14th World Congress on Computational Mechanics (WCCM)
ECCOMAS Congress 2020
Virtual Congress: 11-–15 January 2021
F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

A PRACTICAL APPROACH TO ONTOLOGY-BASED DATA MODELLING FOR SEMANTIC INTEROPERABILITY

THOMAS F. HAGELIEN[1], HEINZ A. PREISIG[2], JESPER FRIIS[1], PETER KLEIN[3], NATALIA KONCHAKOVA[4]

[1] SINTEF, NO-7465 Trondheim, Norway, Thomas.F.Hagelien@sintef.no, jesper.friis@sintef.no

[2] Norwegian University of Science and Technology, Department of Chemical Engineering, NO-7491 Trondheim, Norway, heinz.a.preisig@ntnu.no,

[3] Fraunhofer ITWM, Fraunhoferplatz 1, Kaiserslautern, 67663, Germany, peter.klein@itwm.fraunhofer.de

[4] Institute of Surface Technology, Helmholtz-Zentrum Geesthacht, Max-Planck Str. 1, Geesthacht, 21502, Germany, natalia.konchakova@hzg.de

Key words: Data modelling, Ontology-based semantic framework, Digitalization strategy, Interoperability

Abstract.

Efforts to provide a standard representational framework based on current materials modelling and characterization knowledge facilitating collaboration, digital data representation, knowledge systems and semantic interoperability has been the main agenda for the European Materials Modelling Council (EMMC). One challenge in adopting the technology is related to the on-boarding process, particularly regarding the availability of ontologies and practical aspects to linking data to the ontologies, which requires deep insights in the current knowledge organization system. Latter challenges we address by constructing an interoperability framework based on data-models and explicit ontological mappings. Data models allow for building a representation of a computer system from different perspectives. The conceptual view identifies what real-world concepts the data represents. The logical perspective defines the rules and structures of how to implement the strategy. The physical perspective establishes the relationship between the data model and a specific database system. We show the ontology-based data modelling approach. It addresses the separation of the logical and the conceptual perspective, and allows for the development of the data-models. This approach also puts the data-models directly into production before applying the ontological mappings or developing the domain ontologies. Finally, we show a framework for information exchange that connects data models to physical storage and allows software applications to eliminate the need to support specific input-output operations, file conversion, file versioning, etc.

1 INTRODUCTION

Semantic interoperability is the exchange of information with a shared meaning between two computer systems. Scientific software used by the industry to manage innovation challenges have limited capa-

bilities of exchanging information with other software solutions and data providers and requires manual work to create adopters or convert file-formats etc.

The European Materials Modelling Council (EMMC) supported by a set of surrounding H2020 projects, has focused on establishing and standardising technologies for model and data semantic interoperability based on the European Material Modelling Ontology (EMMO) [1]. EMMO is a multidisciplinary effort by EMMC to provide a standard representational framework based on current materials modelling and characterization knowledge facilitating collaboration, digital data representation, knowledge systems and semantic interoperability. Instead of starting from general upper level concepts, as done by most other ontologies, the EMMO development started from the very bottom level, using the actual picture of the physical world coming from applied sciences, and in particular from physics and material sciences. It has grown from the bottom (i.e. scientific application field) to the top (i.e. conceptualisation), staying focused on the original scope while at the same time maintaining an approach as general as possible. It has a strong philosophical foundation adopting a physicalistic/nominalistic perspective in which all entities are seen as real world 4D objects extended in space and time. EMMO also states that there is a smallest elemental extension of space and time due to the uncertainty principles, termed *quantum mereology*. All relations in EMMO are either mereotopological (causally self-connectedness and parthood) or semiotical (real world objects that stand for other real world objects that are interpreted by an agent). Finally EMMO embrace that the world can be described from different, but interlinked *perspectives*; including the *physicalistic* (categorisation according to physical sciences), *perceptual* (physicals that stand for real world objects that can stimulate a perception), *reductionistic* (strict hierarchical representation of real world objects using direct parthood relations) and *holistic* (processes as whole 4D objects with participants) perspectives.

The ability to manage data used in at least two different software systems is particularly important in the case when the simulations are covering multi-step, multi-discipline or multi-scale analysis of complex scientific or industrial problems. In particular, the materials design process included materials‘ engineering and analysis of business related key performance indicators (KPIs) is the typical example of this issue. In general, all sets of data - original experimental data/ materials data, computational data, business data, are involved in the considered data-centric approach to find the optimal solution by using complex workflows. Thus, all data are subjected to the same procedure of the exchange, transformation and elaboration by different software or different simulation systems / simulation platforms.

