Metamodel-Based Analysis of Domain-Specific Conceptual Modeling Methods

Dominik Bork(✉)

Research Group Knowledge Engineering, University of Vienna,
Waehringer Street 29, 1090 Vienna, Austria
dominik.bork@univie.ac.at

Abstract. Metamodels play a pivotal role in conceptual modeling as they manifest the abstraction level applied when creating conceptual models. Consequently, design decisions made by the metamodel developer determine utility, capabilities, and expressiveness of the conceptual modeling language - and eventually the created models. However, only limited research defines and applies metrics for analyzing the structure and capabilities of a metamodel, and eventually support the development of new metamodels. This not only concerns general-purpose modeling languages, but also domain-specific ones, which usually undergo shorter update cycles. The paper at hand introduces a generic analysis framework to syntactically analyze modeling languages. The framework is applied to 40 metamodels of domain-specific conceptual modeling languages (DSML). This research establishes a foundation to support metamodel development in the future. The contribution of this paper is threefold: (i) an analysis framework for conceptual modeling method metamodels is proposed, (ii) results from applying this framework to 40 ADOxx-based DSML metamodels are presented, and (iii) a human-based reasoning after comparison of these results with Ecore-based metamodels is conducted.

Keywords: Domain-specific modeling · Conceptual modeling
Metamodel · Analysis · OMiLAB · Metrics

1 Introduction

Conceptual modeling historically plays an important role in information and computer science research. Numerous modeling approaches have been designed. Some of which aim for general applicability and wide adoption - general-purpose modeling languages like Unified Modeling Language (UML), and Business Process Modeling and Notation (BPMN) - whereas others aim to precisely address

R. A. Buchmann et al. (Eds.): PoEM 2018, LNBIP 335, pp. 172–187, 2018.
https://doi.org/10.1007/978-3-030-02302-7_11

the specific characteristics of a certain domain - domain-specific modeling languages (DSMLs). While the focus in early years was on the specification of general-purpose modeling languages, nowadays, researchers also emphasize the importance of creating DSMLs (cf. [29] for recently developer DSMLs). Such DSMLs employ an abstraction level that is aligned to the purposes of specific stakeholders in a specific application domain.

Metamodels are at the heart of any conceptual modeling language as they establish the abstraction level to be applied while creating models. This abstraction level is realized by means of the available concepts of a modeling language and the valid combinations thereof. Decisions taken by the metamodel developer determine quality, expressiveness, and utility of the modeling language (cf. [23]). A lot of research is focusing on the evaluation of modeling methods from a semantical point of view [18,19], from a notational point of view [7,37,41], or on methodological guidance in developing modeling languages [16,27] and methods [3,13,36]. By contrast, only limited research focuses on metamodels and their design. *"The rationale behind decisions made during the language/model specification are implicit so it is not possible to understand or justify why, for instance, a certain element of the language was created with that specific syntax or given that particular type."* [26] Thus, there is a research gap in analyzing existing and providing guidance for the development of new metamodels. Moreover, to the best of our knowledge, no comparative analysis has been performed targeting specifically DSML metamodels. The aim of this research is to derive empirical quantitative answers towards filling the identified research gap.

The aim of this paper is to assess current DSML metamodel designs and to derive ideas on how to improve metamodel design in the future. The contribution of this paper is aligned to two research questions (RQ): *RQ-1: How are domain-specific metamodels structured?*, and *RQ-2: Are there differences between ADOxx-based and Eclipse-based metamodels?*. The analysis reported in this paper used 40 openly available DSML metamodels of the Open Models Laboratory (OMiLAB) [8] that have been realized with the ADOxx metamodeling platform [15]. For the analysis we introduce a framework that adopts a set of metamodeling metrics [12,35]. The adoption of the metrics respects the idiosyncrasy of both, conceptual modeling generally and the ADOxx platform in particular. *"Similarly to software, metrics can be used to obtain objective, transparent, and reproducible measurements on metamodels too"* [12, p. 55]. Our work adds to the knowledge base by focusing on metrics rather than a qualitative evaluation of metamodels, and by focusing on ADOxx-based DSML metamodels.

This paper is structured as follows: Sect. 2 defines the foundations of this work by introducing domain-specific conceptual modeling, the Open Models Laboratory, as well as ADOxx and Eclipse as metamodel development platforms. An overview of related works is presented in Sect. 3 before Sect. 4 proposes the generic analysis framework. The results of applying this framework to 40 DSMLs are discussed in Sect. 5. Eventually, the paper closes with some concluding remarks and implications for research and practice in Sect. 6.

2 Foundations

2.1 Conceptual Modeling Methods

Conceptual modeling methods facilitate the reduction of complexity by applying abstraction for a specific purpose. Such methods are composed of *modeling language*, *modeling procedure*, and *mechanisms & algorithms* [28]. A vital part of a modeling method is the modeling language which can be further decomposed into *syntax*, i.e, the available syntactic elements, *notation*, and *semantics*, specifying the graphical representation and the meaning of the syntactic elements, respectively. The modeling procedure describes the steps to be applied by the modeler in order to create valid models. Mechanisms & algorithms define the model processing functionality provided by the modeling method, e.g., simulation, model transformation.

