Validation and Verification in Domain-Specific Modeling Method Engineering

Qin Ma, Monika Kaczmarek-Hess, Sybren De Kinderen

▶ **To cite this version:**

Qin Ma, Monika Kaczmarek-Hess, Sybren De Kinderen. Validation and Verification in Domain-Specific Modeling Method Engineering. 14th IFIP Working Conference on The Practice of Enterprise Modeling (PoEM), Nov 2021, Riga, Latvia. pp.119-133, 10.1007/978-3-030-91279-6_9 . hal-04323856

HAL Id: hal-04323856

https://inria.hal.science/hal-04323856v1

Submitted on 5 Dec 2023

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

This document is the original author manuscript of a paper submitted to an IFIP conference proceedings or other IFIP publication by Springer Nature. As such, there may be some differences in the official published version of the paper. Such differences, if any, are usually due to reformatting during preparation for publication or minor corrections made by the author(s) during final proofreading of the publication manuscript.

Validation and Verification in Domain-Specific Modeling Method Engineering

Qin Ma[1], Monika Kaczmarek-Heß[2][0000−0002−1621−2775], and Sybren de Kinderen[2]

[1] University of Luxembourg, 2 Avenue de l'Université, 4365 Esch-sur-Alzette, Luxembourg
qin.ma@uni.lu
[2] University of Duisburg-Essen, Universitätstrasse 9, D-45141 Essen, Germany
{monika.kaczmarek-hess,sybren.dekinderen}@uni-due.de

Abstract. Enterprise models have the potential to constitute a valuable asset for organizations, e.g., in terms of enabling a variety of analyses. A prerequisite for realizing this potential is that an enterprise model is syntactically, semantically and pragmatically valid. To ensure these three types of validity, verification and validation (V&V) mechanisms are required to be in place while designing the enterprise modeling method, e.g., to validate identified requirements, to check created enterprise models against syntactic rules, or to ensure intra- and inter-model consistency. Therefore, the objective of this paper is to systematically embed verification and validation (V&V) techniques into the design of (enterprise) domain-specific modeling methods (DSMMs). To this end, we integrate steps and considerations of well-established DSMM engineering processes, and enrich them with V&V techniques based upon our earlier experiences and a literature analysis.

Keywords: DSMM design · validation and verification · design method

1 Introduction

Enterprise models have the potential to be of considerable value to organizations. In addition to facilitating communication, capturing and transferring knowledge, enterprise models also have the potential to enable a variety of analyses [1, 45], which in turn may contribute to various organizational activities, such as enterprise transformation, IT-business alignment or business process management.

However, in order for an enterprise model to be valuable and worth the modeling effort, a pre-requisite is to ensure that the enterprise model is valid in the first place. Following [42, 38, 40], we consider validation and verification (V&V) of an enterprise model, and of an enterprise modeling method, from syntactic, semantic, and pragmatic perspectives. Firstly, an enterprise model must be syntactically valid, namely, it should adhere to all the syntactic rules specified for the modeling language in which the model is expressed. Briefly, for modeling languages, such syntactic rules include typing constraints, cardinality

constraints, and additional well-formedness constraints. Secondly, the enterprise model should also be semantically valid, i.e., it should make sense in the context where the model is supposed to be used. Some example requirements for an enterprise model to be semantically valid include the model containing the necessary information [1, 31], or statements from different parts of the enterprise model, which each capture a different perspective on the enterprise, being consistent [30, p. 32]. Thirdly, the enterprise model needs also to be pragmatically valid, i.e., as an enterprise model is considered to serve some purpose, therefore it should serve this purpose accordingly. For example, if the model is built for visualizing a business process, the choice of appropriate symbols and notations, in terms of a fit with the audience, plays a crucial role in a pragmatic validity check. In contrast, if the model is built for being input to a program to perform some analysis, it should at least be interpretable by the program, hence an exporting mechanism would be expected.

Although the perceived target of validation and verification are enterprise models, the validation and verification consideration should take place even before the existence of enterprise models, cf. [42, 4, 39]. Indeed, to ensure the three types of validity of enterprise models mentioned above, validation and verification techniques are required to be in place while designing the enterprise modeling method, e.g., to validate identified requirements, to check created enterprise models against syntactic rules, to ensure intra- and inter-model consistency, and to evaluate the effectiveness of the modeling method to solve its targeted purposes. Some V&V techniques are informal and rely mostly on human intervention. Others, in turn, are formalized and automated, which requires equipping associated meta modeling platforms with corresponding capabilities. For example, meta modeling platforms should allow for specification of syntactic rules (especially well-formedness constraints) of modeling languages, or provide reusable procedures and algorithms for the implementation of checking mechanisms in modeling methods.

As a response to the above, the objective of this paper is to systematically embed validation and verification techniques into typical activities of the life-cycle of domain-specific modeling methods (DSMMs). Particularly, we take two well-established DSMM development processes [18, 32, 33] as a point of departure, and extend them by explicitly including validation and verification techniques based on our earlier experiences, reported among others in [37, 25, 9], as well as experiences reported in the literature. Thereby, we add systemacy to enterprise modeling application scenarios wherein formalization plays a notable role, be it through formal analysis, simulation, or by means of model checking capabilities.