A well known challenge in setting up a complex modelling workflows is the interrelated calculation of different aspects of materials behaviour and optimization, where the output from the first model (A) is not represented in the way expected by the second model (B) [2]. However, if the output of model A is semantically equivalent to the input needed by model B, it supports the successful transformation by post-processing of the output of A to be used by model B, and hence establish a robust workflow for analysis and optimization of considered materials and business KPIs [3]. This concept reflects one aspect of model-driven development, which describes the data transformation processes, input-output of physical and business simulations by data models design of both relative simple or extremely complex computational systems. Following this approach, software could be written focused on the business logic for the generation of codes, when input-output, file-format, version handling and data import-export can be handed by reusable components in the modelling framework reducing data uncertainty and development time, as well as facilitating semantic interoperability.

The model driven development requests standardisation of the modelling data description. One of the possible ways to solve the issue is to use the materials modelling data table (MODA) [4] developed by the EMMC. MODA is an instrument for documentation of simulations and understanding of complex modelling approaches, general modelling project structure as well as the models interaction and data transformation within the considered workflows [5]. It contains a user case description independent on any modelling information, allowing benchmarking different simulation- and experimental approaches [4]. Modelling concepts and their relationships are organized in the MODA in a focused collection, expressing the common structure shared by all simulations. It allows the representation of a materials simulation in the most general way. In that content the template might be mature enough for extended unambiguous model descriptions. Currently the MODA is developed and supported only for physics-based models. Moreover, there is a progressive step for MODA schema elaboration, which includes the extension of the tables to business decision support systems (BDSS) and model selection for considered business user case [3].

One premise for the success of EMMO and the ability to build complex workflows that utilizes semantic technologies, is that the cost of adopting the technology in the on-boarding process is justifiable. The goal here is to demonstrate that this is achievable. One key element in the semantic interoperability framework is the data model driven approach. The data models are artifacts that describes data. The data model captures the bare minimum of what information needed in order to correctly interpret the data contents in a data-set (the physical perspective). The human aspects are considered in the design of the data models, as people will need to be able to write and modify data-models when needed. The data models are aggregated into a collection of models that describes a complete system as a set of entities and the relationships between them. By connecting the data models to a data-space where heterogeneous data sources are linked, the instances of the data models can be used to access data in a unified manner. By mapping the data models to ontological concepts, the information about the data is further enriched - and information can be retrieved from a high level conceptual perspective, via the data models and the data sources.

2 Technology Stack for Semantic Interoperability

In this chapter the technology stack for a software system that supports semantic interoperability is presented. Here we look at basics for the management and organization of knowledge bases in triple-stores and how linked data and information discovery and exchange is managed by a data space component.

The semantic interoperability technology allows for utilizing computational logic (inference), discovering knowledge, aggregate data from distributed heterogeneous data sources and exchange information between two computer systems. The technology to enable semantic interoperability has a strong foundation in concepts and abstractions that includes vocabularies, taxonomies, relations, restrictions and a set of rules and axioms.These are the main ingredients of ontologies. Ontologies in computational science are usually constructed by a set of semantic triples or Resource Description Framework (RDF) triples, as specified by the W3C Consortium[1, 2]. In order to transmit information between two computer systems, a set communication protocols must be established. Web protocols (HTTP/HTTPS) are especially important, as they allow for information sharing across the World Wide Web. Linked data is a struc-

[1]The World Wide Web Consortium (W3C) is the main international standards organization for the World Wide Web

[2]RDF is a standard model for data interchange on the Web `https://www.w3.org/RDF/`

ture that ”connects” data sources/databases across the internet, based on ontologies and web protocols. Linked data enables discoverability through semantic search and queries. Applications build on top of this technology stack has the ability to be semantically interoperable with other semantic applications.

