On Languages for Conceptual Data Modeling in Multi-disciplinary Space Systems Engineering

Christian Hennig[1], Harald Eisenmann[2], Alexander Viehl[1] and Oliver Bringmann[3]
[1]*Intelligent Systems and Production Engineering, FZI Research Center for Information Technology, Karlsruhe, Germany*
[2]*TSOEC3, Space Systems, Airbus Defence and Space, Friedrichshafen, Germany*
[3]*Wilhelm-Schickard-Institute for Computer Science, Eberhard-Karls-University of Tübingen, Tübingen, Germany*

Keywords: Systems Engineering, Model-based Systems Engineering, Conceptual Data Modeling, UML, Ecore, OWL, ORM.

Abstract: The engineering of complex systems is more and more supported through computer-based models that rely on a comprehensive specification of their underlying data. This paper reflects on extensive industrial experience with a sophisticated application of conceptual data modeling, addressing requirements as they arise in the context of space systems engineering. For this purpose identified needs for conceptual data modeling in the scope of Model-Based Systems Engineering are formulated. Established and evolving approaches and technologies for building conceptual data models are characterized, analyzed, and discussed regarding their suitability for modeling engineering data. Based on this analysis of the state of the art, recommendations for the future evolution of conceptual data modeling are formulated.

1 INTRODUCTION

Building systems such as satellites, launch vehicles or other spacecraft requires the interaction of numerous engineering disciplines. Each of these disciplines, such as mechanical engineering, software engineering, or verification engineering, uses their very own computer-based models such as CAD models, UML models, or verification matrices. In the scope of Model-based Systems Engineering (MBSE) a tendency can be observed to integrate critical pieces of data from all of these models in a central engineering database. One fundamental motivation for this evolution towards digitally shared models is the ability to interlink the domain-specific models in order to enable early multi-disciplinary analyses, early verification and validation, and finding model inconsistencies. For integrating these models one approach that can be pursued is the introduction of a central system database, containing a system model. For specifying the engineering concepts that are contained inside the system model, a conceptual data model (CDM) is used. The CDM, in this context, provides a common, resilient, and comprehensive definition of engineering data, incorporating discipline-specific as well as system-level aspects.

A variety of approaches exist for building such models. On the one hand there are approaches strongly driven by the implementation technologies that are used for producing engineering applications, relying on data models specified in UML or Ecore. On the other hand there are techniques almost exclusively focused on representing knowledge that can also be used to specify data, such as the Web Ontology Language OWL or Object Role Modeling ORM. Each of these approaches has its own characteristics with both shortcomings and unique benefits.

Figure 1 consolidates the setting detailed in this paragraph using a couple of examples. The bottom

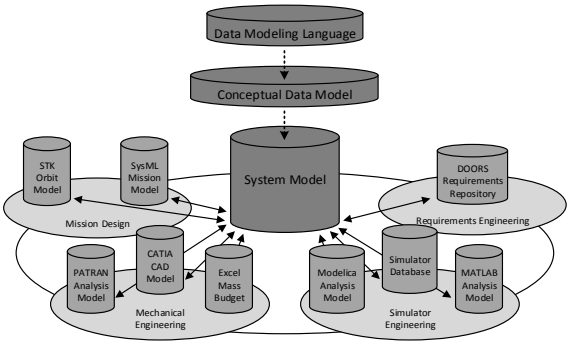

Figure 1: The relation between domain-specific models, system model, CDM, and data modeling language.

Hennig C., Eisenmann H., Viehl A. and Bringmann O..
On Languages for Conceptual Data Modeling in Multi-disciplinary Space Systems Engineering.
DOI: 10.5220/0005329003840393
In *Proceedings of the 3rd International Conference on Model-Driven Engineering and Software Development* (MODELSWARD-2015), pages 384-393
ISBN: 978-989-758-083-3

level describes the system model or user model along with domain-specific models that perform some kind of information exchange. For specifying the concepts that make up the system model a CDM is used, forming the system model's metamodel. This CDM is described using a data modeling language.

In the following sections, selected languages for conceptual data modeling are evaluated. Industrial requirements on such a language that arise in the context of MBSE are identified and an examination regarding how well each of these languages is able to fulfill the functions specified in the requirements is performed. Based on this examination, an overview of the state of the art in conceptual data modeling for MBSE is sketched, highlighting present shortcomings, and a proposal for future evolution is given.

2 BACKGROUND

This section explains the nature of systems engineering, its relation to models, and how system-wide models are described in the context of MBSE.

