory" from a "Double" to a "String" on a lower level of abstraction would not be allowed. As stated in Section 3.2, UML redefinition is at the very least not clearly defined and underspecified in how it maintains consistency, making it likely that one can violate monotonicity. In addition, as stated in Section 4.1 with multi-level modeling the different levels of classification are all part of one and the same model – conceptually speaking at least. As a result, no matter what adaptations are made, one is in principle adapting one and the same reference model. While this introduces new challenges in its own right, at the very least, it means avoiding redundancies during adaptation.

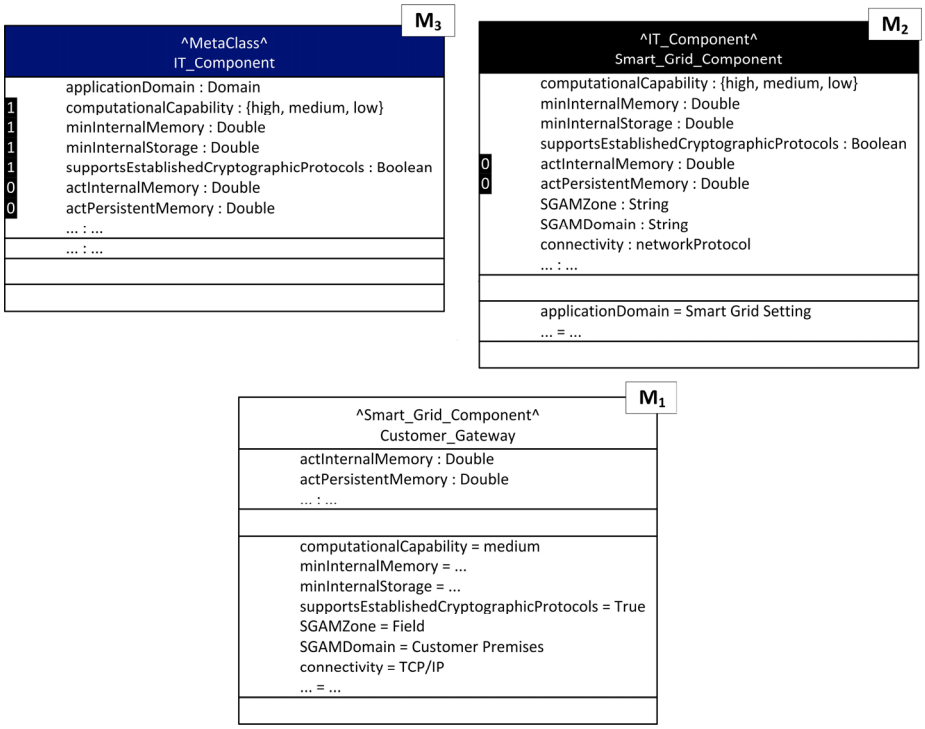

Figure 3 An excerpt from the NISTIR cyber security reference model, reconstructed with the FMML^x

4.3 Summarizing Comparison

Table 1 provides a summarized comparison between using conventional meta modeling and multi-level modeling, as illustrated by their respective application to the same, smart grid cyber-security, reference model.

Table 1. Comparing UML and FMML^x for addressing reference model challenges

Lang.	Challenge 1	Challenge 2	Challenge 3
UML	+ creating hierarchies of concepts with generalization /specialization + assigning values using default values and redefinition - modification of default values restricted to type level - underspecified semantics of redefinition, violation of monotonic model extensions - overloading the level + mature tools and mechanisms	+ covering variability and redundancy partly using the abstraction mechanism mentioned in Challenge 1 - "shallow instantiation", no possibility to constrain model elements according to their classification level; - same issues as under Challenge 1, e.g., monotonic model extensions are likely not guaranteed + mature tools and mechanisms	- reference model adaptation either (1) needs additional consistency checking mechanisms, when simply duplicating it, or (2) leads to redundant model elements, when instantiating it (due to a strict separation between types & instances), or (3) lacks consistency checking mechanisms when using power types. + mature tools and mechanisms
FMML^x	+ an arbitrary number of classification levels allowing to create hierarchies of concepts + relaxed type-instance dichotomy allowing to assign state to classes - immaturity of model management mechanisms	+ intrinsicness (deferred instantiation) + an arbitrary number of classification levels + relaxed type-instance dichotomy - immaturity of model management mechanisms	+ monotonic model extensions + adapting one and the same body of model - immaturity of model management mechanisms