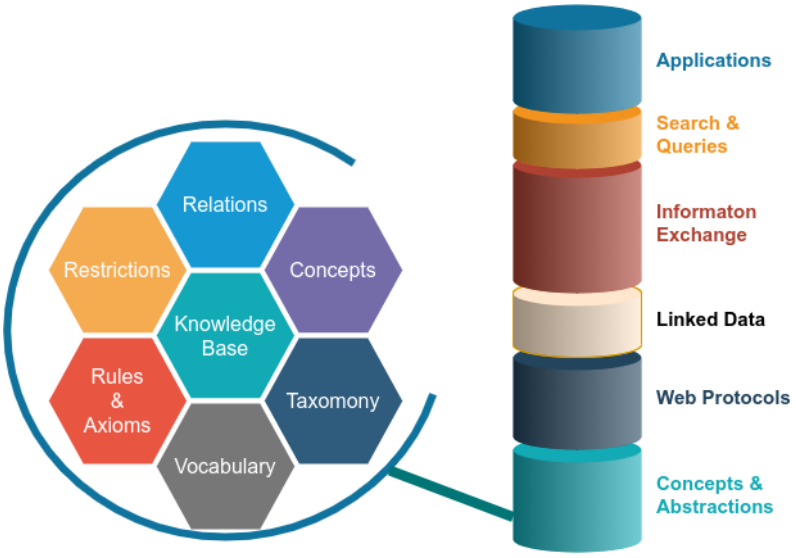

Figure 1: The technology stack for a software system that supports semantic interoperability. Ontologies such as EMMO are defined as a set of concepts and abstractions. A Structure of interlinked distributed heterogeneous datasets sits on top of standard web protocols and can be shared. The combinations of knowledge bases and linked data allows information to be discovered through reasoning or querying. On top of the interoperability technology stack sits the application that brings business value.

The main goal here is to employ semantic interoperability to simplify the construction and validation of complex workflows. The high-level concepts and abstraction are defined in EMMO and the derived domain ontologies. The data sources are linked to the semantic data models defined in this paper. The data models are further mapped to concepts in the ontologies. With the data model as a mediator, there is now a clear relation between a specific data point or value stored in a data source and the semantic concept that describes its meaning. In the simulation software, the data models are instantiated either as objects in the code itself, or through a wrapper layer. The serialization of data from the data source into the objects are handled by the interoperability framework where different reusable drivers for data sources are implemented. The interoperability framework employs a triple-store in the back-end, that allows for applications to perform search and knowledge retrieval. The simulation software will produce new knowledge that can be represented by ontology-mapped data models. This information can either be passed to a different simulation tool, or stored in a shared database. In the latter case a reference to the information will be passed further.

2.1 Managing the Knowledge Base

The backbone of the semantic interoperability technology stack is the knowledge base where the ontologies that encodes information about concepts, taxonomies, vocabularies, rules and axioms etc are managed. Very commonly a triple-store is employed as a database for the knowledge bases.

A triple-store is a special kind of database that stores and retrieves information in the shape of triples. As with SQL databases, information in triple-stores can be retrieved via queries. Triple-stores will often also support semantic reasoning. A reasoner is an engine that performs logical inference from a set of rules and axioms. This allows new information to be generated. The triple-store is the backbone of the semantic interoperability technology stack, and the information contained there (i.e. the ontologies) is shared with all information systems that performs knowledge exchange. The main usage for the triple-store here is as back-end for the interoperability platform to store the EMMO with the domain ontologies, the data models and mapping between these, as well a storing necessary metadata for the external data sources.

2.2 Data Space

The *data space* (Figure 2) is a software component that connects heterogeneous databases with data models and manages the mapping of data models to ontological concepts. The data space employs an in-memory database that stores the state (values of instances) of a software, derived from the data models. In a typical implementation, the data-model is realized as a *Class* where the state of an object instantiated from the class is stored in the data space. This makes it possible to take a snapshot of the application state (value of all local variables in a software program) by storing the data in the data space. This is basically a serialization/storage of the in-memory database contents, with the additional feature that the snapshot will be identified by a universally unique identifier (UUID). It is also possible to include a reference to previous snapshots (parent UUIDs) in a snapshot. This will enable be ability to track all state changes through a sequence of snapshots. In the inverse operation, when a state is to be restored, a specific snapshot ID will be pulled into the in-memory database, and effectively populate a new set of objects with values. To organize the set of objects that are managed by the data space, a set of relations between the objects can be defined. The set of objects and the relations between them are called *collections*.

The data space can be extended with a triple-store that allows for data models that are mapped to ontological concepts to be discovered and queried. The triple-store will allow for reasoning based on contents, rules and axioms and opens for possibilities such as validating workflow based on the knowledge about required inputs and expected outputs in connected materials simulation tools and post-processors.

3 Ontology Based Data Modelling

This section presents the construction of the proposed data-model. The qualities of the data-model considers both the human and software system perspectives. The main contribution is the ontological mapping of the data-models to the EMMO, that is the key to knowledge organization and an enabler for semantic interoperability.

3.1 Data Model Construction

While data models can be generated from schemas and other data sources, data models are often created and edited by people. It is therefore important to consider the qualities of the data model from the perspective of a human developer. Figure 3 illustrates the required qualities for our data model. For a data model to be editable, visual editor could be employed. Otherwise, the textual representation must be light-weight and readable without too much markup and visual noise.