Based on the pragmatics and purpose, domain-specific modeling methods can be distinguished from general-purpose ones. The former has the potential to address domain-specificity in all aspects of a modeling method, while the latter aims for comparability, interoperability, and standardization across domains. A further differentiation can be drawn when considering the purpose of modeling methods. In computer science, most modeling methods are designed for model-driven systems engineering using the Eclipse Modeling Framework[1] which rely on Ecore metamodels (see Sect. 2.3). Such models often lack proper visualization and focus instead on the capabilities of model transformation and code generation. By contrast, conceptual modeling methods are used to create abstract representations of some part of the real world for *"human users, for purposes of understanding and communication"* [38]. In this perception, which is the one we apply in this paper, modeling of software systems and code generation is only one out of many possible purposes for conceptual modeling.

2.2 The Open Models Laboratory (OMiLAB)

OMiLAB, www.omilab.org is an open platform for the conceptualization of modeling methods, combining open source and open communities with the goal of fostering conceptual modeling [8]. Modeling tools realized as a project within the OMiLAB are based on the ADOxx metamodeling platform (see Sect. 2.3). Relevance of the OMiLAB is reflected in the high number of international contributors. 40 different DSMLs have been successfully conceptualized - addressing diverse domains like enterprise modeling [14], enterprise architecture management [4], design thinking [5,22], and knowledge acquisition [9,10]. A detailed description with sample conceptualizations is given in [29].

2.3 Metamodeling Platform

Metamodeling platforms are used for the development of modeling tools by raising the abstraction level to a more elaborate level that is adequate for

---

[1] Eclipse Modeling Framework [online], https://www.eclipse.org/modeling/emf/, last checked: 28.08.2018

method engineers to realize their modeling tools. The goal is to enable also non-programmers to realize modeling tools. This is achieved by providing a rich set of preconfigured functionality attached to a generic meta-metamodel. The method engineer then only needs to adapt this meta-metamodel to her domain. Moreover, engineers can benefit from existing tool developments and reuse/extend existing implementations.

ADOxx. ADOxx[2] has been successfully used in academia and industry for over two decades. The platform comes with a rich set of domain-independent functionality like model management, user management, and user interaction. What is left to be done for metamodel developers is to [2]: (1) configure the specific metamodel by referring its concepts to the meta-metamodel concepts of ADOxx; (2) provide a visualization for the concepts and combine them into logical chunks, i.e., ADOxx modeltypes; and (3) realize additional functionality like model transformations, queries, or simulations on top of the modeling language.

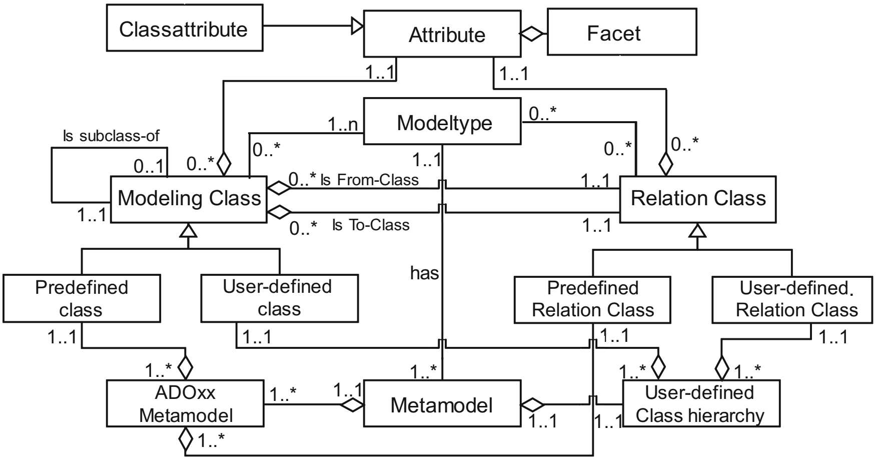

Fig. 1. Excerpt of the ADOxx meta-metamodel (adapted from [15])

A metamodel realized with ADOxx is composed of *modeltypes* which themselves comprise predefined and user-defined *modeling classes* and *relation classes* (Fig. 1). Following a graph-based structure, modeling classes refer to nodes and relation classes to edges between nodes. Attributes define the semantics of all ADOxx classes. Functionality in ADOxx is attached to predefined abstract meta classes of the ADOxx meta-metamodel (see Fig. 1). When defining an inheritance relationship between domain-specific concepts and predefined abstract meta classes, the functionality is inherited. Consequently, the metamodel design decisions determine the functionality of the resulting modeling tool.

---

[2] ADOxx platform [online], http://www.adoxx.org, last checked: 27.08.2018

Table 1 briefly introduces the most important ADOxx meta classes which are also part of the metrics introduced in Sect. 4. ADOxx meta classes are either static (prefix 'S') or dynamic (prefix 'D'). The former employ a tree-based structure for hierarchies between static classes while the latter employ a graph-based structure for realizing simulations. Furthermore, ADOxx modeling classes can be either *abstract*, thus not instantiable by the modeler, or *concrete*, thus can be instantiated thereby creating a conceptual model.