As can be observed, UML offers several mechanisms which at least partly address the challenges discussed so far. Especially, the combined use of generalization/specialization, redefinition, and default values allows us to account for both (a) general concepts and their relevant properties. For example, in our scenario an "IT Component" and its attributes cover a wide range of specific smart grid components, and (b) specific concepts and their properties, to cover specific scenarios. For example, a "Customer Gateway", as relevant for scenarios specifically involving customer premises. This addresses partly Challenges 1 and 2, in the sense of balancing

the general and the specific, avoiding redundancy (by using generalization/specialization), and by allowing for variability, in the sense of covering a wide range of constraints/application scenarios. Also, while using UML we can partially account for reference model adaptation (Challenge 3), in the three manners summarized in Table 1.

As pointed out above, UML provides notable capabilities to satisfy our purposes. In particular, this regards the design of domain hierarchies by using a combination of generalization/specialization, default values, and redefinition. However, UML also comes with a set of limitations. As we have seen these limitations are alleviated by FMML^x mainly since, being a multi-level modeling approach, FMML^x treats *classification levels as a first-class citizen* and relaxes the *strict separation between types and instances*. As such, as we have illustrated with our simplified example, FMML^x offers many features that fit naturally with the idea of reference modeling. Especially, having (1) an arbitrary number of classification levels, as well as (2) a relaxed type-instance dichotomy, allows one to naturally mirror hierarchies of domain information (which is important to Challenge 1 and Challenge 2). Importantly, compared to the UML in FMML^x such hierarchies can be created without redundancies or inconsistencies, and can also conceptually speaking be succinctly created (e.g., without the need to overload the level, or having to use both redefinition and default values as workarounds, when instantiation can suffice). In addition, the intrinsicness offered by FMML^x specifically, and the notion of deferred instantiation generally, allows us to specify at what level of classification a model element should be instantiated. For example, to express that "minInternalMemory : Double" must be instantiated on level M_1, whereas "actInternalMemory : Double" must be instantiated on level M_0. This contributes to expressing variability in the reference model (Challenge 2). In contrast, with the shallow instantiation of UML this kind of constraining ac- cording to level of classification is simply not possible. Finally, FMML^x lends itself naturally to model adaptations (Challenge 3), since it enforces monotonic model extensions, and at least conceptually speaking, one makes the adaptations in one and the same body of model.

However, while promising in terms of the underlying ideas, multi-level modeling, being a relatively novel language architecture, introduces also several challenges. Firstly, multi-level modeling approaches still need to mature in terms of model management. Especially, given that adaptations are made to one and the same body of model, additional mechanisms are needed in order to deal with the increased complexity [43], [44]. This directly impacts Challenge 1–3: while it can naturally mirror domain hierarchies (Challenge 1), deal with variability (Challenge 2) and can ensure consistent model adaptation (Challenge 3), the *usability* of multi-level modeling approaches in terms of typical model management mechanisms (navigating the models, viewpoints, etc.) is, as it currently stands, limited.

For example, returning to our cyber security reference model, beyond our small excerpt the NISTIR 7628 provides a comprehensive coverage of cyber security concerns, which requires an equally comprehensive reference model. It can be foreseen that managing such a comprehensive reference model is challenging to manage with the current state of multi-level modeling approaches. Taking also into consideration the

rapid changes in the electricity sector and its according cyber security concerns, this motivates further the mechanisms to deal with the complexity of multi-level models.