This paper is structured as follows. First the concept of validation and verification as applied to conceptual modeling is explained, and existing approaches to engineer domain-specific modeling methods are shortly introduced. Then, we present the merge of existing approaches with validation and verification being a first-class citizen, and discuss V&V mechanisms and capabilities in each stage of the process. The paper concludes with final remarks and an outlook on future work.

2 Background

2.1 Validation, Verification and Quality Assessment

Verification and validation are well-established techniques for ensuring the quality of a product within the overall software development life-cycle, and as such, are also important in the field of conceptual modeling, cf. [42]. There is no one commonly accepted definition of quality of a conceptual model, neither of the way how it may be exactly checked. Nevertheless, [42] already pointed to the need to account for syntactic quality (adhering to modeling language syntax), semantic quality (correct elements and relations of the domain), and pragmatic quality (the interpretation of the audience of the model) of conceptual models created, and thus, also on the need to be able to verify and validate those qualities. Others have extended this quality model by adding other quality goals [38], leading to the definition of the SEQUAL framework, cf. [40], being one currently well-established quality framework for conceptual modeling.

To assess model quality verification and validation techniques can play a useful role. An often used definition of verification and validation states that verification is "the process of determining that a model implementation and its associated data accurately represent the developer's conceptual description and specifications" and that validation is "the process of determining the degree to which a model and its associated data provide an accurate representation of the real world from the perspective of the model's intended use" [10, p. 2]. In the context of enterprise modeling, the goal of verification is usually defined as proving that enterprise models are correctly built [8, 7], cf. syntactic quality. Such a verification is usually perceived to be an objective process conducted in a formal manner. Indeed, when it comes to syntactic and well-formedness analysis, there is a wide range of formal methods available, based on various mathematical foundations, such as first order logic, set theory, algebra, and process calculi [29]. In turn, validation usually aims at checking that the obtained result (an enterprise model) is an accurate representation of the domain under study, i.e., to check whether "it corresponds exactly to the expected needs by taking into account (and then limited to) the actor's viewpoint" [7]. Here informal, subjective validation strategies are usually followed such as interviews with domain stakeholders, interactive workshops, and scenario-based evaluation.

2.2 Approaches to Design Domain-Specific (Enterprise) Modeling Methods

According to [19, p. 40] a (domain-specific) modeling method "consists of at least one [domain-specific] modelling language [(DSML)] and at least one corresponding [domain-specific] process model which guides the construction and analysis of models." The process of engineering a modeling method is usually supported by different language workbenches and meta modeling platforms [5, 15, 44, 28]. In the enterprise modeling community especially Eclipse EMF [50], ADOxx [16] and MetaEdit+ [51] gained popularity.

Different approaches have been proposed aiming at guiding the engineering of a modeling method. Based on our study, the two most commonly used in the enterprise modeling community seem to be: the Agile Modeling Method Engineering (AMME) approach, proposed by [32, 33] and further extended, e.g., by [14, 13], and the approach of Frank [18, 20]. When it comes to the Frank's approach, the proposed method is meta modeling language and meta modeling platform independent, and encompasses seven main steps (including feedback-loops): (1) clarification of scope and purpose, (2) analysis of generic requirements, (3) analysis of specific requirements, (4) language specification, (5) design of graphical notation, (6) development of modeling tool, and (7) evaluation and refinement. In addition, for each step a corresponding micro-process and information on participants, input, risks and results are also provided. In turn, the AMME approach, often used together with the meta modeling environment ADOxx [16], as presented in [14] and based, among others, on [16, 35], encompasses the following phases (also including feedback loops): (1) create, i.e., investigation of the system under study, scenarios and requirements, (2) design, i.e., specification of a modeling language and mechanisms/algorithms, (3) formalize, i.e., formalization of the specification of the modeling language for the needs of its implementation, (4) develop, i.e., implementation of the modeling method in a meta modeling environment, and (5) deploy, i.e., creation of a stand-alone application and its distribution.

Although both approaches focus on the full spectrum of a modeling method design, they seem to have different foci. While the AMME approach and its further extensions emphasize the need to be agile, as well as the formalize phase being supported by the proposed meta modeling environment, the distinguished phases have been defined predominantly on a high level, and some of the phases are only detailed in later publications (e.g., the create phase [34]). When it comes to the V&V mechanisms, evaluating acceptance and various quality criteria are explicitly mentioned in the deploy phase, as well as within produce-use cycles, cf. [33], however without providing additional details. Yet, additional mechanisms may be found in other works. For instance [14] focused on the create and design phases of AMME and proposed a modeling method called "Modeling Method Design Environment" (MMDE). The mechanisms and algorithms of MMDE include transformation of models in RDF format with the motivation, among others, to enable validation of meta models using semantic web technologies. In addition, in [34] a modeling method called CoChaCo (Concept-Characteristic-Connector) has been proposed, that among others ensures better traceability of the requirements and their validation in the create process.