The data models needs to be version controlled in order to ensure that software based on the models

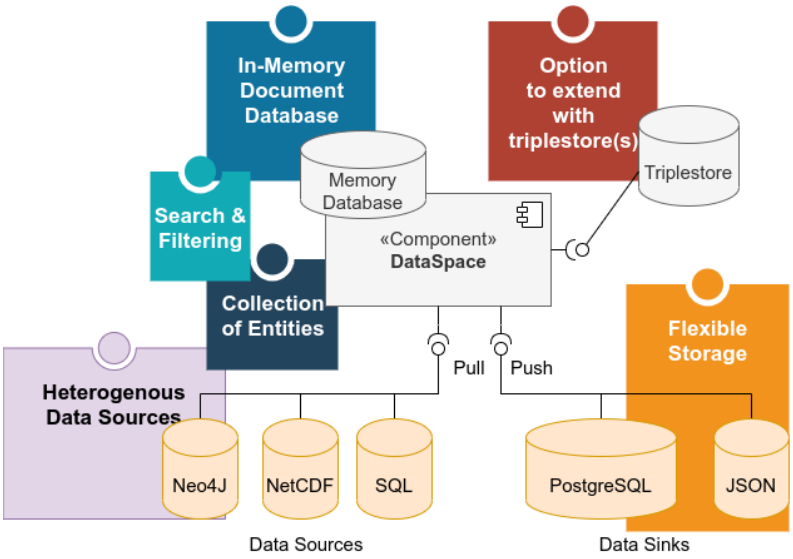

Figure 2: The DataSpace is a component of the interoperability framework that manages program state in shared memory (in-memory document database). This allows for snapshot to be stored (in data sinks) and restored on demand. The DataSpace can connect to heterogeneous data sources and pull information that will be linked to the data models. By connecting the DataSpace to a triple-store, the data models can be mapped to ontological concepts.

are able to reproduce results. Version control also have the ability to safely upgrade and downgrade between versions. The data models needs to be aggregatable, i.e. the ability to form a larger system of data models. In order for the data models to be semantic, it also needs to have the ability to map to ontological concepts in one or several ontologies.

The metadata schema (Figure 4) describes the abstract data model that may be instantiated to represent a real world object or *entity*. Each entity has a unique identity. While the values associated with the entity may change over time, the identity is constant. The values associated with the entity are called properties. Each property can have attributes such as data type, unit and labels. In the case where the values are not scalars, the shape of the data can be specified. The shape consist of a set of named dimensions that a associated with the entity. This enables data to share dimensional information. This is useful for instance in cases where the data is organized as a grid, and the shape of the data is defines the grid dimension sizes. The data types will usually be intrinsic data types such as integers, characters, floating point numbers, boolean numbers etc., but it is often useful to also support more complex types such strings, enumerators and sum types.

3.2 Ontological Mapping

Figure 5 compares the metadata hierarchy between the data model and an ontology, where metadata in this context is a formalised description of data confirming to the metadata, rather than providing annotations or additional context to the data. In the data model, the data objects are described by their metadata (or class), which in turn is described by their *metadata schema* as it is named in Figure 5. It is possible to consider a hierarchy of abstractions where each new level of abstraction can describe the abstraction

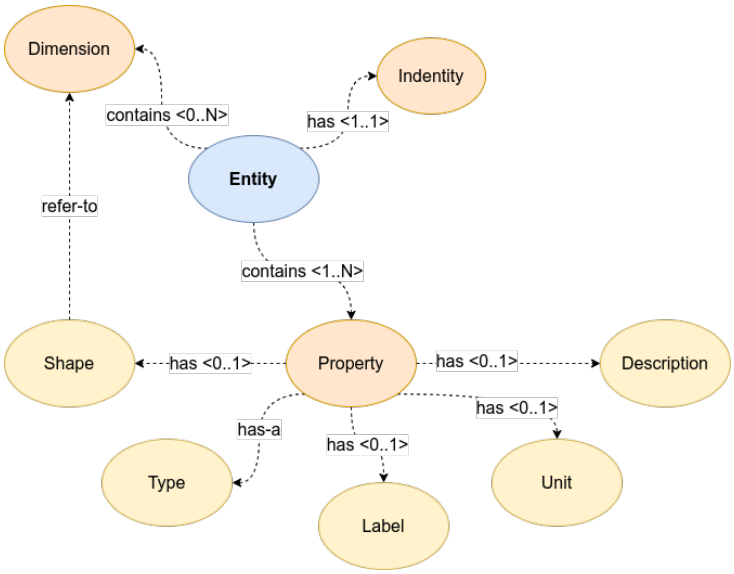

Figure 4: The proposed data-model is a representation of an Entity that has a unique identity, a set of dimensions and properties with attributes such as type, label, unit and descriptions. In addition, multi-dimensional properties can be defined with a shapes as being an array of dimensions.

level just below it. The proposed data model shown in Figure 5 has four levels of abstractions, where the top-level *basic metadata schema* is able to describe itself and thereby removing the need to define any further levels of abstractions. Ontologies typically have three levels of abstractions, individuals, classes and the language used to describe the classes with, which is OWL-DL[3] in this case.