2.1 Systems Engineering

In many industrial engineering projects today, a multitude of disciplines is involved in building the systems of interest. For space projects such as satellites, launch vehicles, and resupply spacecraft these disciplines involve, only to name a few, mechanical engineering, electrical engineering, thermal engineering, requirements engineering, software engineering, verification engineering, and their respective sub-disciplines. Each of these disciplines specifies their designs and verifies specific aspects of the system. In order to provide an all-encompassing understanding of the system of interest, the unique, yet complementary views from every involved discipline are combined. The science and art of integrating different views on one system towards system thinking is called Systems Engineering. As NASA (2007) elegantly puts it: "Systems engineering is a holistic, integrative discipline, wherein the contributions of structural engineers, electrical engineers, mechanism designers, power engineers, human factors engineers, and many more disciplines are evaluated and balanced, one against another, to produce a coherent whole that is not dominated by the perspective of a single discipline."

2.2 Systems Engineering and Models

Many of the engineering activities performed inside these discipline domains are already well supported by computer-based models. Mechanical design models built with tools such as CATIA V5 (Dassault Systemes, 2014), mechanical analysis models built with tools such as PATRAN (MSC Software, 2014) and thermal analysis models built with tools such as ESATAN-TMS (ITP Engines UK, 2014) are well established in the space engineering community today. Furthermore, there are requirements models based on DOORS, software design models specified in the Ecore language (Eclipse Foundation, 2014) using the Eclipse Modeling Framework (Eclipse Foundation, 2014), as well as mission design models specified in SysML (OMG, 2012) with tools such as MagicDraw (No Magic, 2014) Furthermore, "traditional" tools such as Excel or Visio are used on a regular basis for building models. The practice of supporting engineering activities with models is called Model-based Engineering (MBE) or Model-driven Engineering (MDE).

These tools and the models they produce differ significantly from each other. They are provided by different vendors, rely on different implementation technologies and are based on different formats (Kogalovsky and Kalinichenko, 2009). Each model and the associated design methodology follow their own principles and paradigms and define their very own semantics. As a result of this heterogeneity, these models and tools are not yet integrated with each other and with the multi-domain systems engineering process (INCOSE, 2014). For a truly multi-disciplinary representation of a system, relevant aspects from all involved domains and their models have to be combined on the model level (Eisenmann, 2012).

2.3 Describing System-wide Models

A computer-based model consists of two basic parts. The layer directly visible to the user is the instance model or user model, where the user enters his data and works with it. In order to specify what bits of information can be described in the user model, a data model or metamodel is required that specifies the concepts of the user model (Hong and Maryanski, 1990). It is worthy to note at this point that metamodel is a relative term. It describes concepts one abstraction level above the model that is currently the focus of interest.

2.3.1 The System Model

For such models in engineering the "working level" is represented by the system model or user model. In this model the system of interest is described. This includes domain-specific aspects of the system and the data relevant to systems engineering activities. Examples for such data are all requirements that are specified for the system, its logical decomposition, or the entirety of the functions the system performs.

2.3.2 The Conceptual Data Model

In order to be able to specify the system model, the concepts that make it up have to be specified somehow. This happens in the CDM, forming the metamodel of the user model. In this model, the definition about what a requirement is, how requirements relate to other requirements, how requirements relate to the system decomposition and how the system structure relates to system functions is taking place, for example. All in all the CDM describes the entities, conceptual structures, and characteristic relationships of the Universe of Discourse (UoD) (Kogalovsky and Kalinichenko, 2009) (Halpin and Morgan, 2008).

A significant amount of information exchange occurs between the system model and existing discipline-specific models. Consequently the system model's metamodel, the CDM, specifies how these models interface with each other. The approach of using a CDM to interface between domain models on the one side and physical databases on the other side has already been described in 1975 in the interim report of the ANSI/X3/SPARC Study Group on Data Base Management Systems (1975). The CDM in this context is used for specifying the concepts that come from the domains, interfacing with the domain disciplines and forming the basis for an implementation of a system-wide database. Consequently, the conceptual data model can be seen as the backbone of MBSE (Eisenmann, 2012).

It is worthy to note that the currently predominant approach in most engineering domains is to exchange knowledge between all discipline-specific models in a document-based fashion. This means that the knowledge stored in a computer model of a specific domain is written in a document which is then handed to another domain. Engineers from the other domain then extract their required bits of information from the document and employ it accordingly. It is evident that this document-based exchange of information is a tedious process prone to errors and inconsistencies, often resulting in a significant amount of unnecessary overhead. Consequently, a large tendency to support such engineering processes with models, making the information accessible in an automated way, can be observed. It is expected that model-based information exchange significantly reduces the effort and consequently the costs involved in inter-disciplinary and inter-domain information exchange. Moreover, engineering processes relying on MBSE are expected to benefit from improved quality, increased productivity, and reduced risk (Friedenthal, et al., 2009).