Compared to AMME, Frank's approach is more detailed. Although Frank's method places less emphasis on formalization and the role of tooling, it offers an extensive set of guidelines and hints regarding the language design. Moreover, for all phases additional aspects are discussed like required inputs, involved stakeholders, and risks. The need for evaluation of developed artifacts is explicitly accounted for within micro-steps. Nevertheless, specific evaluation methods are only selectively described. For instance, Frank's method includes, on the one

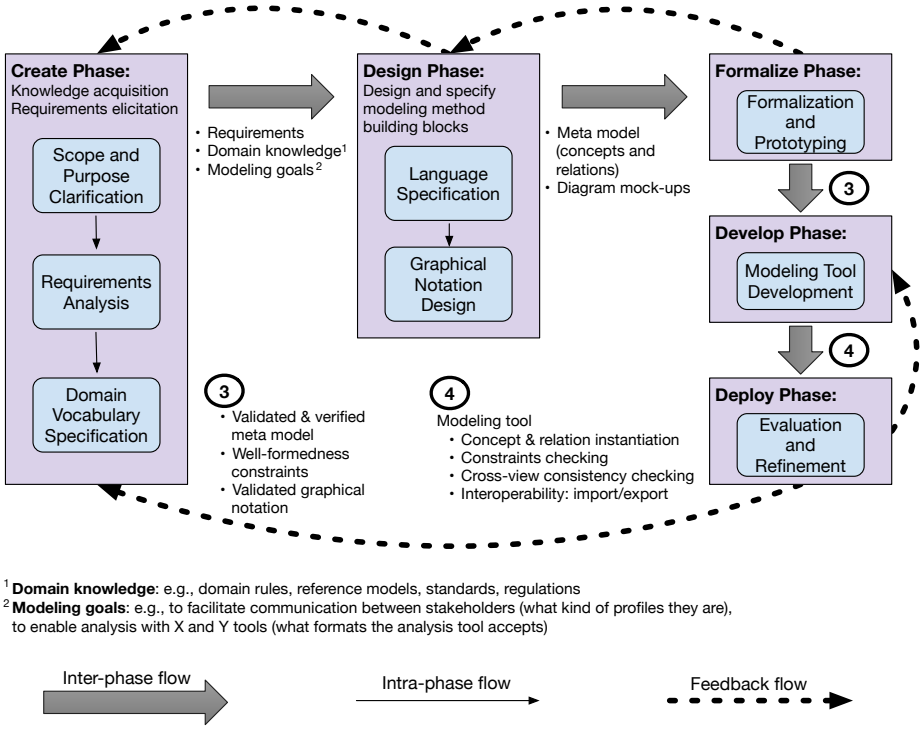

Fig. 1: A DSMM design approach with a focus on V&V techniques across the design life-cycle

hand, activities such as creation of collection of test cases, checking the DSML against requirements, and an analysis of current practice or effects of the DSML's use. However, on the other hand, using formal methods to verify the correctness of a model, being of interest in this paper, are not considered in further detail.

3 Validation and Verification in Enterprise Modeling Method Life-Cycle

The different DSMM engineering approaches discussed in the previous section, do not consider validation and verification as first-class citizens in the engineering process. Yet, as also argued by [42, 4, 39], a suitable modeling method is a prerequisite to ensuring V&V of enterprise models. Therefore, in the following we first integrate two well-established and still complementary approaches to DSMM engineering, namely the AMME approach and Frank's approach. Thereafter, we systematically embed validation and verification techniques to address syntactic, semantic and pragmatic qualities of both the DSMM and models built using it. Doing so, we benefit from our experiences in designing domain specific

modeling methods, application of those, as well as experiences and examples reported in literature.

We conceptualize the integrated DSMM engineering approach in Fig. 1, where the phases are mainly taken from the AMME approach, while being enriched with steps taken from the Frank's approach. We argue that in addition to a sequence of (modeling) steps, each phase also involves critical validation and verification activities, and depending on the result of the verification/validation, corresponding decision/action activities. In the following, we zoom into each phase and focus on the V&V aspect. More specifically, after a brief introduction of modeling activities in each phase, we mainly investigate the artifacts to be validated, types of quality of interest, goals and scopes of V&V, actors performing V&V activities, and exemplary techniques (see Tab. 1–Tab. 5 for a summary).

Create. This phase can be roughly characterized as "early-phase requirements engineering" for DSMM engineering, in the sense that one elicits the purposes, analysis scenarios, and requirements to be fulfilled, which guide the later design decision of the DSMM.

Typical activities: the activities of the create phase include a clarification of the scope and purpose, which the DSMM is supposed to fulfill, as well as an analysis of the (specific) requirements. Additionally, one inventories domain vocabulary, in terms of a reconstruction of the professional terminology.

V&V support: in the create phase, validation takes center stage. On the one hand, we need to ensure that the modeling method being engineered corresponds to the exact needs of actors. On the other hand, we also need to demonstrate that concepts in the modeling language constitute a reasonable representation of the domain at hand. Note that in this phase, validation mechanisms are mainly of an informal character, pertaining mostly to requirements elicitation techniques with stakeholder involvement.

