

A methodological approach for ontology-based meta-modelling

Emanuele Laurenzi

*FHNW University of Applied Sciences and Arts
Northwestern Switzerland, Riggengbachstrasse 16, 4600 Olten, Switzerland*

Abstract

Ontology-based meta-modelling is a technique for the creation of domain-specific modelling languages (DSMLs) grounded in an ontology. Such languages enable the design of ontology-based models having the advantage of being both human and machine-interpretable. Thus, ontology-based meta-modelling fosters the business-IT alignment in enterprises by establishing a shared layer of semantics with respect to a domain of discourse among human and machine actors. In this paper, a methodological approach for the ontology-based meta-modelling is proposed, to ensure rigor to the produced ontology-based DSML. Specifically, the approach enables the creation of semantic rules that given a required domain-specific adaptation, it propagates the changes to an ontology. A research project sets the motivational context for the creation of an ontology-based DSML as well as provides real-world scenarios for the evaluation of the proposed approach.

Keywords

ontology-based meta-modelling, domain-specific conceptual models, human-machine interpretable models, business-IT alignment¹

1. Introduction

There exist various enterprise modelling languages that provide sets of pre-defined modelling constructs from which enterprise models can be created. For example, the standard Business Process Modelling & Notation (BPMN 2.0) [1] is used for business process modelling. Although such languages bring the benefit of creating uniform and sharable models across enterprises, in practice they are not sufficiently expressive to address every application domain and sometimes are too complex for domain experts to be fully understood [2]. The BPMN 2.0 specification, for example, spans more than 500 pages and the definition of elements is distributed across different sections and sometimes with conflicting semantics [3].

These issues can be overcome with an ontology-based meta-modelling approach [4], where domain-specific adaptations are applied on modelling languages to either extend modelling constructs, or to remove unnecessary concepts or properties. The approach substitutes the meta-model with an ontology, which formalizes the semantics of modelling constructs with a W3C standard like RDF(S). The advantages are manifold. An ontology-based meta-model, in contrast to a pure meta-model, enables automatic reasoning, thus resolving the above-mentioned conflicting semantics. Also, it resolves the interoperability problem that exists among meta-models of different meta-modelling tools. It also allows the creation of ontology-based models, for which human and machine actors can have the same interpretation. In turn, the ontology-based meta-modelling technique approach promotes the continuous business-IT alignment [5] and creates the foundation for intelligent enterprises [6].


A set of operators for domains-specific adaptations of ontology-based meta-models have been presented in [7]. This paper adds rigor to the domain-specific adaptations by proposing a methodological approach that outputs semantic rules. These are triggered by the afore-

BIR-WS 2023: BIR 2023 Workshops and Doctoral Consortium, 22nd International Conference on Perspectives in Business Informatics Research (BIR 2023), September 13-15, 2023, Ascoli Piceno, Italy

✉ emanuele.laurenzi@fhnw.ch (E. Laurenzi)

ORCID iD 0000-0001-9142-7488 (E. Laurenzi)

© 2023 Copyright for this paper by its authors.
Use permitted under Creative Commons License Attribution 4.0 International (CC BY 4.0).

 CEUR Workshop Proceedings (CEUR-WS.org)

mentioned operators and ensure the propagation of the desired adaptation from a visual to an ontology representation.

The paper is structured as follows: Section 2 describes the background and related work, ending with the problem statement. Section 3 introduces the patient transferal management case and artifact requirements. Section 4 discusses the proposed methodology. Section 5 shows a running example about how the methodology has been used and validates it. The paper concludes with Section 6 with a summary and future work.

2. Background and related work

Meta-modelling is a renowned technique in the Enterprise Modelling (EM) discipline [8] and enables the creation of domain-specific modelling languages (DSML) [9].

DSMLs capture concepts, relations and constraints tailored to specific problem domains or application areas. Examples for DSMLs and respective domain-specific models can be found in [10]. Domain-specific models offer a powerful means to company stakeholders (with different skillset) to quickly share understanding of an addressed reality, thus supporting decision-making to generate business value like the continuous Business-IT alignment, improvements of business processes, and innovative business models [11].

The meta-modelling framework of Meta-Object Facility (MOF) [12] presents four levels of abstraction where elements and relations of UML Class Diagram are used to create meta-models for modelling standards. A paradigm that emerged in the recent years is the ontology-based meta-modelling [4]. Differently from MOF, this new paradigm substitutes the meta-model with an ontology, enabling the creation of ontology-based models as an instantiation of an ontology-based meta-model.