Table 1. Excerpt of ADOxx meta classes

Meta class	Description
D_Aggregation	Every modeled object 'a' having its x/y coordinates within the drawing area of any container 'b' has the relation *'a' is-inside 'b'*. Moreover, subclasses come with a self-defined "drawing area" by means of resizeable rectangles
D_Swimlane	Also provides the "is-inside" relation but the "drawing area" is limited to strict horizontal or vertical rectangles
D_Event	Encapsulates all nodes of a graph necessary for its simulation. Subclasses are e.g., D_Start, D_Subgraph, D_Activity, D_Decision
S_Group	This class represents a node in a tree structure
S_Aggregation	Special kinds of nodes in a tree structure. Similar semantics as for the dynamic counterparts
S_Swimlane	
S_Person	Implements person-dependent aspects like wages and working hours

Eclipse Modeling Framework. The Eclipse Modeling Framework (EMF) provides a generic metamodel called Ecore, one can inherit from in order to

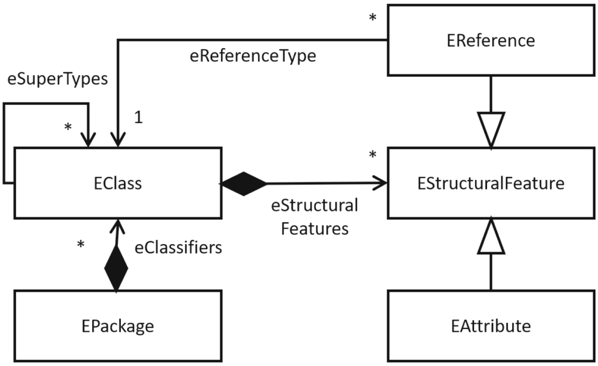

Fig. 2. Excerpt of the Ecore meta-metamodel [30]

develop metamodels. A dominant focus of using EMF is the generation of code from models in model-driven development. Thus, models are primarily perceived as “structured data models”. EMF comes with a rich set of functionality that eases the generation of Java classes from EMF models.

The Ecore metamodel comes with a plethora of predefined meta classes necessary for code generation purposes and general model management. Relevant for conceptual modeling are particularly the classes visualized in Fig. 2. Ecore-based metamodels are clustered in *EPackages* which are comprised of *EClasses*. Every EClass itself is comprised of *EStructuralFeatures* like changeability or volatility. Two special kinds of features are further distinguished: *EReferences* relate two EClass instances to each other, whereas *EAttributes* define additional properties of EClasses.

3 Related Works

A lot of research can be found on the analysis of models, focusing for example on the usage of modeling concepts by modelers [33], the evaluation of modeling languages according to their notation [7,32,37], their semantics [19,42], their ontological completeness [18], their metamodels [34], their specification techniques [6], or their applicability in certain use cases [20] and domains [24,25]. These approaches however never investigate the syntactic metamodel backbone of the modeling language and the way metamodels are structured. Up to now, only limited research structurally assesses metamodels by applying metrics. The relevant works will be reviewed in the following sub-sections.

An approach for metamodel analysis was proposed in [43]. The authors introduced metrics for the syntactic analysis of metamodels. They distinguish between *metrics concerning meta-classes* and *metrics concerning meta-features*. The former comprises the number of (abstract) classes, and whether classes have features. Moreover, average numbers for *features*, *attributes*, and *references* are computed. The metrics for meta-features consider some of the class-level metrics, however, applied to the whole metamodel. The authors developed a script that automatically analyzed over 500 Ecore metamodels.

Di Rocco [12] proposed a set of metrics to analyze metamodels. A focus of their study was computing the correlations between different metrics toward the identification of structural characteristics of metamodels. The metrics have been applied to a corpus of Ecore metamodels. They identified e.g., that *the adoption of inheritance is proportional to the size of metamodels* [12, p. 59].

Ma et al. [35] proposed a quality model for metamodels. The aim of their work was to provide guidance for researchers and practitioners on how to design metamodels of “good quality” by introducing the following quality attributes: *syntactic*, *semantic*, *pragmatic*, *capability*, and *evolvability*. Their approach remains on a theoretical level, contributing a research model that, based on questionnaires with 15 metamodel developers, quantifies the relationships between the quality parameters and the quality properties of a metamodel. Eventually, the authors apply the quality model to evaluate a set of evolutionary UML metamodels.

Recently, Lopez et al. [34] proposed 30 quality properties for metamodels, comprising the categories *design*, *best practice*, *naming conventions*, and *metric*. The focus was on measuring ex post the quality of a given metamodel. The metrics introduced by the authors establish some threshold values, e.g., for the number of direct children (10-max as default), mostly related from object-oriented design. The five metrics focus on coupling and inheritance aspects. The metrics have been applied to EMF metamodels.

The reviewed approaches all analyze EMF metamodels. This is not surprising, as up until recently, no corpus of metamodels developed with any other metamodeling platform was available. Consequently, the introduced metrics are also designed for EMF metamodels, omitting aspects of DSMLs like relations and modeltypes. The paper at hand extends the knowledge base by: i) establishing a framework to comprehensively analyze DSMLs; and ii) applying this framework to 40 DSML metamodels.