2.3.3 The Data Modeling Language

Being the center of MBSE-based activities the CDM can be specified in a number of languages. For developing relational databases, the conceptual model is often specified in Entity–relationship models (Halpin and Morgan, 2008) or MS Access database schemas. For furthering tool integration, the EXPRESS language (ISO, 2004) was developed. Other approaches directly rely on languages that are usually employed for specifying software, such as UML or Ecore (Kogalovsky and Kalinichenko, 2009) (Olivé, 2007) while knowledge-focused modeling languages such as Gellish (Van Renssen, 2005) Object Role Modeling (ORM) models (Halpin and Morgan, 2008) and the Web Ontology Language (OWL) (W3C, 2012) have also been employed for specifying a wide variety of UoDs.

Some of those languages did not deliver the hoped for results (EXPRESS), others meanwhile reached their limits (ER, Access, Gellish, UML) while yet others are rather gaining momentum than losing ground (Ecore, ORM, OWL) in the context of MBSE.

3 TECHNOLOGY EVALUATION

In this section requirements on a CDM language are formulated, and selected modeling languages are evaluated against a defined evaluation scheme.

3.1 Requirements on a CDM Language

The requirements on languages for conceptual data modeling explained in this section are based upon extensive experience employing CDMs in the context of MBSE. They incorporate a large amount of facets derived from lessons learned in past projects (ESA, 2012) (ESA, 2013) (Fischer, et al., 2014). These requirements have been partitioned into five categories. Explaining every required function of each category in equal detail would go beyond the

scope of this paper. Instead, an emphasis is given on specific, not so obvious, or difficult to understand language functions and properties. These categories and requirements are listed in **Error! Reference source not found.**.

3.1.1 Semantic Relevance for MBSE

In this category very specific requirements that arise from using the modeling language in the scope of MBSE are consolidated.

Ability to Work with Closed World Facts: One specific requirement in this context is the ability of the language to produce models based upon the Closed World Assumption. This principle implies that everything that is not explicitly stated as true is false. For instance, if a model of a satellite would state that the satellite has mounted four solar arrays on it and a query would be made, asking for the number of solar arrays, the result would be "four". In contrast to the Closed World Assumption stands the Open World Assumption that is common for ontology languages. The same query asking for the number of solar arrays would return the result "unknown" since it is not explicitly stated that there are exactly four solar arrays. It is evident that such behavior can become problematic when producing engineering models (Hennig, 2012). Some ontology extensions allow "closing down" such facts, enabling the correct answer of such queries, but resulting in a significant amount of additional modeling effort (Mehdi and Wissmann, 2013).

Understandability of Language for Non-modeling Experts: CDMs are often produced in accordance with experts of the UoD. These people are mostly mechanical engineers, electrical engineers, or software engineers, not modeling experts. Consequently it is desirable for the modeling language to be able to be understood by non-modeling experts.

Ability to Separate between Tool-development and Tool-customization Concepts: A common activity in space engineering is the practice of tailoring. Tailoring means that an existing model, procedure, or application is being customized in order to exactly fit the needs of a specific project or mission. In case of applications based on a CDM, that would imply adapting the CDM. However, especially for larger projects, it is often not feasible to alter the CDM to every mission that is conducted, since it would result in also altering every application built upon it. Consequently, solutions have been proposed that introduce a secondary part to the CDM that is used for tailoring, being able to be adapted during model runtime (ESA, 2012) (ESA, 2013).

Ability to Integrate Knowledge from other Resources and existence of a Transformation Mechanism: Since one of the focal points of CDMs is information exchange between different models, it should also be possible to map different CDMs to each other, i.e. to integrate another CDM as a resource, including model transformations, should they become necessary after the mapping is defined.

Existence of a Language extension Mechanism: For introducing custom concepts to the language, some kind of extension mechanism is highly desirable (Eisenmann, 2012).

3.1.2 Adequacy for Developing MBSE Applications

The second category deals with the aspects related to producing pieces of software that implement the CDM, as it is commonly done in MBSE. One example for such an application is an engineering database intended for storing the system model.

Independence from Implementation Technology: In order to ensure maximum flexibility for implementing applications it is desirable for the CDM language to be independent from any implementation technology.

Low effort to get from CDM to Software Structure Specification: If the language used for modeling the CDM is suited for modeling software, specifying a piece of software is less effort compared to a case where the CDM is specified in a language that cannot be used for describing software.

Suitability for Code Generation: Code generation has shown to significantly reduce development costs (Eisenmann, 2012) and is seen as a key element for producing software intended for MBSE.

3.1.3 Adequacy for MBSE Data Modeling

A number of activities are usually performed when producing CDMs. Such activities include deriving the CDM from an engineering process, producing a specification, and validating the CDM.

Support by a CDM Engineering Method: One requirement for a data modeling language is that it is supported by a method that helps in knowledge acquisition, configuration management, knowledge integration, and similar activities guiding the modeler through the process of building the CDM. These methods are common in the area of knowledge engineering (Halpin and Morgan, 2008) (Sure, et al., 2004) (Suárez-Figueroa, 2010).