To ensure a fit to *the needs* for the modeling method, there exist a variety of requirements engineering techniques, ranging from goal-oriented requirements engineering to scenario analysis [46]. Some of them have been employed in the context of DSML design. For example, [9] employed goal-oriented requirements engineering to ensure a fit between a DSML's expressiveness and the actor goals the DSML is supposed to achieve, while [49, pp. 36-37] and [20, pp. 11-12] proposed the use of domain scenarios and stating analysis questions in analyzing specific requirements for the DS(M)L at hand. Moreover, for DSMLs specifically, [20, pp. 11-12] also recommended the creation of mock-up diagrams to, already in an early phase, gain a further feedback on the language to be designed, based upon potential, drafty diagrams created with it. In this context, CoChaCo (Concept-Characteristic-Connector) and CoChaCo4ADOxx mentioned earlier may also be used to support requirements gathering, domain knowledge acquisition, concept identification and decomposition, concept selection and mapping, and for the definition of modeling method building blocks [34].

To ensure that the concepts used in the DSMM constitute a *reasonable representation of the domain*, it is important to ensure validation of the domain vocabulary of the DSML. This validation includes a confrontation of the devel-

Table 1: V&V in the Create Phase

Artifact	**Quality**	**Goals and Scope**	**Actor**	**V&VApproach**
Goals and main assumptions, application and use scenarios, mock-up diagrams, requirements, glossary of domain concepts	semantic and pragmatic	Validation of the content of the artifacts developed and the extent to which their correspondent with the domain, expectation of clients and users etc.; Goal: Refining the created artifacts in-line with the feedback obtained	Domain experts, users, clients, business analysts	Workshops and interviews with the corresponding actors, interactive sessions, using techniques such as scenario analysis, repertory grid, and semantic differential. Validation against documents available or available domain description

oped domain vocabulary against documents typical to the domain at hand, or generally speaking corpora of domain concepts [20, 41]. Moreover, it is also vital to receive a feedback on used domain concepts from the prospective end users themselves. Especially interesting in this regard, is to gain an insight into the "personal semantics" [41] of prospective end users, i.e., the semantics end users associate with a concept based upon their experiences with past instances of that concept [26]. Such personal semantics of end users can deviate from those specified in the domain vocabulary, and a lack of awareness of this may lead to considerable issues in the use of a DSML [41]. To gain an impression of such personal concept interpretations, [41] suggested the use of repertory grids and semantic differential studies to elicit the characteristics end users associated with concepts.

Design. In this phase, one specifies core constituents of the DSMM's language specification, in terms of its abstract syntax and corresponding semantics. Also, one specifies the visual notation.

Typical activities: design a language specification (in terms of the abstract syntax and its associated semantics), design a draft visual notation.

V&V support: like in the create phase, the design phase emphasizes validation, and it is likewise done mostly in an informal manner. In the design phase, validation concerns the shaping of the language specification, especially in terms of deciding what concepts and relations will be part of it. Also one can decide on the ontological foundation of DSML concepts. Finally, one should validate the visual notation.

Concerning *shaping of the language specification* one can employ guidelines for the design and assessment for both DSL design generally [36] and DSML design specifically [20, pp. 14-17]. Particularly, for the design phase such guidelines allow one to, for a given set of candidate concepts, decide upon its inclusion in the DSMM being engineered. To this end relevant guidelines include ensuring relevance, pertaining to what extent a candidate concept is relevant to the stated purpose of the DSML, and having invariant semantics (also termed "avoid con-

Table 2: V&V in the Design Phase

Artifact	Quality	Goals and Scope	Actor	V&VApproach
Guidelines for abstract syntax design and assessment, foundational ontologies, guidelines for visual notation design and assessment, glossary of domain concepts	syntactic, semantic and pragmatic	Ensure validation of the content of the key language specification constituencies in terms of it being a reasonable reflection of, both, the domain concepts at hand (when it comes to the abstract syntax), and the development of a suitable visual notation	Domain experts, language users, DSML designer	Workshops and interviews, in addition to conceptual-argumentative work (as it pertains to consistency checks, e.g., as with the use of foundational ontologies)

ceptual redundancy" by [36]), pertaining to a candidate concept having its own essential meaning which sets it apart from other concepts.

It is also possible to validate a language specification against a foundational ontology, with the aim of assessing the semantics of the concepts standing behind a DSMM. Thus, one can address potential ambiguities of interpretations of these semantics. Foundational ontologies have for example been used for assessing several of the ArchiMate extensions [3, 2], and the DEMO transaction patterns [47]. For example, [2, p. 29] pointed out that the "stakeholder" concept from the motivation extension of ArchiMate can have a manifold interpretations, encompassing both agent universals and roles, while one would ideally like to distinguish between these interpretations.

Concerning *validation of the (draft) visual notation*, a well-known means for validating the visual notation are the guidelines proposed in the Physics of Notation (PoN) [43]. PoN also has various adaptations. For example, building on PoN's notion of semantic transparency, i.e., the extent to which the visualization of a concept suggests its meaning, the work of [6] presented an approach for evaluating and improving the semantic transparency of concept notations.