4 Metamodel Analysis Metrics

In the following, a novel analysis metrics framework is proposed targeting the comprehensive analysis of syntactic and structural aspects of DSMLs. The framework includes generic metamodel metrics found in literature and extends them in two ways: First and foremost, generic metrics for conceptual modeling methods. Second, some metrics specifically for ADOxx metamodels.

Table 2. Metamodel analysis metrics

Metric	Description
Generic metamodel metrics	
Concrete classes	The number of concrete classes
Abstract classes	The number of abstract classes
Attributes	The number of attributes
References	Number of references between two concepts
Inheritance	Maximal inheritance level
Conceptual Modeling-specific metamodel metrics	
Modeltypes	The number of modeltypes
Relation classes	The number of relation classes
ADOxx-specific metamodel metrics	
Dynamic modeltypes	The number of dynamic modeltypes
Static modeltypes	The number of static modeltypes
Dynamic classes	The number of dynamic classes
Static classes	The number of static classes

Generic metamodel metrics Analyzing the relevant literature [12,34,35,43], a set of recurring metamodel metrics can be identified (see Table 2). For these metrics, average values, min-max values, and statistical measures like median, quartiles, and standard deviation can be computed.

Conceptual Modeling-specific metamodel metrics The set of metrics in literature does not consider important characteristics of conceptual modeling methods. Relation classes are not considered explicitly but subsumed in the classes metric. Moreover, the decomposition of a modeling language into modeltypes is neglected. Consequently, corresponding metrics are introduced in Table 2, particularly addressing these shortcomings.

ADOxx-specific metamodel metrics In addition to the metrics described previously, meta class-specific metrics are introduced in order to enable a deeper analysis of the realization of DSMLs by means of the inheritance relationships to the predefined ADOxx meta classes (see Table 1). These metrics indicate the functionality utilized by a DSML and contribute toward externalizing the implicit design decisions made by the metamodel developer. Thus revealing the rationale behind metamodel designs (cf. [26]).

5 Analyzing Domain-Specific Metamodels

In the following, the metrics will be applied to 40 DSML metamodels. All metamodels have been realized within the OMiLAB using the ADOxx platform. Section 5.1 will first describe the research procedure followed while Sect. 5.2 reports on the key findings. Eventually, Sect. 5.3 compares the results with metrics of Ecore-based metamodels.

5.1 Preparing the Analysis

The analysis was aligned to extensively used literature survey methodologies [31]. However, instead of surveying articles, we surveyed metamodels. Thus, we followed a three-phased approach, comprising: 1. Planning, 2. Conducting, and 3. Analyzing. In the *planning phase*, we defined the research objectives. We were interested in empirically analyzing metamodels of DSMLs. Besides, we were also interested in how our results differ from Ecore metamodels. As a consequence, we chose the openly available metamodel repository of the OMiLAB as a source.

In the *conducting phase*, we queried the OMiLAB and collected 44 DSMLs metamodels. We then applied two exclusion criteria: Ex-1: the metamodel combines several completely independent modeling languages; and Ex-2: metamodels realized on an old version of ADOxx, these methods are neither maintained nor used anymore. In total, 4 metamodels matched the exclusion criteria, resulting in 40 metamodels which were analyzed in the *analysis phase* by applying the metamodel analysis metrics introduced in Sect. 4.

5.2 Results of the Analysis

The average number of modeltypes for a DSML is 7.15, whereas dynamic modeltypes following a graph-based structure are dominant with 6.68 compared to static ones following a tree-based structure with 0.48. All investigated DSMLs have at least one dynamic modeltype whereas only 40% have at least one static modeltype. The maximum number of modeltypes was found for LearnPAD [11] with 23 (22 + 1), followed by CuTiDe [5] with 21 (20 + 1), HORUS [40] with 19 (19 + 0), and MEMO4ADO [1] with 18 (18 + 0) modeltypes (dynamic + static), respectively. Fig. 3 illustrates the dominance of dynamic modeltypes.

Heterogeneous results were derived by looking at the number of classes. The average number of concrete classes is 49.23, with a median of 36. The maximum number of classes was found in CuTiDe [5] with 180 whereas the minimal number was found in the SERM [17] metamodel containing five classes. Abstract classes are used in 57.5% of the DSMLs, whereas the average number of abstract classes per metamodel is only 3.45 with a median of 1. CuTiDe has the most abstract classes (24). We found an average number of 20.2 relation classes, the median was 15. The most relation classes were found for the MEMO4ADO method [1] with 81, whereas the lowest number was found for PGA [39] and JCS [21] which both only contain one relation class. Fig. 4 visualizes analysis results for concrete, abstract, and relation classes of the DSML metamodels.