Ontologies are typically used to semantically define relevant concepts in one or several domains in terms of classes and individuals. Via formalised relations between the concepts, ontologies are able to provide a high level of context and enable inter- and cross-domain interoperability by providing a common representational framework. This interoperability can be transferred to the data model, by mapping classes in the data model to classes in the ontology and data objects in the data model to individuals in the ontology. However, often the mappings are only done at class-level.

Figure 6 shows an ontologization of the proposed data model as a formal language. A formal language is in EMMO a symbolic object respecting the specific syntactic rules of the language, who's structure is described with spatial direct parthood relationships. Hence, two types of relations are used, `rdf:subclassOf` and `emmo:hasSpatialDirectPart`. Direct parthood is a non-transitive parthood relation defined in EMMO that makes it possible to create well-defined parthood hierarchies.

Figure 6a shows the root of our data model branch with `DataModel` its three subclasses; `Object`, `DataType` and `Type`. Objects (Fig. 6b) have a set of direct parts including a unique identifier (UUID), a reference to its Class (Meta) and the actual values of its dimensions and properties. Classes (Fig. 6c) are identified by their namespace/version/name combined into a uniform resource identifier (URI). By

[3]OWL-DL is a sub-language of the Web Ontology Language (OWL) based on description logics that combines mediate expressiveness with decidability and computational completeness. Since description logics is a decidable subset of first order logic, it allows for automatically computation of the classification hierarchy and check for inconsistencies using a logical reasoner.

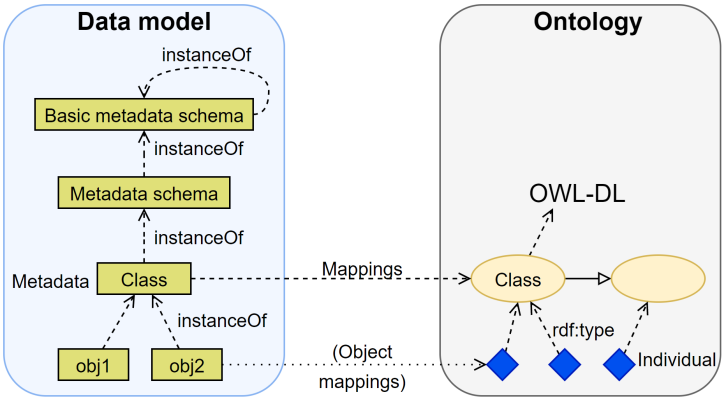

Figure 5: The corresponding meta-levels in the data model and ontology. Note that the meta-levels (vertical axis) are orthogonal to the relations between classes or between individuals (horizontal axis) in the ontology. Mappings maps classes in the data model to classes in the ontology and data objects in the data model to individuals in the ontology.

including the version into the identifier allows for proper versioning. In addition classes have a human readable description and a set of dimensions and properties. Note that classes are themselves objects and therefore also inherits the part of `Object`. The UUID of a class is a unique hash of its URI. The `Meta` part of a class refers its metadata schema. The `Meta` part of a metadata schema refers the the basic metadata schema, and so forth.

The `Dimension` and `Property` parts of a class specify what dimensions and properties its instance objects may have. Their actual values are parts of the individual objects. Dimensions has a name and has a human readable description. Properties has a label, type, shape, unit and a human readable description. To homogenise the handling of types that can have different sizes, like integers or binary blobs, a type has in this data model two components, a data type and the size in memory that one instance of the data type occupies in memory. Array properties have a shape that refer to one or more named dimensions. Finally, units are a key part of a quantity alongside its value and is included in order to allow properties to represent quantities, which is a common need in physical sciences.