Formalize. This phase concerns the application of formal methods to formalize the design of the DSMM in order to enable its formal verification and validation.

Typical activities: language specification (which typically consists of a meta model specification and a set of well-formedness rules) is defined in terms of mathematical notations. Such mathematical notations include set theory, first-order logic, or algebra. In practice, this can be achieved by implementing a prototype of the language specification using a meta modeling platform that has a formal foundation. For example, the ADOxx platform is based on the FDMM formalism which describes the core constituents of ADOxx meta models in terms of set theory and first order logic statements [17]. The Lightning Language Workbench [24, 21] is another example of a DSML engineering platform with a formal foundation. In Lightning, language specification and concrete syntax design are both implemented as Alloy models [29], and their connection in terms of an F-

Alloy model transformation [23]. The implementation is then executed by the Alloy Analyzer and the F-Alloy Interpreter to generate instance models of the language (as it is designed), and to render the generated models in a domain specific visualisation [22]. Here, the visualized models adhere to the design of the visual notation.

V&V support: this phase is dedicated to the validation and verification of the DSMM. Firstly, mathematical proofs or verification can be run on the formal language specification, in order to analyze properties of the DSML, to detect errors in the language specification or inconsistency among well-formedness rules, or to simulate the formal semantics of the language. Taking the Alloy-based Lightning language workbench for example, the following two verification scenarios can be envisioned: (1) To check if the language (as it is designed) has a property P, we can wrap P in the form of an assertion and check it against the Alloy model, which is the formal counterpart of the language specification. In case a counter-example is found by the Alloy Analyzer, this counter-example constitutes an instance of the language for which the property is violated. The parts of the instance that violate the property are also indicated to make debugging easier; (2) Errors in the language specification, such as inconsistent type constraints, or a conflict between two well-formedness rules, will be signaled by the fact that no instances can be generated from the language specification (as it is designed currently). The first type of verification scenario can be exploited to ensure that semantic qualities are respected, such as the necessary amount of information (in terms of concepts and relations) is captured, and to ensure that domain rules or standards are respected. During the second type of verification scenario, we check the integrity of the DSML itself. Note that formal method based verification is often fully automated and offers more rigor and thorough checking of the language specification than any human checking could do.

Secondly, example instance models generated from the specification of the modeling language (as it is designed so far) enable validation of both semantic and pragmatic qualities. On the semantic side, these example models basically constitute the extensional definition of the language. Omissions (e.g., missing well-formedness rules), excess (e.g., unnecessary concepts), or mismatching (e.g., a relation between wrong types of concepts) can be detected by simply reviewing these example models. The reviewing activity is further enhanced when the example models are visualized in a domain specific notation, because example models foster understanding for domain experts and non-technical users. Hence, these stakeholders will be able to pinpoint errors. Moreover, the domain-specific notation used to visualize example models is indeed the notation which is required by users in the Create phase, and which has been defined and agreed upon among the stakeholders during the Design phase (using the discussed validation techniques of said phase). As such, seeing the visual notation applied to concrete examples also allows the users validate the fitness of the notation itself, being one kind of pragmatic quality.

Table 3: V&V in the Formalize Phase

Artifact	**Quality**	**Goals and Scope**	**Actor**	**V&VApproach**
Formal language specification, generated instance models	Semantic quality and pragmatic quality	Ensuring semantic soundness and completeness; ensuring fitness of the concrete syntax	Domain expert, user, and language engineer	Formal verification, reviewing

Develop. Based on the DSMM definition that has been designed, formalized, verified and validated in previous phases, this phase concerns the implementation of a modeling tool to enable modeling activities with the DSMM.

Typical activities: Among others, the modeling tool typically consists of a graphical editor (to support different diagram types to model different subsets of concepts and relationships), validation mechanisms (to ensure syntactic validity of models and to ensure consistency across different diagrams), and provides import/export functions (to ensure interoperation with external tools). As a consequence, typical activities during this phase amount to realizing all these components of the modeling tool, manually, automatically, or in a hybrid manner.

V&V support: After several iterations of the previous phases, we arrive at a formally verified and user validated specification of the DSMM, with the desired syntactic, semantic, and pragmatic qualities. It is thus crucial in this phase to preserve these qualities of the DSMM (and subsequently to ensure these qualities of models that can be built with the modeling method), by demonstrating that the modeling tool produced in this phase indeed respects the specification. Given that the modeling tool is a piece of software, one can leverage software verification techniques, such as software testing, formal verification [48] (i.e., to formally demonstrate that a program satisfies a formal specification), and program derivation [11, 27, 12][3].