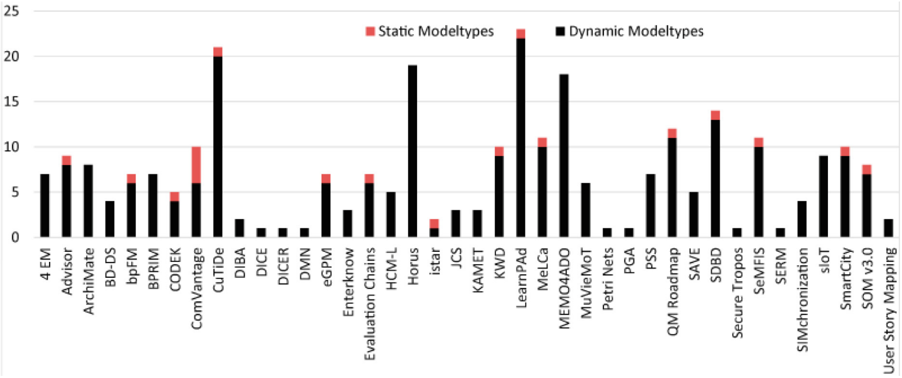

Fig. 3. Dynamic and static modeltypes per metamodel

In conceptual modeling, the majority of the semantics is encoded with the attributes of classes and relation classes. It is thus interesting to analyze, e.g., how many and which kind of attributes have been introduced as an indicator for the complexity of the domain to be addressed by the modeling method. Three kinds of attributes have been analyzed: *regular attributes*, e.g., of datatype string, integer, or boolean; *reference attributes*, enabling the creation of relationships between concepts within one or between different models; and *record table*

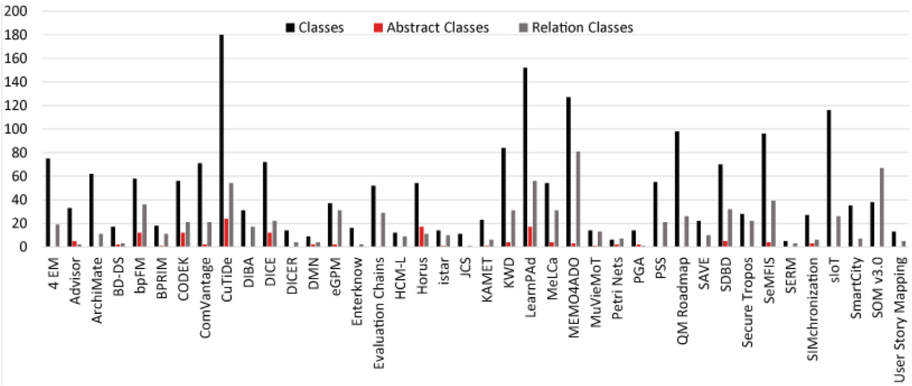

Fig. 4. Number of Concrete, Abstract, and Relation Classes per metamodel

attributes, used to create multi-dimensional attributes, i.e., tables. Finally, the DSMLs were analyzed for featureless classes - classes with no own attributes.

The average number of regular attributes is 11,76. The DICE method has the highest average amount of regular attributes per class (30.5), JCS has the lowest amount with 1.67. Reference attributes are used by 87.5% of the DSMLs with an average number of 45 per metamodel. By contrast, record table attributes are used by 67.5% of the DSMLs with an average number of only 8.75.

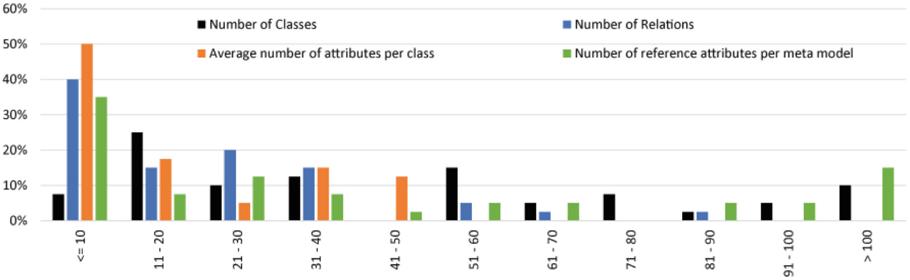

Fig. 5. Distribution of DSMLs based on classes, relations, attributes, and references

Besides the average and total numbers, it is also interesting to analyze the distribution of metrics criteria. We focus in the following on the most interesting ones due to limited space. Fig. 5 provides the results of grouping the DSMLs according to the total number of classes and relations, the average number of attributes per class, and the number of reference attributes per metamodel. It can be derived, that the biggest group of metamodels comprises 11 to 20 classes (25%), less than 10 relations (40%), and less than 10 references (35%). Moreover, in 50% of the metamodels, a class has in average less than 10 attributes. The majority of metamodels have less than 40 classes, less than 20 relations, less than 10 attributes, and less than 30 references.

Next, we investigated the inheritance relationships of the abstract and concrete classes. Within the analyzed metamodels, the abstract ADOxx meta class D_Aggregation was inherited from the most (in 67.5%), followed by D_Swimlane and S_Group from which was inherited from by 40% of the metamodels.

It can be derived from the complete analysis summarized in Table 3, that predefined abstract meta classes are more often used in the dynamic modeltypes compared to the static ones. Moreover, abstract classes for geographical containment (e.g., aggregation and swimlane) are used more frequently compared to simulation-specific classes like D_Event which was only inherited from by 10% of the DSMLs. Table 3 also provides for each applied metric, the total number of appearances in the 40 DSML metamodels and an average number, the maximal and minimal number of appearances, and the percentage of occurrences.