The different data type subclasses are not specified at this point, but would typically be boolean, integer, float, string, etc. In addition dimensions and properties are considered to be datatypes,

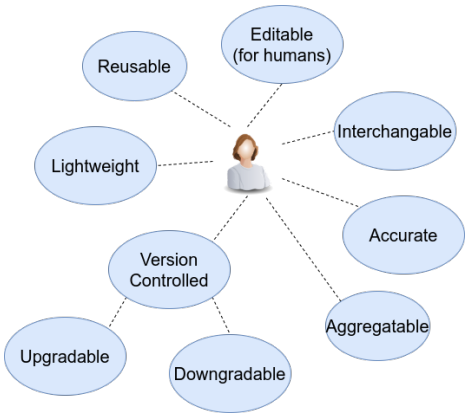

Figure 3: The qualities of a data model to be human-friendly as well as computer interpreted includes using a syntax that is easily readable and editable with relatively few attributes to consider. Version control is essential as models with be modified during the lifetime of a software system, with the ability to upgrade or downgrade between versions.

which makes it possible for metaclasses to describe classes and meta-metaclasses to describe meta-classes. Adding a specific type for RDF subject-predicate-object triplet has also proven to be very useful.

The data model described here has been implemented and its versatility has been demonstrated in the open source project dlite [6].

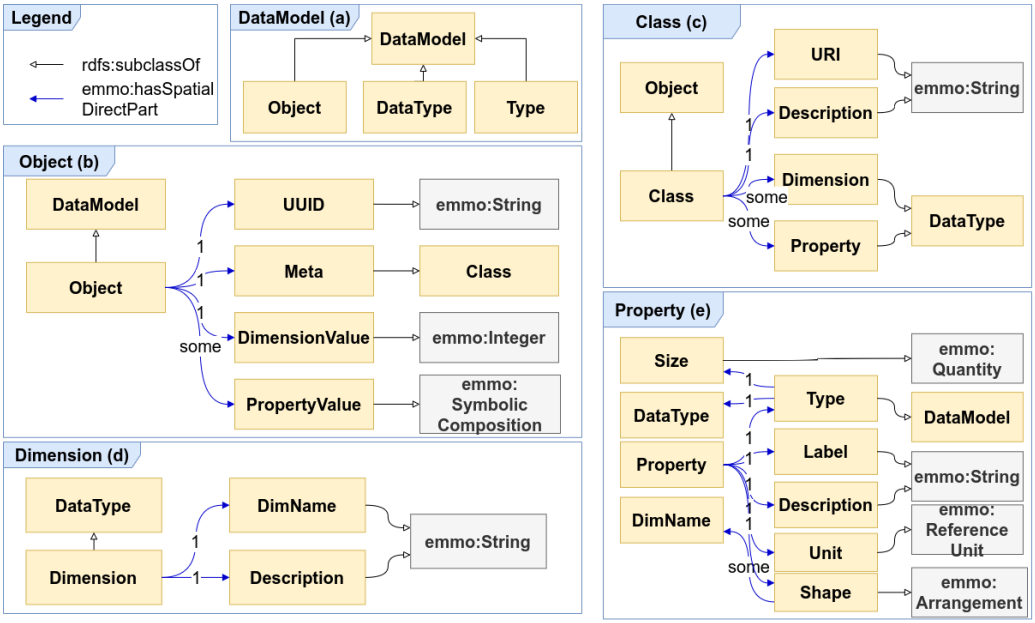

Figure 6: Ontological representation of the proposed data-model showing the five main classes; (a) DataModel, (b) Object, (c) Class, (d) Dimension and (e) Property. All boxes represent classes. Gray boxes refer to EMMO, while yellow boxes are introduced in our proposed data model ontology.

3.3 Code Generation

Previous experience with applying datacentric modelling frameworks (such as SOFT (SINTEF Open Framework and Tools) [7] and dlite [6]) for industrial applications, has shown that code generation is an invaluable tool. Advantages include rapid development and high quality code. In addition, code generation allows for adopting to different targeted programming languages and data-schemas, as well as supporting automatic test generation and documentation.

Template engines such a Jinja and Mustache[4] allows for code generation based on structured data. They are great for creating programming-language specific classes that represent entities created from the data-model. This allows for developers to specify the generic properties and attributes of an entity, and the code generation tool will generate classes, APIs, bindings to interoperability frameworks and documentation. When using programming languages which support dynamic creation of classes, it is possible to have the classes representing the entities created at run-time. This is a huge advantage especially in the case when a dynamic system requires to read in an external state of unknown entities (i.e. produced by a different software system). In such cases, the software can fetch the entities, generate the code

[4]See templating engine Jinja `https://github.com/pallets/jinja` and Mustache `https://github.com/Mustache/Mustache`

and instantiate the entities without having to stop or restart the system. In the Python programming language, features such as *lambda functions* can construct closures for setters and getters to access private attributes in the generated Python classes. It is also possible have these setters and getters reading and storing the values from shared memory. This allows sharing state between copies of entities either in the same language, or across languages in cases where the software is written in mixed languages.