The validation mechanisms implemented in the modeling tool ensure syntactic validity of models created using the modeling tool. This can be achieved either in a preventive manner or through checking. A preventive manner forbids syntactically incorrect models to be created at all in the first place. For example, prevention can be achieved by means of auto-completion of integral missing elements, or disallowing relating two type incompatible instances. Syntactic validity can also be achieved by checking the models against well-formedness constraints after models are created. Moreover, validation mechanisms should ensure coherence between (parts of) models that are obtained either by vertical refinement (e.g., decomposition of a business process into sub-processes, or the internal description of an organizational unit), or by horizontal refinement (e.g.,

[3] Program derivation means to derive an executable program from a formal specification through mathematical means. The program thus obtained is correct by construction.

Table 4: V&V in the Develop Phase

Artifact	**Quality**	**Goals and Scope**	**Actor**	**V&VApproach**
Modeling tool	Syntactic quality, semantic quality and pragmatic quality	To preserve the semantic and pragmatic qualities that have already been achieved in previous phases	Software developer, language engineer, user	Software testing, formal verification, program derivation

two diagrams describing the same phenomenon but from different views, or the same organization but from different perspectives).

Many meta modeling platforms, such as ADOxx, MetaEdit+, or EMF+Sirius, come with generic configurable components ready to be reused for the implementation of the basic functions of the modeling tool, such as the graphical editor. However, when it comes to reusable algorithms or procedures for validation mechanisms, more support is still desired, for example by capitalising on formal methods, as suggested in [37].

Deploy. Within the deploy phase the systematic evaluation of the modeling tool, and thus, also the proposed modeling method takes place. If needed, the revision of the modeling tool, or parts of the modeling method follows, cf. Fig. 1.

Typical activities: in this phase one evaluates the modeling tool supporting the designed domain-specific modeling method regarding, both, (1) its capabilities and extent to which the requirements formulated in the first phase are fulfilled, and (2) the contribution of the designed domain-specific modeling method, compared to the current situation, to improving productivity or quality of the targeted class of problems, [18, pp. 57-58], e.g., whether the domain-specific modeling method allows us to address the targeted problem/finding the solution by using less resources (such as requiring less time).

V&V support: in this phase one focuses on the informal evaluation of the modeling tool and the modeling method, thus relying on subjective experience of modelers. By using the tool to solve targeted problems, i.e., using the tool in practice, the users evaluate mainly the pragmatic quality, nevertheless, they may also identify syntactic or semantic concerns. Apart from checking whether created models constitute a reasonable representation of the domain under study, and whether the requirements specified are accounted for, this phase enables also the detailed analysis of the modeling method using practical use scenarios. Following [18], this allows to compare current practice against the use of domain-specific modeling method in order to check, e.g., learning effort, productivity or quality of the solution and documentation. In addition, a comparative analysis of utility and costs can also be conducted in this phase [18, p. 58].

Table 5: V&V in the Deploy Phase

Artifact	Quality	Goals and Scope	Actor	V&V Approach
Modeling method and supporting tool	Pragmatic, semantic and syntactic quality	A comparative assessment of current practice and effects of using the modeling method on problem solving/analysis	Domain Experts, Users	Informal evaluation using use scenarios, workshops with domain experts, controlled experiments and walk-through

4 Conclusions

To foster valid enterprise models, in this paper we promoted validation and verification to first-class citizens during the engineering of a domain-specific modeling method. To this end we synthesized existing approaches to engineer a domain-specific modeling method, and discussed the validation and verification mechanisms and their roles in different phases of DSMM engineering. In our discussion, we explicitly differentiated between the syntactic, semantic and pragmatic quality of DSMM's artifacts, as well as stress the role of meta modeling platforms and other language engineering workbenches in the validation and verification process.

Due to space restrictions, we do not provide a detailed discussion on the resulting requirements that meta modeling platforms or language engineering workbenches should fulfill, in order to provide a desired support to engineer domain-specific modeling methods in line with the validation and verification discussions presented in this paper. However, considering the arguments we have raised, availability of such requirements would be necessary in order to be able to answer the question: To what extent existing tools/platforms enable validation and verification in the domain of enterprise modeling? Our initial study clearly indicates that while support for the specification and checking of cardinality constraints and typing constraints is provided by most of the platforms, the support for other types of V&V varies substantially. For future work, it would thus be interesting to systematically derive requirements using this paper as a baseline, and to confront different meta modeling platforms against these.

References

1. Antunes, G., Barateiro, J., Caetano, A., Borbinha, J.: Analysis of federated enterprise architecture models. In: ECIS 2015 Completed Research Papers (2015), paper 10
2. Azevedo, C.L., Almeida, J.P.A., van Sinderen, M., Quartel, D., Guizzardi, G.: An ontology-based semantics for the motivation extension to ArchiMate. In: 2011 IEEE 15th International Enterprise Distributed Object Computing Conference. pp. 25–34. IEEE (2011)
3. Azevedo, C.L., Iacob, M.E., Almeida, J.P.A., van Sinderen, M., Pires, L.F., Guizzardi, G.: Modeling resources and capabilities in enterprise architecture: A well-