5.3 Comparison with Ecore-Based Metamodels

As mentioned in Sect. 3, related works exist that analyze Ecore-based metamodels. An assumption underlying both, Ecore and ADOxx metamodels is that the former primarily concentrates model-driven development and code generation whereas the latter primarily focuses on applying abstraction in order to create conceptual models for the purpose of communication and understanding by human beings [38]. Thus, we were interested in testing this hypothesis by comparing those metrics that are applicable to Ecore and ADOxx metamodels.

The results of this comparison are summarized in Table 4. The metamodel size is on average quite similar with 49.23 classes in ADOxx metamodels and 39.3 classes in Ecore. However, the median of ADOxx metamodels is almost 3 times higher compared to Ecore ones (36 compared to 13 classes). Abstract classes are almost equally used with 56% and 57.5%. Interestingly, Ecore metamodels differ significantly from ADOxx metamodels when analyzing the attributes and references. Ecore metamodels have a median of 13.5 references (ADOxx median is only 0.75), and a median of 8 attributes (ADOxx median is 7.5). They also differ with respect to the depth of the inheritance hierarchy. ADOxx metamodels have an average depth of 2.65 (Ecore: 5) and a maximal depth of 6 (Ecore: 10). The distribution of the size of the metamodels differs significantly. For ADOxx metamodels, only one third have less than 20 classes, whereas 69% of Ecore-based metamodels are this small.

It seems that the ADOxx based DSML metamodels are significantly larger compared to Ecore-based ones. This indicates, that the Ecore-based modeling languages are mostly designed for really narrow purposes which fits to the model-driven development domain. On the other hand, the usage of reference attributes is way more common in Ecore-based metamodels. This fact could be explained by the purpose of Ecore metamodels to act as structured data model. These references could solve referential integrity in the resulting data models.

Table 3. DSML metamodel metrics results

Metric	Total	Average per MM	Max	Min	Used by % of MM
Modeltype metrics					
Dynamic modeltypes	267	6.68	22	1	100 %
Static modeltypes	16	0.48	4	0	40 %
Classes and relation classes metrics					
Abstract classes	138	3.45	24	0	57.5 %
Concrete classes	1969	49.23	180	5	100 %
Dynamic classes	1782	44.55	169	5	100 %
Static classes	187	4.675	33	0	47.5 %
Relation Classes	808	20.2	81	1	100 %
Dynamic relation classes	655	16.38	81	1	100 %
Static relation classes	153	3.825	14	0	37.5 %
ADOxx-specific inheritance metrics					
D_Aggregation	80	2	8	0	67.5 %
D_Swimlane	41	1.03	8	0	40 %
D_Event	6	0.15	3	0	10 %
S_Group	44	1.1	4	0	40 %
S_Aggregation	15	0.38	2	0	35 %
S_Swimlane	22	0.55	2	0	27.5 %
S_Person	24	0.6	2	0	37.5 %
Attribute metrics					
Regular	27161	679.03	3411	14	100 %
References	1802	45.05	175	0	87.5 %
Record tables	350	8.75	59	0	67.5 %

Table 4. ADOxx vs. Ecore-based metamodel metrics

Metric	DSML metamodels	Ecore metamodels [43]
Average number of classes	49.23	39.3
Median number of classes	36	13
Max. number of classes	180	912
% metamodels using abstract classes	57.5 %	56 %
Median number of attributes per class	7.6	8
Median number of references per class	0.75	13.5
Average depth of inheritance	2.65	5
Max. depth of inheritance	6	10
Metamodels with <20 classes	33 %	69 %

6 Concluding Remarks and Future Work

To improve the development of new metamodels, analysis of existing ones seems promising. The paper at hand first introduced a generic metamodel analysis framework for analyzing conceptual modeling metamodels. This framework has then been applied to analyze 40 domain-specific conceptual modeling languages. Eventually, the results have been compared with Ecore-based metamodels.

It can be derived, that DSML metamodels are generally larger by nature considering the number of classes. When looking at the attributes, similarities and differences can be found. Ecore metamodels significantly more often use references, whereas the usage of regular attributes is almost equal. Moreover, Ecore metamodels have significantly deeper metamodel hierarchies.

As for any analysis, the results also have some threats to validity. All analyzed metamodels were realized with ADOxx. Thus, a platform bias is inevitable. Finally, it needs to be stated, that the Ecore metrics are based on a larger corpora of publicly available metamodels. Further application of the metrics need to verify completeness of the analysis framework and validity of the results.

From a practical perspective, the results indicate which concepts are actually used in DSMLs. It thus gives empirical insights into previously implicit metamodel design decisions and points metamodeling platform developers to aspects worthwhile for improvement - and others that can be lower prioritized.

We will prepare an open source webservice implementation of the metrics that will enable method engineers to apply the metrics to their metamodels by themselves. Moreover, we will now focus on identifying best practices and anti-patterns of metamodel design by investigating their quality impact. Moreover, research is left to be done in analyzing e.g., the metamodel domain, the communities developing the metamodels, and linguistic/semantic analysis of metamodels.

Acknowledgments. Part of this research has been funded through the South Africa / Austria Joint Scientific and Technological Cooperation program with the project number ZA 11/2017.