Ability to Describe Process Activities and Process Artefacts: In model-based space systems

engineering a convenient starting point for building the CDM of a UoD is examining the engineering processes it should support. The CDM has to be able to represent the data of the input and output artefacts to the process activities (Hennig and Eisenmann, 2014). Such processes are usually documented in UML activity diagrams or BPMN diagrams. It is of benefit for the modeling language to provide means for modeling the engineering process and its artefacts together with the CDM and to provide mappings between the process artefacts and the elements of the CDM, bridging the gap between both models.

Ability to Provide Validation Instances: Another requirement is the ability to cultivate validation instances while the CDM is still in production, aiding the modeler by providing examples or even letting the modeler create own examples. Using examples together with the model has been shown to make the CDM more tangible (Halpin and Morgan, 2008).

Suitability for Verbalization: Verbalization of model elements is the process of representing knowledge stored in the model available in a sentence. These sentences are usually formed using a subset of a natural language, e.g. English, made up by a controlled vocabulary and syntax (Halpin and Wijbenga, 2010).

Employment of Surrogate Names for Relations: Some modeling languages require a unique name for each relation between model entities. This can become problematic once a lot of relations exist or once a number of commonly occurring relations is used. For instance a satellite might have the relations “has Designation”, “has UniqueID”, and “has Abbreviation”, whereas “has” is the reference and occurs multiple times. Although “has” is not a well-formed relation in many cases, it is used frequently. For circumventing this problem, a surrogate name can be used for describing relations, e.g. “SatelliteHasDesignation” as unique name for the reference, but only displaying “has” to the user, acting as a surrogate for the actual reference.

3.1.4 Richness of Data Structures

Data structures form the central building blocks of a CDM. A modeling language for MBSE should be able to describe elements such as classes, attributes, and data types. The most basic relations between classes are binary relationships. Sometimes ternary relationships are needed for describing elementary facts of the UoD and some languages even support n-ary relationships between classes. It might become necessary to objectify a relation between classes for the purpose of handling it more like a class instead of a relation later on. Furthermore the data modeling language has to be able to modularize the data model, i.e. dividing it up into several packages and subpackages for the purposes of structuring and enabling de-coupled definition with subsequent integration. The language should also be able to specify an explicit hierarchical structure apart from the taxonomic structure of classes.

3.1.5 Richness of Constraint Structures

The most basic category of constraints are the cardinalities of attributes and references that specify whether e.g. a reference is mandatory or optional, and how many references of one type can exist at the same time. Other constraints related to references include subsets (a reference can only exist if it is already represented by another specific reference) exclusion constraints (if reference A exists in the user model, reference B cannot exist), equality constraints (if reference A exists then reference B must also exist), or ring constraints (for cyclic references, e.g. a reference cannot reference the same instance it originated from). Some languages support more specific constraints, e.g. a maximum limit on the number of instances that can exist of one class at the same time, and some set theory constraints between subtypes of classes. Some languages also permit the specification of entirely user-defined consistency checks.

3.2 Definition of an Evaluation Scheme

The evaluation presented in Table 1 is based upon identified necessities derived from past experience using CDMs in the context of MBSE. The analysis employs weighted score evaluation and consists of two dimensions. The first dimension is made up of the requirements on the data modeling language. These requirements are assigned a weight (column “W”) with a value of either 1, 3, or 9, determining how important the requirement is in the context of MBSE for space systems. A 1 states that the feature stated in the requirement is nice to have, a 3 means that the feature is definitively helpful while 9 marks a critical feature. The second dimension determines how well each of the modeling languages can cope with the feature stated in the requirement. The language columns can take values of 0 (not possible/not available/not publicly documented), 1 (possible, but with limitations), and 3 (fully supported). The language evaluation consists of two columns for each language. The first column contains the score, the

second column contains the product of score and requirement weight.

In the row of each category the maximum attainable score is given, along with the total score of the language in said category.

While the detailed and comprehensive evaluation is presented in Table 1. The paragraphs below explain some specifics of the evaluation that might not directly be traceable without explanation

3.3 Evaluation of Selected Languages

While all data modeling languages outlined previously in 2.3.3 are using the same basic building blocks (classes, attributes, and references, although sometimes named differently) for specifying knowledge structures at first sight, they exhibit entirely different nuances when closer examined. For closer examination UML, Ecore, ORM, and OWL have been selected.

3.3.1 The Unified Modeling Language UML

UML has been established as the de-facto standard for describing software systems for quite some time and forms the current mainstream approach for data modeling (Kogalovsky and Kalinichenko, 2009) (Olivé, 2007).