founded ontology-based proposal for ArchiMate. Information systems **54**, 235–262 (2015)
4. Barjis, J.: Collaborative, participative and interactive enterprise modeling. In: International Conference on Enterprise Information Systems. pp. 651–662. Springer (2009)
5. Bork, D.: Metamodel-based analysis of domain-specific conceptual modeling methods. In: Buchmann, R.A., Karagiannis, D., Kirikova, M. (eds.) The Practice of Enterprise Modeling. pp. 172–187. Springer International Publishing, Cham (2018)
6. Bork, D., Roelens, B.: A technique for evaluating and improving the semantic transparency of modeling language notations. Software and Systems Modeling pp. 1–25 (2021)
7. Chapurlat, V., Braesch, C.: Verification, validation, qualification and certification of enterprise models: Statements and opportunities. Computers in Industry **59**(7), 711–721 (2008), enterprise Integration and Interoperability in Manufacturing Systems
8. Chapurlat, V., Kamsu-Foguem, B., Prunet, F.: A formal verification framework and associated tools for enterprise modeling: Application to ueml. Comput. Ind. **57**(2), 153–166 (Feb 2006)
9. De Kinderen, S., Ma, Q.: Requirements engineering for the design of conceptual modeling languages. Applied Ontology **10**(1), 7–24 (2015)
10. Department of Defense: Instruction 5000.61: Dod modeling and simulation (m&s) verification, validation and accreditation (vv&a). Tech. rep., Department of Defense (2003)
11. Dijkstra, E.W.: A Discipline of Programming. Pearson (1976)
12. Dijkstra, E.W., Feijen, W.: A Method of Programming. Addison Wesley Longman (1988)
13. Efendioglu, N., Woitsch, R.: Modelling Method Design: An ADOxx Realisation. In: 2016 IEEE 20th International Enterprise Distributed Object Computing Workshop (EDOCW). pp. 1–8. IEEE Computer Society, Los Alamitos, CA, USA (sep 2016)
14. Efendioglu, N., Woitsch, R., Karagiannis, D.: Modelling method design: a model-driven approach. In: Anderst-Kotsis, G., Indrawan-Santiago, M. (eds.) Proceedings of the 17th International Conference on Information Integration and Web-based Applications & Services, iiWAS 2015, Brussels, Belgium, December 11-13, 2015. pp. 59:1–59:10. ACM (2015)
15. Erdweg, S., van der Storm, T., Völter, M., Tratt, L., Bosman, R., Cook, W.R., Gerritsen, A., Hulshout, A., Kelly, S., Loh, A., Konat, G., Molina, P.J., Palatnik, M., Pohjonen, R., Schindler, E., Schindler, K., Solmi, R., Vergu, V., Visser, E., van der Vlist, K., Wachsmuth, G., van der Woning, J.: Evaluating and comparing language workbenches: Existing results and benchmarks for the future. Computer Languages, Systems & Structures **44**, 24–47 (2015), special issue on the 6th and 7th International Conference on Software Language Engineering (SLE 2013 and SLE 2014)
16. Fill, H., Karagiannis, D.: On the Conceptualisation of Modelling Methods Using the ADOxx Meta Modelling Platform. EMISA **8**(1), 4–25 (2013)
17. Fill, H.G., Redmond, T., Karagiannis, D.: Formalizing meta models with FDMM: The ADOxx case. In: Proceedings of the 14th International Conference on Enterprise Information Systems (ICEIS 2012). LNBIP, vol. 141, pp. 429–451 (2013)
18. Frank, U.: Outline of a method for designing domain-specific modelling languages. ICB Research Report 42, University of Duisburg-Essen, Essen (2010)
19. Frank, U.: Multi-perspective enterprise modelling: background and terminological foundation. ICB-Research Report 46, University of Duisburg-Essen, Essen (2011)