Secondly, an additional concern is that basic multi-level modeling terms have not been properly defined, such as "level", cf. [45]. This in turn impacts the design of multi-level modeling features, such as deferred instantiation, being one of the unique features of multi-level modeling. Particularly, the question emerges to what extent we can assume levels to be absolute, or rather if they are relative to the problem at hand.

Finally, the creation of multi-level models requires a change in the users' mindset: they need to think in multiple levels of classification, and not only two. Assigning concepts to multiple classification levels is not a trivial task, and currently there is lack of guidelines and heuristics for designing such a multi-level model [44], [46].

5 Conclusions and Outlook

In this paper, we have shown how multi-level modeling provides a natural candidate for the creation and use of reference models, compared to reference models based on conventional meta modeling. Especially, as shown on the basis of a comparative scenario, as a language architecture multi-level modeling is a natural fit to address reference model challenges since (a) it treats the notion of a classification level as a first class citizen. As opposed to conventional meta modeling, which is restricted to two classification levels, this allows one to naturally mirror hierarchies of domain concepts inherent to reference modeling, (b) due to relaxing the difference between types and instances, it is easier to adapt and synchronize a reference model conformant to the data of a running organization. Finally, please note that modeling languages not subscribing to the MOF can also be used for the creation and use of reference models. For creating reference models with an emphasis on a static perspective such languages include the data modeling language ERM (as mentioned in Sect. 2.1), or the fact modeling language Object Role Modeling (ORM, [47]). These modeling languages provide abstraction mechanisms which differ from those in conventional meta modeling, like the different set-based generalization/specialization mechanisms inherent to ERM [48, pp. 92-94]. Yet, importantly, even when using an alternative language, these languages still do not treat abstraction levels as a first class citizen. And so, even while these languages may offer additional flexibility, when it comes expressing different abstraction levels, they are still expected to suffer similar fundamental limitations as in the MOF. Of course, still a comparison of non-MOF based to multi-level modeling languages may be warranted for future research.

References

1. Frank, U., Strecker, S.: Open reference models-community-driven collaboration to promote development and dissemination of reference models. EMISA **2**(2) (2007)
2. Fettke, P., Loos, P.: Perspectives on reference modeling. In Fettke, P., Loos, P., eds.: Reference Modeling for Business Systems Analysis. (2007) 1–21