UML models by default are based on the Closed World Assumption (3 points). However, data models in UML are considered difficult to understand for people not trained in software design with regard to specifying knowledge (0 points). An extension mechanism is given by introducing custom stereotypes through profiling, but has been shown to be not sufficient for many data modeling efforts in MBSE (Eisenmann, 2012) (1 point). A lot of methods exist for producing pieces of software with UML, but these methods do not provide guided, prescriptive instructions for modeling the data structure of the UoD and do not cater to the needs of CDMing in MBSE (1 point). Processes can be modeled using UML activity diagrams, but this approach also does not completely fit the needs for MBSE (1 point). UML contains all of the basic data structures, can objectify relations using association classes, modularize models through packages, and describe explicit hierarchical structures using the composition and aggregation elements. UML has a limited number of constraints built in, such as the cardinalities of associations and subsets between associations. Other constraints, such as exclusion constraints, equality constraints, or ring constraints, are not directly built-in and have to be asserted using OCL. However, OCL, due to its highly technical nature, is not desirable for modeling domain knowledge, only yielding a value of 1 point for each these constraints.

3.3.2 Ecore

The Eclipse Modeling Framework EMF is gaining considerable momentum in the area of model-driven software engineering. EMF comes with a dedicated modeling language called Ecore that picks up some model elements of UML. The capability for code generation, an agile software reuse process and sophisticated tool support have shown to considerably reduce costs for developing data model-intensive applications while still retaining high functionality and flexibility (Eisenmann, 2012).

Ecore models are also by default based upon the Closed World Assumption (3 points) and require equal technical understanding as UML models (0 points). A tailoring mechanism does also not exist (0 points). Similar to UML Ecore supports integrating a multitude of resources and performing model transformations (3 points each). Ecore is highly dependent on the Eclipse Modeling Framework for implementing software (0 points). The effort to get from the CDM to the specification of the software implementing the CDM is equally manageable compared to UML. Code generation is one of the main selling points of EMF and Ecore (3 points). Ecore focuses solely on modeling structures and consequently does not model behavior such as processes (0 points). A limited mechanism for providing validation instances exist by creating dynamic instances, but still needs considerable improvement to cater to the needs of MBSE (1 point). Ecore can describe classes, attributes, data types, and pack-ages (3 points each). An explicit hierarchical structure can be specified using the containment property of references (3 points). Ecore supports binary relations (3 points), but none of higher arity (0 points). Objectification is also not directly sup-ported (0 points). Specification of constraints is similar to UML with the exception that the subset constraint is not directly built into Ecore (1 point), relying on OCL for specifying such semantically rich constraints.

3.3.3 Object Role Modeling ORM

For designing relational databases a data modeling language called ORM (Halpin and Morgan, 2008) has gained importance. This language offers a wide variety of built-in constraints, such as subset constraints, uniqueness constraints, and constraints regarding possible combinations of model entities. ORM is a kind of Fact Based Modeling (Halpin and Morgan, 2008). Due to the focus on business

Table 1: Evaluation of Conceptual Data Modeling Languages.