3.4 Physical modelling – the generation of "talking" simulations

A practical approach of employing an interoperability framework is briefly discussed here using the ProMo[8] simulation environment. As a rule, executable tasks are to communicate with each other through the data space provided by the interoperability framework. ProMo is a simulation environment that builds simulations of physical processes augmented with control and analysis tools, such as life-cycle analysis and economical analysis. The modelling of the physical system uses elementary building blocks for the construction of composite models. Such elementary building blocks are fundamental entities defined by the smallest granulation level required to capture the behaviour of a process on a given time scale. These building blocks are specific to the discipline being employed for the description. More details about ProMo can be found in companion paper [8] on code generation [9] and on the indexing [10] on discipline-dependent fundamental building blocks aiming at the modelling of physic in [11]. An ontology defines the basic building blocks – the fundamental entities. Their behaviour is defined by an equation editing tool that uses the rules defined by the ontology and generates a multi-bipartite graph of equations and variables. Figure 7 captures the greater picture. The equation editor imports the ProMo

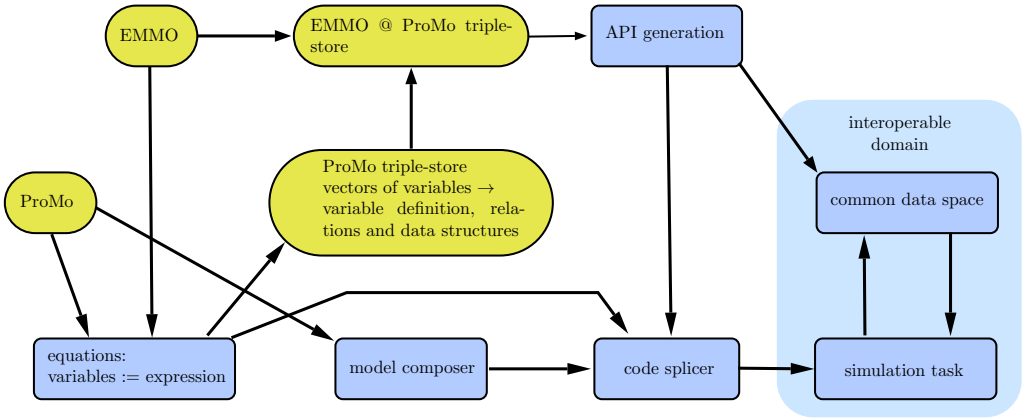

Figure 7: The ProMo and the EMMO ontology provide the structured knowledge to define the behaviour equations of the principle model building blocks. They are used to generate the code in the code splicere for the models constructed from the building blocks in the model composer. They are also compiled into OWL and added as an extension to the EMMO, from which one generates the APIs using the discussed framework. The APIs enter through the code splicer in the simulation task thus realising the interoperation with the common data space.

ontology and the EMMO ontology to define the equations and save them in a triplestore using the Python package Owlready2 [5]. The ProMo variable data structure contains the information of a global identifier

[5]Owlready2 is a module for ontology-oriented programming in Python. It can load OWL 2.0 ontologies as Python objects, modify them, save them, and perform reasoning via HermiT. `https://owlready2.readthedocs.io/en/latest/`

(enumeration type), a name (string), dimensions (index set identifiers), index sets and physical units. By augmenting the EMMO with ProMo's triplestore, one generates an extension of the EMMO, which then can be processed to create the API's for the communication with the common data space.

4 Discussion & Conclusions

A practical approach to semantic interoperability has been demonstrated by illustrating an implementation of an interoperability framework covering ontologies, linked data and information exchange through the use of data modelling techniques, supporting discoverability and reasoning as a foundation for building industrial applications. In industrial cases, the semantic data models presented here can be employed to enrich different types of data that is essential for the decision making processes. Different stakeholders from engineering, manufacturing, procurement etc needs to build integrated models that exchange information as part of the decision tools that will eventually be measured against a set of KPIs. The data involved includes materials data and metadata including physical parameters, modelling parameters, business data, measuring methods, optimization criteria and manufacturing conditions. In addition it is also possible to link information with updated regulation databases that can be used for instance to verify no dangerous or illegal chemicals are included in the process. In order to build a complex system with different tools that fits the purpose for different stakeholders, the conceptual perspective of the entire business domain along needs to be defined in an ontology that captures all necessary abstract concepts, terms, relations, restrictions, rules and axioms. By mapping data models to the EMMO, knowledge is not only defining the specific business domain, but is also able to exchange information with external systems that share the same knowledge base. Since EMMO is based on physics and is able to describe physical processes and materials in all possible levels of granularity in a natural and physically correct way, there is a huge benefit of using this ontology in domains within applied sciences. The domain ontologies derived from the EMMO mid-layer ontology can in addition be mapped to workflow ontologies to describe and manage the complexity of different organizational processes. Information discovery can be further improved by mapping concepts to other external knowledge bases such as DBPedia[6] and WordNet[7].