3. Thomas, O.: Understanding the term reference model in information systems research: history, literature analysis and explanation. In: International Conference on Business Process Management, Springer (2005) 484–496
4. Sonntag, A., Fettke, P., Loos, P.: Inductive reference modelling based on simulated social collaboration. In: Proc. of the 2017 Wirtschaftsinformatik conference, AIS (2017)
5. Rehse, J.R., Fettke, P., Loos, P.: A graph-theoretic method for the inductive development of reference process models. SoSyM **16**(3) (2017) 833–873
6. Schütte, R.: Reference models for standard software—scientific myth instead of practical reality? In Bergener, K., Räckers, M., Stein, A., eds.: The Art of Structuring: Bridging the Gap Between Information Systems Research and Practice. Springer International Publishing, Cham (2019) 125–136
7. Kraume, K., Voormanns, K., Zhong, J.: How a global customer service leader is using a reference model to structure its transformation while remaining fast and agile. In Bergener, K., Räckers, M., Stein, A., eds.: The Art of Structuring: Bridging the Gap Between Information Systems Research and Practice. Springer International Publishing, Cham (2019) 101–111
8. Janiesch, C., Winkelmann, A.: The goat criteria—a structured assessment approach for reference models. In Bergener, K., Räckers, M., Stein, A., eds.: The Art of Structuring: Bridging the Gap Between Information Systems Research and Practice. Springer International Publishing, Cham (2019) 63–74
9. Scholta, H., Niemann, M., Delfmann, P., Räckers, M., Becker, J.: Semi-automatic inductive construction of reference process models that represent best practices in public administrations: A method. Information Systems **84** (2019) 63–87
10. NIST Smart Grid Cybersecurity Panel: NISTIR 7628-guidelines for smart grid cyber security vol. 1-3 (2010)
11. Neureiter, C., Engel, D., Uslar, M.: Domain specific and model based systems engineering in the smart grid as prerequisite for security by design. Electronics **5**(2) (2016) 24
12. de Kinderen, S., Kaczmarek-Heß, M.: Multi-level modeling as a language architecture for reference models: On the example of the smart grid domain. In Becker, J., Novikov, D.A., eds.: 21st IEEE Conference on Business Informatics, CBI, Moscow, Russia, July 15-17, Volume 1 - Research Papers, IEEE (2019) 174–183
13. Bagheri, E., Ghorbani, A.A.: UML-CI: A reference model for profiling critical infrastructure systems. Information Systems Frontiers **12**(2) (2010) 115–139
14. Frank, U., Lange, C.: E-MEMO: a method to support the development of customized electronic commerce systems. ISeB **5**(2) (2007) 93–116
15. OMG: The OMG Unified Modeling Language (OMG UML), version 2.5.1. Technical report (2017)
16. Atkinson, C., Kühne, T.: Reducing accidental complexity in domain models. SoSyM **7**(3) (2008) 345–359
17. Atkinson, C., Kühne, T.: The essence of multilevel metamodeling. In: Proc. of the 4th Int. Conf. on The Unified Modeling Language, Modeling Languages, Concepts, and Tools, London, Springer (2001) 19–33
18. Frank, U.: Multilevel modeling – toward a new paradigm of conceptual modeling and information systems design. BISE **6**(6) (2014) 319–337
19. Thomas, O.: Version management for reference models: Design and implementation. In Becker, J., Delfmann, P., eds.: Reference Modeling: Efficient Information Systems Design Through Reuse of Information Models. Physica-Verlag (2007) 1–26

20. vom Brocke, J.: Design principles for reference modeling: reusing information models by means of aggregation, specialisation, instantiation, and analogy. In: Reference modeling for business systems analysis. IGI Global (2007) 47–76
21. Chen, P.P.: The entity-relationship model - toward a unified view of data. ACM Trans. Database Syst. **1**(1) (1976) 9–36
22. Scheer, A.W.: ARIS – Modellierungsmethoden, Metamodelle, Anwendungen. 4 edn. Springer, Heidelberg (2001)
23. OMG: Business Process Model and Notation (BPMN), Version 2.0 (January 2011)
24. Rosemann, M., van der Aalst, W.: A configurable reference modelling language. Information Systems **32**(1) (2007) 1 – 23
25. Fettke, P., Loos, P., Zwicker, J.: Business process reference models: Survey and classification. In: International Conference on Business Process Management, Springer (2005) 469–483
26. Frank, U.: Evaluation of reference models. In: Reference modeling for business systems analysis. IGI Global (2007) 118–140
27. Koch, S., Strecker, S., Frank, U.: Conceptual modelling as a new entry in the bazaar: The open model approach. In: IFIP International Conference on Open Source Systems, Springer (2006) 9–20
28. Matook, S., Indulska, M.: Improving the quality of process reference models: A quality function deployment-based approach. DSS **47**(1) (2009) 60 – 71
29. Reinhartz-Berger, I., Soffer, P., Sturm, A.: Organisational reference models: Sup- porting an adequate design of local business processes. International Journal of Business Process Integration and Management **4**(2) (2009) 134–149
30. Kotut, L., Wahsheh, L.A.: Survey of cyber security challenges and solutions in smart grids. In: 2016 Cybersecurity Symposium, IEEE (2016) 32–37
31. Abercrombie, R.K., Sheldon, F.T., Hauser, K.R., Lantz, M.W., Mili, A.: Risk assessment methodology based on the NISTIR 7628 guidelines. In: System Sciences (HICSS), 2013 46th Hawaii International Conference on, IEEE (2013) 1802–1811
32. Chan, A., Zhou, J.: On smart grid cybersecurity standardization: Issues of designing with NISTIR 7628. IEEE Communications Magazine **51**(1) (2013) 58–65
33. Neureiter, C., Uslar, M., Engel, D., Lastro, G.: A standards-based approach for domain specific modelling of smart grid system architectures. In: System of Systems Engineering Conference (SoSE), 2016 11th, IEEE (2016) 1–6
34. OMG: Meta Object Facility (MOF) core specification (2016) Version 2.5.1.
35. Atkinson, C., Gerbig, R.: Melanie: multi-level modeling and ontology engineering environment. In: Proceedings of the 2nd International Master Class on Model- Driven Engineering: Modeling Wizards. (2012) 1–2
36. Bildhauer, D.: On the relationships between subsetting, redefinition and association specialization. In: 9th Conf. on Databases and Information Systems. (2010)
37. Nieto, P., Costal, D., Gmez, C.: Enhancing the semantics of UML association re- definition. Data & Knowledge Engineering **70**(2) (2011) 182 – 207
38. Warmer, J.B., Kleppe, A.G.: The object constraint language: getting your models ready for MDA. Addison-Wesley Professional (2003)
39. Odell, J.J.: Advanced object-oriented analysis and design using UML. Volume 12. Cambridge University Press (1998)
40. Booch, G., Rumbaugh, J., Jacobson, I.: The unified modeling language reference manual, second edition. Addison-Wesley Reading (2005)