ID	Data Modeling Language Requirement	W	UML		Ecore		ORM		OWL	
1	Semantic Relevance of Modeling Language for MBSE	162		69		87		78		69
1.1	Ability to work with closed world facts	9	3	27	3	27	3	27	1	9
1.2	Understandability of language for non-modeling experts	9	0	0	0	0	3	27	1	9
1.3	Ability to represent aspects in a granular manner	3	0	0	0	0	0	0	3	9
1.4	Ability to ensure the logical coherence of aspects	3	1	3	1	3	3	9	3	9
1.5	Ability to specify business rules of the UoD	3	1	3	1	3	1	3	1	3
1.6	Ability to separate between tool-dev. and -customization concepts	9	0	0	0	0	0	0	0	0
1.7	Ability to integrate knowledge from other resources	3	3	9	3	9	0	0	3	9
1.8	Existence of transformations of the CDM into other languages	3	3	9	3	9	1	3	1	3
1.9	Existence of mechanism for specifying queries on the user model	3	3	9	3	9	3	9	3	9
1.10	Existence of a language extension mechanisms	9	1	9	3	27	0	0	1	9
2	Adequacy for Developing MBSE Applications	63		45		54		3		12
2.1	Independence from implementation technology	3	3	9	0	0	0	0	3	9
2.2	Low effort to get from CDM to software structure specification	3	3	9	3	9	1	3	1	3
2.3	Suitability for code generation	9	1	9	3	27	0	0	0	0
2.4	Tight integration of language into tool chain	3	3	9	3	9	0	0	0	0
2.5	Free availability of end-to-end development tool chain	3	3	9	3	9	0	0	0	0
3	Adequacy for MBSE Data Modeling Activities	111		31		19		55		33
3.1	Support by a CDM engineering method	9	1	9	1	9	3	27	1	9
3.2	Ability to describe process activities and process artefacts	9	1	9	0	0	0	0	0	0
3.3	Support for data model specification and verification activities	9	0	0	0	0	0	0	0	0
3.4	Ability to provide validation instances	3	3	9	1	3	3	9	3	9
3.5	Ability to propagate data model changes to the user model	1	1	1	1	1	1	1	3	3
3.6	Suitability for verbalization	3	1	3	1	3	3	9	1	3
3.7	Employment of surrogate names for relations	3	0	0	1	3	3	9	3	9
4	Richness of Data Structures	159		159		144		132		108
4.1	Ability to describe classes	9	3	27	3	27	3	27	3	27
4.2	Ability to describe attributes	9	3	27	3	27	3	27	3	27
4.3	Ability to describe data types	9	3	27	3	27	3	27	3	27
4.4	Ability to describe binary relations	9	3	27	3	27	3	27	3	27
4.5	Ability to describe ternary relations	3	3	9	0	0	3	9	0	0
4.6	Ability to describe n-ary relations	1	3	3	0	0	3	3	0	0
4.7	Ability to objectify relations	1	3	3	0	0	3	3	0	0
4.8	Ability to modularize the data model	3	3	9	3	9	0	0	3	9
4.9	Ability to describe an explicit hierarchical structure	9	3	27	3	27	1	9	0	0
5	Richness of Constraint Structures	213		155		149		177		178
5.1	Ability to specify internal uniqueness	9	3	27	3	27	3	27	3	27
5.2	Ability to specify external uniqueness	1	1	1	1	1	3	3	0	0
5.3	Ability to specify simple mandatory roles	9	3	27	3	27	3	27	3	27
5.4	Ability to specify internal frequency constraints	9	3	27	3	27	3	27	3	27
5.5	Ability to specify external frequency constraints	1	1	1	1	1	3	3	3	3
5.6	Ability to specify subsets of relations	3	3	9	1	3	3	9	3	9
5.7	Ability to specify sub-relation chains	3	1	3	1	3	0	0	3	9
5.8	Ability to specify exclusion constraints	3	1	3	1	3	3	9	3	9
5.9	Ability to specify exclusive or constraints	1	1	1	1	1	3	3	3	3
5.10	Ability to specify inclusive-or constraints	3	1	3	1	3	3	9	3	9
5.11	Ability to specify equality constraints	1	1	1	1	1	3	3	1	1
5.12	Ability to specify ring constraints	3	1	3	1	3	3	9	3	9

Table 1: Evaluation of Conceptual Data Modeling Languages (Cont.).

ID	Data Modeling Language Requirement	W	UML		Ecore		ORM		OWL	
5.13	Ability to specify subtype constraints	9	1	9	1	9	3	27	1	9
5.14	Ability to specify object cardinality constraints	3	1	3	1	3	3	9	0	0
5.15	Ability to specify role cardinality constraints	1	1	1	1	1	3	3	0	0
5.16	Ability to specify user-defined constraints	9	3	27	3	27	0	0	3	27
5.17	Ability to specify textual constraints	3	3	9	3	9	3	9	3	9

modeling ORM is built with the principle in mind that the model and the information stored in it should be read in a “natural way”, enabling the domain experts that model and the information stored in it should be read in a “natural way”, enabling the domain experts that are usually not modeling experts to understand and work with the model.

Models specified using Object Role Modeling are also based upon the Closed World Assumption (3 points). The language was designed with readability and understandability for non-modeling experts in mind (3 points). A tailoring mechanism as required for MBSE does not exist (0 points). The same is true for a mechanism to integrate different ORM resources towards one CDM (0 points). Some transformations exist, but are mainly focused on transforming ORM models into relational database schemes (1 point). An interface for extending the language does not exist (0 points). Directly developing software from ORM schemes is out of scope of the language, yielding considerably low scores in the whole category. However, ORM has substantial support through modeling methods such as NIAM2007 (Lemmens, et al., 2007) and the Conceptual Schema Design Procedure CSDP (Halpin and Morgan, 2008) (3 points). Modeling processes is out of ORM’s scope (0 points). Using validation instances for designing the CDM is an integral part of ORM (3 points). Furthermore ORM is suited for verbalization and has a mechanism for surrogate names (3 points each). ORM supports all of the main building blocks, along with n-ary relations (3 points across the board), but a partitioning mechanism does not exist (0 points). A hierarchical structure cannot be explicitly described, but the semantic prerequisites for it exist (1 point). ORM scores considerably high in the constraint structure category, but misses out on user-defined constraints (0 points).

3.3.4 The Web Ontology Language OWL

In the context of the Semantic Web (Hendler, et al., 2002), OWL has proven its applicability for providing a logical sub-structure of data made available through the Internet. It is standardized by the W3C (2012) and forms one of the backbones of semantic information exchange between web resources. OWL is based upon the Open World Assumption (Hennig, 2012) and is focused on integrating vast amounts of knowledge across a variety of sources. OWL is also based upon a set theory principle called Description Logics that enables functions such as automatic classification of new knowledge and inference.