References

1. Bock, A., Frank, U.: Multi-perspective enterprise modeling—conceptual foundation and implementation with ADOxx. In: Karagiannis, D., Mayr, H.C., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, pp. 241–267. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39417-6_11
2. Bork, D., Sinz, E.J.: Design of a SOM business process modelling tool based on the ADOxx meta-modelling platform. In: Pre-proceedings of the 4th international workshop on graph-based tools. University of Twente, Enschede, pp. 90–101 (2010)
3. Bork, D.: Using conceptual modeling for designing multi-view modeling tools. In: 21st Americas Conference on Information Systems, AMCIS 2015, Puerto Rico, August 13–15 2015

4. Bork, D. et al.: Requirements engineering for model-based enterprise architecture management with ArchiMate. In: Enterprise and Organizational Modeling and Simulation, 14th International Workshop, EOMAS 2018, Held at CAiSE 2018, Tallinn, Estonia (2018), in press
5. Bork, D., Karagiannis, D., Hawryszkiewycz, I.T.: Supporting customized design thinking using a metamodel-based approach. In: Proceedings of the Australasian Conference on Information Systems (ACIS), Hobart, Australia (2017)
6. Bork, D., Karagiannis, D., Pittl, B.: How are Metamodels specified in practice? empirical insights and recommendations. In: Twenty-fourth Americas Conference on Information Systems, pp. 1–10 (2018)
7. Bork, D., Karagiannis, D., Pittl, B.: Systematic analysis and evaluation of visual conceptual modeling language notations In: 2018 12th International Conference on Research Challenges in Information Science (RCIS), pp. 1–11. IEEE (2018)
8. Bork, D., Miron, E.T.: OMiLAB - an open innovation community for modeling method engineering. In: Niculescu, A., Negoita, O.D., Tiganoaia, B. (eds.) 8th International Conference of Management and Industrial Engineering (ICMIE 2017), pp. 64–77 (2017). http://eprints.cs.univie.ac.at/5145/
9. Cairó, O., Guardati, S.: The KAMET II methodology: knowledge acquisition, knowledge modeling and knowledge generation. Expert Syst. Appl. **39**(9), 8108–8114 (2012)
10. Cairó Battistutti, O., Bork, D.: Tacit to explicit knowledge conversion. Cognit. Process. **18**(4), 461–477 (2017)
11. De Angelis, G., Pierantonio, A., Polini, A., Re, B., Thönssen, B., Woitsch, R.: Modeling for learning in public administrations—the learn PAd approach. In: Karagiannis, D., Mayr, H., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, pp. 575–594. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39417-6_26
12. Di Rocco, J., Di Ruscio, D., Iovino, L., Pierantonio, A.: Mining Metrics for understanding metamodel characteristics. In: Proceedings of the 6th International Workshop on Modeling in Software Engineering, pp. 55–60. MiSE 2014, ACM, New York, NY, USA (2014)
13. Efendioglu, N., Woitsch, R., Utz, W.: A toolbox supporting agile modelling method engineering: ADOxx.org modelling method conceptualization environment. In: Horkoff, J., Jeusfeld, M.A., Persson, A. (eds.) PoEM 2016. LNBIP, vol. 267, pp. 317–325. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-48393-1_23
14. Ferstl, O.K., Sinz, E.J., Bork, D.: Tool support for the semantic object model. In: Karagiannis, D., Mayr, H., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, pp. 291–310. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39417-6_13
15. Fill, H.G., Karagiannis, D.: On the conceptualisation of modelling methods using the ADOxx meta modelling platform. Enterp. Modell. Inf. Syst. Architect. **8**(1), 4–25 (2013)
16. 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, Heidelberg (2013)
17. Glässner, T.M., Heumann, F., Keßler, L., Härer, F., Steffan, A., Fill, H.G.: Experiences from the implementation of a structured-entity-relationship modeling method in a student project. In: Bork, D., Karagiannis, D., Vanthienen, J. (eds.) Proceedings of the 1st International Workshop on Practicing Open Enterprise Modeling within OMiLAB (PrOse 2017). CEUR Proceedings (2017)
18. Guizzardi, G.: Ontological foundations for structural conceptual models. CTIT, Centre for Telematics and Information Technology (2005)