20. Frank, U.: Domain-specific modeling languages: Requirements analysis and design guidelines. In: Reinhartz-Berger, I., Sturm, A., Clark, T., Cohen, S., Bettin, J. (eds.) Domain Engineering, pp. 133–157. Springer, Berlin (2013)
21. Gammaitoni, L.: On the Use of Alloy in Engineering Domain Specific Modeling Languages. Phd thesis, University of Luxembourg (2017)
22. Gammaitoni, L., Kelsen, P.: Domain-specific visualization of alloy instances. In: Proceedings of the 4th International ABZ Conference, ABZ 2014. LNCS, vol. 8477, pp. 324–327 (2014)
23. Gammaitoni, L., Kelsen, P.: F-alloy: a relational model transformation language based on alloy. Software and Systems Modeling **18**(1), 213–247 (2019)
24. Gammaitoni, L., Kelsen, P., Glodt, C.: Designing languages using lightning. In: Proceedings of the 2015 ACM SIGPLAN International Conference on Software Language Engineering (SLE 2015). p. 77–82 (2015)
25. Gammaitoni, L., Kelsen, P., Ma, Q.: Agile validation of model transformations using compound F-Alloy specifications. Science of Computer Programming **162**, 55–75 (2018)
26. Geeraerts, D.: Theories of lexical semantics. Oxford University Press (2010)
27. Gries, D.: The Science of Programming. Springer (1981)
28. Iung, A., Carbonell, J., Marchezan, L., Rodrigues, E., Bernardino, M., Basso, F.P., Medeiros, B.: Systematic mapping study on domain-specific language development tools. Empirical Software Engineering **25**(5), 4205–4249 (2020)
29. Jackson, D.: Software Abstractions: Logic, Language, and Analysis. The MIT Press (2 2012), revised edition
30. Jeusfeld, M.A.: SemCheck: Checking Constraints for Multi-perspective Modeling Languages, pp. 31–53. Springer International Publishing, Cham (2016)
31. Johnson, P., Lagerström, R., Närman, P., Simonsson, M.: Enterprise architecture analysis with extended influence diagrams. Information Systems Frontiers **9**(2-3), 163–180 (2007)
32. Karagiannis, D.: Agile modeling method engineering. In: Karanikolas, N.N., Akoumianakis, D., Nikolaidou, M., Vergados, D.D., Xenos, M., Giaglis, G.M., Gritzalis, S., Merakos, L.F., Tsanakas, P., Sgouropoulou, C. (eds.) Proceedings of the 19th Panhellenic Conference on Informatics, PCI 2015, Athens, Greece, October 1-3, 2015. pp. 5–10. ACM (2015)
33. Karagiannis, D.: Conceptual Modelling Methods: The AMME Agile Engineering Approach. In: Silaghi, G.C., Buchmann, R.A., Boja, C. (eds.) Informatics in Economy. pp. 3–19. Springer International Publishing, Cham (2018)
34. Karagiannis, D., Burzynski, P., Utz, W., Buchmann, R.A.: A metamodeling approach to support the engineering of modeling method requirements. In: Damian, D.E., Perini, A., Lee, S. (eds.) 27th IEEE International Requirements Engineering Conference, RE 2019, Jeju Island, Korea (South), September 23-27, 2019. pp. 199–210. IEEE (2019)
35. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) E-Commerce and Web Technologies, Third International Conference, EC-Web 2002, Aix-en-Provence, France, September 2-6, 2002, Proceedings. Lecture Notes in Computer Science, vol. 2455, p. 182. Springer (2002)
36. Karsai, G., Krahn, H., Pinkernell, C., Rumpe, B., Schindler, M., Völkel, S.: Design guidelines for domain specific languages. arXiv preprint arXiv:1409.2378 (2014)
37. de Kinderen, S., Ma, Q., Kaczmarek-Heß, M.: Towards Extending the Validation Possibilities of ADOxx with Alloy. In: Grabis, J., Bork, D. (eds.) The Practice of Enterprise Modeling. pp. 138–152. Springer International Publishing, Cham (2020)

38. Krogstie, J.: Evaluating UML Using a Generic Quality Framework, p. 1–22. IGI Global, USA (2003)
39. Krogstie, J., Sindre, G., Jørgensen, H.: Process models representing knowledge for action: a revised quality framework. European Journal of Information Systems **15**(1), 91–102 (2006)
40. Krogstie, J., Sindre, G., Lindland, O.I.: 20 years of quality of models. In: Jr., J.A.B., Krogstie, J., Pastor, O., Pernici, B., Rolland, C., Sølvberg, A. (eds.) Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE, pp. 103–107. Springer (2013)
41. van der Linden, D., Hoppenbrouwers, S., Lartseva, A., Molnar, W.: Beyond terminologies: Using psychometrics to validate shared ontologies. Applied Ontology **7**(4), 471–487 (2012)
42. Lindland, O.I., Sindre, G., Sølvberg, A.: Understanding quality in conceptual modeling. IEEE Softw. **11**(2), 42–49 (Mar 1994)
43. Moody, D.L.: The physics of notations: Toward a scientific basis for constructing visual notations in software engineering. IEEE Transactions on Software Engineering **35**(6), 756–779 (2009)
44. Negm, E., Makady, S., Salah, A.: Survey on domain specific languages implementation aspects. International Journal of Advanced Computer Science and Applications **10**(11) (2019)
45. Niemann, K.D.: From enterprise architecture to IT governance, vol. 1. Springer (2006)
46. Pohl, K.: Requirements engineering: fundamentals, principles, and techniques. Springer Publishing Company, Incorporated (2010)
47. Poletaeva, T., Guizzardi, G., Almeida, J.P.A., Abdulrab, H.: Revisiting the DEMO transaction pattern with the unified foundational ontology (UFO). In: Enterprise Engineering Working Conference. pp. 181–195. Springer (2017)
48. Seligman, E., Schubert, T., Kumar, M.V.A.K.: Formal Verification: An Essential Toolkit for Modern VLSI Design, 1st Edition. Morgan Kaufmann (2015)
49. Sobernig, S.: Variability support in dsl development. In: Variable Domain-specific Software Languages with DjDSL, pp. 33–72. Springer (2020)
50. Steinberg, D., Budinsky, F., Merks, E., Paternostro, M.: EMF: eclipse modeling framework. Pearson Education (2008)
51. Tolvanen, J.P., Kelly, S.: Metaedit+ defining and using integrated domain-specific modeling languages. In: Proceedings of the 24th ACM SIGPLAN conference companion on Object oriented programming systems languages and applications. pp. 819–820 (2009)