The cost and investments needed in the on-boarding process for adopting this new technology is an important factor. The technology and terms are often foreign and the emerging technology has not been widely adopted in the industry yet. In this work the human factor is considered in the definitions of the qualities a data model must satisfy in order start adopting key elements of the technology. A simple yet powerful data model, that represents information and allows for a unified explicit description of information across different databases and software solutions, is the first step. Here the physical perspective of the data model is essential, as the data model will closely relate to the the existing information structure in specific database systems. The next step is to map concepts from the data models to a domain ontology. This will enable information to be interpreted ubiquitously across different computer systems.

Future work includes the further demonstrations of the usefulness of ontologies in interoperability frameworks, and practical guidelines of how to approach this technology. This will be realized in a running demonstration of the a real industrial application in the domain of metal surface degradation and protective coating performance[12] where numerous factors from business and modelling perspectives will be

[6]DBpedia is large-scale multi-lingual knowledge base extracted from Wikipedia `https://wiki.dbpedia.org/`

[7]WordNet is a lexical database of English `https://wordnet.princeton.edu/`

inputs in a decision making process and optimal product design.

Acknowledgment

The authors acknowledge the following European Union's research and innovation projects for the financial support: (i) OntoTRANS, H2020-DT-NMBP-10-2019, Grant agreement 862136; (ii) MarketPlace, H2020-NMBP-25-2017, Grant agreement 760173; (iii) FORCE, H2020-NMBP-23-2016, Grant agreement 721027; (iv) VIPCOAT, H2020-DT-NMBP-11-2020, Grant agreement 952903.

REFERENCES

[1] Ghedini, E., Goldbeck, G., Friis, J., Hashibon, A. and Schmitz, G.J. European Materials & Modelling Ontology, `https://github.com/emmo-repo/EMMO`.

[2] Mir Z.M., Friis J., Hagelien T.F., Svenum I.H., Ringdalen I., Konchakova N., Zheludkevich M.L., and Hoeche D., Interoperability architecture for bridging computational tools: Application to steel corrosion in concrete, *Modelling and Simulation in Materials Science and Engineering*, (2020), **28**:025003.

[3] Dykeman, D., Hashibon, A., Klein, P., Belouettar S., Guideline for Business Decision Support System (BDSS) for Materials Modelling, (2020) (64pp). https://zenodo.org/record/4054009

[4] CEN-CWA Workshop Agreement 17284: *Materials modelling - Terminology, classification and metadata*. https://www.cen.eu/news/workshops/Pages/WS-2017-012.aspx (2018).

[5] de Bass, A. F. *What makes a material function?* `https://op.europa.eu/en/publication-detail/-/publication/ec1455c3-d7ca-11e6-ad7c-01aa75ed71a1` (2017).

[6] Friis, J et. al DLite - lightweight data-centric framework for working with scientific data `https://github.com/SINTEF/dlite`

[7] Hagelien, T. F., Chesnokov, A., Johansen, S. T., Meese, E A. and Løvfall, B. T. **SOFT**: a framework for semantic interoperability of scientific software. *Progress in Applied CFD – CFD2017 Selected papers from 12th International Conference on Computational Fluid Dynamics in the Oil & Gas, Metallurgical and Process Industries*

[8] Preisig, H. A., Hagelien, T. F., Friis, J., Klein, P. and Konchakova, N., Ontologies in computational engineering 14th World Congress on Computational Mechanics (WCCM) ECCOMAS Congress 2020 Virtual Congress: 11-15 January 2021 F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)

[9] Elve, A.T., Preisig, H.A. From ontology to executable program code. *Comp & Chem Eng* **2018**. doi:https://doi.org/https://doi.org/10.1016/j.compchemeng.2018.09.004.

[10] Preisig, H.A. Constructing and maintaining proper process models. *Comp & Chem Eng* **2010**, *34(9)*, 1543–1555. doi:https://doi.org/10.1016/j.compchemeng.2010.02.023.

[11] Preisig, H. A., Ontology-based process modelling – with examples of physical topologies, MDPI, Processes, submitted (2021).

[12] Hoeche, D., Konchakova, N., Zheludkevich M.L., Hagelien T.F., Friis, J. Ontology Assisted Modelling of Galvanic Corrosion of Magnesium 14th World Congress on Computational Mechanics (WCCM) ECCOMAS Congress 2020 Virtual Congress: 11-15 January 2021 F. Chinesta, R. Abgrall, O. Allix and M. Kaliske (Eds)