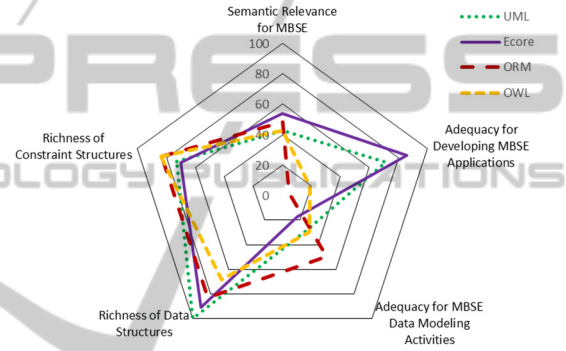

Figure 2: Language Evaluation per Category.

OWL is based upon the Open World Assumption. Some parts of the model can be locally closed down using workarounds (Mehdi and Wissmann, 2013), but resulting in substantial additional modeling effort (1 point). The basic structures of OWL ontologies are not hard to be understood, but the semantically rich structures are often difficult to grasp for non-ontology experts (1 point). A tailoring mechanism as required for MBSE does not exist (0 points). Different OWL ontologies can be integrated with each other (3 points) and some transformation capabilities exist (1 point). OWL can be extended by adding custom concepts, but running the risk of losing decidability (1 point). Developing applications is not really a focus of the OWL language, resulting in a low score in the software engineering category. The structure of the OWL language allows dynamic handling of (validation) instances in parallel to the CDM (3 points). OWL supports the basic data structures, but does not support object properties with an arity other than 2. (0 points). Furthermore an explicit hierarchical structure, apart from the class taxonomy, is not part of the language (0 points).The OWL language specifies many con-

straints related to set theory through restrictions and axioms. User defined constraints can be asserted using language extensions such as SPARQL or SWRL (3 points).

4 CONCLUSIONS

This section summarizes the evaluation and sketches a picture of how to approach identified shortcomings.

4.1 Summary of Language Evaluation

The analysis made evident that the ideal language for conceptual data modeling does not exist. Each of the discussed languages stands out when regarding characteristic aspects.

Due to the very specific requirements on the modeling languages in the semantic relevance category no language is able to fulfil all of the requirements. Some features, such as the separation between tool-development and tool-customization concepts are not supported by any language. In the software engineering category, both UML and Ecore score well, but Ecore comes out with the highest score due to the extremely tight integration and attunement with EMF, compensating for the low score at implementation-independence. The third category dealing with CDM engineering activities, also due to very specific requirements, is not well supported by any of the languages. ORM comes closest to fulfilling them, but misses out on the process aspects. Regarding data structures UML fulfills all of the requirements, with Ecore and ORM scoring a bit lower and OWL coming out last. No language is able to fulfill all requirements in the constraint category, but both of the knowledge-oriented modeling languages, OWL and ORM, attain a similarly adequate score. Figure 2 visualizes the results of the evaluation by presenting the performance of the languages per category. A value of 100 expresses that the language has attained the highest possible score in a category.

4.2 Necessity for Future Development

Since the ideal modeling language does not exist, significant improvement is becoming necessary. A combination of each of the language's strong suits is one possible course of action in order to cope with the requirements needed for effective and efficient MBSE. One possible approach for bringing together the advantageous features of these languages is using Ecore with its meta-modeling mechanism for leveraging its software engineering capabilities. An Ecore metamodel extension based on ORM can serve as abstract and concrete syntax for CDMs, utilizing ORM's capability for producing semantically rich data and constraint structures. Ontological concepts derived from OWL are used for employing additional knowledge-management functions while avoiding the Open World Assumption. In addition a number entirely newly developed functional aspects, including a tool-customization mechanism and a connection of the CDM to its originating engineering processes has to be included. Only such a harmonized approach, merging the strong suits of each of these modeling languages, combined with new aspects, is able to facilitate genuinely cost-efficient, effective and consistent MBSE across a wide variety of engineering domains as required for space systems engineering.

REFERENCES

ANSI/X3/SPARC Study Group on Data Base Management Systems, 1975. *Interim Report. FDT, ACM SIGMOD bulletin. Volume 7, No. 2,* s.l.: s.n.