41. Neumayr, B., Schrefl, M., Thalheim, B.: Modeling techniques for multi-level abstraction. In Kaschek, R., Delcambre, L., eds.: The Evolution of Conceptual Modeling. Springer, Berlin (2011) 68–92
42. Neumayr, B., Grün, K., Schrefl, M.: Multi-level domain modeling with m-objects and m-relationships. In: Proceedings of the Sixth Asia-Pacific Conference on Conceptual Modeling-Volume 96, Australian Computer Society, Inc. (2009) 107–116
43. Töpel, D., Benner, B.: Maintenance of multi-level models – an analysis of elementary change operations. In: MULTI@ MoDELS. (2017)
44. Kaczmarek-Heß, M., Nolte, M., Fritsch, A., Betz, S.: Practical experiences with multi-level modeling using FMMLx: A hierarchy of domain-specific modeling languages in support of life-cycle assessment. In Clark, T., Neumayr, B., Rutle, A., eds.: Proc. of the 5th Int. Workshop on Multi-Level Modelling 2018. (2018)
45. Kühne, T.: A story of levels. In Hebig, R., Berger, T., eds.: Proceedings of MOD- ELS 2018 Workshops: ModComp, MRT, OCL, FlexMDE, EXE, COMMitMDE, MDETools, GEMOC, MORSE, MDE4IoT, MDEbug, MoDeVVa, ME, MULTI, HuFaMo, AMMoRe, PAINS co-located with ACM/IEEE 21st International Con- ference on Model Driven Engineering Languages and Systems (MODELS 2018), Copenhagen, Denmark, October, 14, 2018. Volume 2245 of CEUR Workshop Proceedings., CEUR-WS.org (2018) 673–682
46. Almeida, J.P.A., Frank, U., Kuehne, T.: Multi-level modelling (report from Dagstuhl seminar 17492). Dagstuhl Reports **7**(12) (2018) 18–49
47. Halpin, T.: ORM 2. In: OTM Confederated International Conferences „On the Move to Meaningful Internet Systems", Springer (2005) 676–687
48. Elmasri, R.: Fundamentals of Database Systems. Pearson Education India (2004)