19. Guizzardi, G., Herre, H., Wagner, G.: On the general ontological foundations of conceptual modeling. In: Spaccapietra, S., March, S.T., Kambayashi, Y. (eds.) ER 2002. LNCS, vol. 2503, pp. 65–78. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45816-6_15
20. Gupta, H.V., Clark, M.P., Vrugt, J.A., Abramowitz, G., Ye, M.: Towards a comprehensive assessment of model structural adequacy. Water Resour. Res. **48**(8) (2012)
21. Hara, Y., Masuda, H.: Global service enhancement for japanese creative services based on the early/late binding concepts. In: Karagiannis, D., Mayr, H., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, pp. 509–526. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39417-6_23
22. Hawryszkiewycz, I.T., Prackwieser, C.: MELCA—customizing visualizations for design thinking. In: Karagiannis, D., Mayr, H., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, pp. 383–396. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39417-6_17
23. Hinkel, G., Kramer, M., Burger, E., Strittmatter, M., Happe, L.: An empirical study on the perception of metamodel quality. In: 2016 4th International Conference on Model-Driven Engineering and Software Development (MODELSWARD), pp. 145–152. IEEE (2016)
24. Houy, C., Fettke, P., Loos, P.: Understanding understandability of conceptual models – what are we actually talking about? In: Atzeni, P., Cheung, D., Ram, S. (eds.) ER 2012. LNCS, vol. 7532, pp. 64–77. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-34002-4_5
25. Houy, C., Fettke, P., Loos, P.: On the theoretical foundations of research into the understandability of business process models. In: Avital, M., Leimeister, J.M., Schultze, U. (eds.) 22st European Conference on Information Systems, ECIS 2014, Tel Aviv, Israel, June 9–11, 2014 (2014). http://aisel.aisnet.org/ecis2014/proceedings/track06/7
26. Izquierdo, J.L.C., Cabot, J.: Collaboro: a collaborative (meta) modeling tool. Peer J. Computer. Sci. **2**, e84 (2016)
27. Karagiannis, D.: Agile modeling method engineering. In: Proceedings of the 19th Panhellenic Conference on Informatics, pp. 5–10. ACM (2015)
28. Karagiannis, D., Kühn, H.: Metamodelling platforms. In: Bauknecht, K., Tjoa, A.M., Quirchmayr, G. (eds.) EC-Web 2002. LNCS, vol. 2455, pp. 182–182. Springer, Heidelberg (2002). https://doi.org/10.1007/3-540-45705-4_19
29. Karagiannis, D., Mayr, H.C., Mylopoulos, J.: Domain-Specific Conceptual Modelling. Springer, Heidelberg (2016)
30. Kern, H., Hummel, A., Kühne, S.: Towards a comparative analysis of meta-metamodels. In: Proceedings of the Compilation of the Co-located Workshops on DSM'11, TMC'11, AGERE! 2011, AOOPES'11, NEAT'11, & VMIL 2011, pp. 7–12. ACM (2011)
31. Kitchenham, B., Brereton, P.: A systematic review of systematic review process research in software engineering. Inf. Softw. Technol. **55**(12), 2049–2075 (2013)
32. Koschmider, A., Figl, K., Schoknecht, A.: A comprehensive overview of visual design of process model element labels. In: Reichert, M., Reijers, H.A. (eds.) BPM 2015. LNBIP, vol. 256, pp. 571–582. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-42887-1_46
33. Langer, P., Mayerhofer, T., Wimmer, M., Kappel, G.: On the usage of UML: initial results of analyzing open UML models. In: Fill, H., Karagiannis, D., Reimer, U. (eds.) Modellierung 2014, pp. 289–304. GI (2014)

34. López-Fernández, J.J., Guerra, E., de Lara, J.: Assessing the quality of metamodels. In: Boulanger, F., Famelis, M., Ratiu, D. (eds.) Proceedings of the 11th Workshop on Model-Driven Engineering, Verification and Validation, MoDeVVa@MODELS 2014, pp. 3–12 (2014)
35. Ma, Z., He, X., Liu, C.: Assessing the quality of metamodels. Front. Comput. Sci. **7**(4), 558–570 (2013)
36. Michael, J., Mayr, H.C.: the process of creating a domain specific modelling method (Extended Abstract). In: Mendling, J., Rinderle-Ma, S. (eds.) Proceedings of the 7th International Workshop on Enterprise Modeling and Information Systems Architectures, EMISA 2016. vol. 1701, pp. 40–43. CEUR-WS.org (2016)
37. Moody, D.: The "physics" of notations: toward a scientific basis for constructing visual notations in software engineering. IEEE Trans. Softw. Eng. **35**(6), 756–779 (2009)
38. Mylopoulos, J.: Conceptual modelling and Telos. In: Loucopoulos, P., Zicari, R. (eds.) Conceptual Modelling, Databases, and CASE: an Integrated View of Information System Development, pp. 49–68. Wiley, New York (1992)
39. Roelens, B., Steenacker, W., Poels, G.: Realizing strategic fit within the business architecture: the design of a process-goal alignment modeling and analysis technique. Softw. Syst. Model. (2017)
40. Schoknecht, A., Vetter, A., Fill, H.-G., Oberweis, A.: Using the horus method for succeeding in business process engineering projects. In: Karagiannis, D., Mayr, H., Mylopoulos, J. (eds.) Domain-Specific Conceptual Modeling, pp. 127–147. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39417-6_6
41. Stark, J., Braun, R., Esswein, W.: Systemizing colour for conceptual modeling. In: Leimeister, J.M., Brenner, W. (eds.) Proceedings der 13. Internationalen Tagung Wirtschaftsinformatik (WI 2017), St. Gallen, pp. 256–270 (2017)
42. Wand, Y., Weber, R.: On the ontological expressiveness of information systems analysis and design grammars. Inf. Syste. J. **3**(4), 217–237 (1993)
43. Williams, J.R. et al.: What do metamodels really look like? Eessmod@ Models **1078**, 55–60 (2013)