Dassault Systemes, 2014. *CATIA Version 5-6 Release 2012.* [Online] Available at: http://www.3ds.com/products/catia/portfolio/ catia-v5/latest-release/

Eclipse Foundation, 2014. *Eclipse Modeling Project.* [Online] Available at: http://www.eclipse.org/modeling/emf/

Eclipse Foundation, 2014. *org.eclipse.emf.ecore (EMF Documentation).* [Online] Available at: http://download.eclipse.org/modeling/emf/emf/javadoc/2.9.0/org/eclipse/emf/ecore/package-summary.html

Eisenmann, H., 2012. *VSD Final Presentation.* [Online] Available at: http://www.vsd-project.org/download/presentations/VSD_P2_FP_2012-05-15_v3.pdf/

ESA, 2012. *The Virtual Spacecraft Design Project.* [Online] Available at: http://vsd.esa.int/

ESA, 2013. *EGS-CC - European Ground Systems - Common Core.* [Online] Available at: http://www.egscc.esa.int/

Fischer, P. M., Eisenmann, H. and Fuchs, J., 2014. Functional Verification by Simulation based on Preliminary System Design Data. *6th International Workshop on Systems and Concurrent*

Engineering for Space Applications (SECESA), 8-10 October.

Friedenthal, S., Griego, R. and Sampson, M., 2009. *INCOSE Model-Based Systems Engineering Workshop.* [Online] Available at: http://www.incose.org/Chicagoland/docs/SanDiego/3-18-09%20INCOSE%20Model%20Based%20Systems%20Engineering%20(MBSE)%20Workshop.pdf

Halpin, T. and Morgan, T., 2008. *Information Modeling and Relational Databases.* 2nd ed. Burlington: Morgan Kaufmann.

Halpin, T. and Wijbenga, J. P., 2010. FORML 2. In: I. Bider, et al. eds. *Enterprise, Business-Process and Information Systems Modeling.* Berlin: Springer, pp. 247-260.

Hendler, J., Berners-Lee, T. and Miller, E., 2002. Integrating Applications on the Semantic Web. *Journal of the Institute of Electrical Engineers of Japan,* 122(10), pp. 676-680.

Hennig, C., 2012. *Evaluating Ontologies for Use in Model-Based Systems Engineering,* München: Technische Universität München.

Hennig, C. and Eisenmann, H., 2014. Applying Selected Knowledge Management Technologies and Principles for Enabling Model-based Management of Engineering Data in MBSE. *6th International Workshop on Systems and Concurrent Engineering for Space Applications (SECESA)*, 8-10 October.

Hong, S. and Maryanski, F., 1990. Using a Meta Model to Represent Object-Oriented Data Models. *6th International Conference on Data Engineering*, 5-9 Febuary, pp. 11-19.

INCOSE, 2014. *Systems Engineering Vision 2025.* [Online] Available at: http://www.incose.org/ProductsPubs/products/sevision2025.aspx

ISO, 2004. *ISO 10303-11: Industrial automation systems and integration – product data representation and exchange – Part 11: Description methods: The EXPRESS language reference manual.*. s.l.:s.n.

ITP Engines UK, 2014. *ESATAN-TMS: Home.* [Online]
Available at: https://www.esatan-tms.com/

Kogalovsky, M. R. and Kalinichenko, L. A., 2009. Conceptual and Ontological Modeling in Information Systems. *Programming and Computer Software,* 35(5), pp. 241-256.

Lemmens, I., Nijssen, M. and Nijssen, S., 2007. A NIAM2007 Conceptual Analysis of the ISO and OMG MOF Four Layer Metadata Architectures. In: R. Meersman, Z. Tari and P. Herrero, eds. *On the Move to Meaningful Internet Systems 2007: OTM 2007 Workshops.* Berlin: Springer, pp. 613-623.

Mehdi, A. and Wissmann, J., 2013. EQuIKa System: Supporting OWL Applications with Local Closed World Assumption. *GI-Jahrestagung*, Volume 220 of LNI, pp. 1943-1948.

MSC Software, 2014. *Patran.* [Online] Available at: http://mscsoftware.com/product/patran

NASA, 2007. *NASA Systems Engineering Handbook (NASA-SP-2007-6105) Rev1,* s.l.: s.n.

No Magic, 2014. *MagicDraw.* [Online] Available at: http://www.nomagic.com/products/magicdraw.html

Olivé, A., 2007. *Conceptual Modeling of Information Systems.* Berlin: Springer.

OMG, 2012. *OMG Systems Modeling Language (OMG SysML), Version 1.3.* s.l.:s.n.

Suárez-Figueroa, M. C., 2010. *NeOn Methodology for Building Ontology Networks,* Madrid: Universidad Politécnica de Madrid.

Sure, Y., Staab, S. and Studer, R., 2004. On-To-Knowledge Methodology (OTKM). *Handbook on Ontologies,* pp. 117-132.

Van Renssen, A. S. H. P., 2005. *Gellish - A Generic Extensible Ontological Language,* Delft: Technische Universiteit Delft.

W3C, 2012. *OWL 2 Web Ontology Language Primer (Second Edition).* [Online] Available at: http://www.w3.org/TR/owl2-primer/