Conceptually, an ontology-based model can be regarded as an instantiation of a conceptual model. “Conceptual models are artifacts produced with the deliberate intention of describing a conceptualized reality” [13] with the goal of supporting computationally human communication, domain understanding and problem solving. Conceptual models are used to support the design of databases, software, business processes, enterprises etc. [14]. Conceptual modelling received continuous attention by scholars [15]. Guarino et al. [13] examine the fundamental notion of conceptual models and characterize them with conceptual semantics and ontological commitments.

Ontology-based models can therefore be regarded as conceptual models where the semantics of a visual model is made formal through an ontology language, like RDF(S) [16]. Ontology-based models are a powerful means of organizing, representing, and leveraging knowledge, facilitating data integration, enabling intelligent systems, and enhancing decision-making processes.

There exists various and emerging works that strive to combine visual models with ontologies or knowledge graphs. The authors in [17] focus on legacy data sources containing domain-specific graphical models and export them into ontologies to make them queryable in a Linked Data environment. The automatic transformation is made possible by an AODxx-to-RDF plugin. A similar approach is discussed in [8]. Similarly, the plugin Archi-to-Neo4J² converts a model in ArchiMate language into a property graph. Compared to this plugin, the work of [18] provides a more generic approach where any conceptual model can be transformed into a property graph (also in Neo4J). The conceptual models can be created in EMF or ADOxx metamodeling platforms, but also from the Ecore-based modeling platforms Papyrus (for UML, SysML, and UML profiles) and Archi (for ArchiMate). Most recently, [19] proposed BPMN2KG as a transformation tool from BPMN 2.0 process models into knowledge graphs.

The presented approaches are characterized by one-shot transformation from the graphical to the ontology representation. Differently from these approaches, the ontology-based meta-modeling gives continuity to the two knowledge representations, where the graphical one is in continuous sync with the ontology representation, throughout language adaptations. As such, it requires mechanisms that can ensure the continuous propagation of changes. For this, a set of ten

² <https://www.hosiaislouma.fi/blog/archimate-neo4j>

operators were proposed in [7] for domain-specific adaptations through ontology-based meta-modeling. This paper describes a methodological approach that to a domain-specific adaptation it follows the correct and rigorous definition of ontological statements. Hence allowing the continuous sync between a visual modelling language and the respective ontology.

3. Patient transferal management case

The patient transferal management case derives from the Swiss research project Patient-Radar [20]. To reduce costs and keep a high-quality treatment of patients affected by complex physical surgeries, the project aimed to enable intersectoral collaboration between acute hospitals and rehabilitation clinics, where rehabilitative expertise had to be brought early into the acute somatic treatment process.

The process identification phase of BPM (Business Process Management) [21], led to the identification of most crucial process as the patient transferal management process. The latter can be defined as “a set of actions designed to ensure the coordination and continuity of care received by patients as the transfer between different locations or levels of care” [22]. The set of actions, also called administrative pathways, includes medical information and excludes the treatment of the patient, which is referred to as clinical pathways [23]. Such collaboration takes place within the complicated settings of the transferal management domain, where many domain experts are involved, i.e., from acute hospitals, rehabilitation clinics, and health insurance for cost reimbursements.

Following the best practices of BPM lifecycle [21], the patient transferal management process had to be modelled before it could be analyzed to make it efficient. Since existing modelling languages lacked specific elements such as hospital-related documents or activities, a DSML was developed (see DSML4PTM [24]). The DSML provides all the relevant concepts and decision types of the patient transferal management domain and also offers graphical notations that are familiar to domain experts such as physicians, nurses and transferal manager, thus enabling them to quickly understand and adapt models.

A set of ten operators allowed the domain-specific adaptation of modelling languages. Since these operators aim to crystallize the knowledge produced by the language adaptations formally into an ontology, they take the name of “operators for the ontology-based meta-modelling”. The 10 operators for the ontology-based meta-modelling are the following: *Operator 1: Create sub-class*; *Operator 2: Update class*; *Operator 3: Delete sub-class*; *Operator 4: Create relation*; *Operator 5: Update relation*; *Operator 6: Delete relation*; *Operator 7: Create attribute*; *Operator 8: Assign attribute type or value*; *Operator 9: Delete attribute*; *Operator 10: Update attribute*. The operators are further detailed in [7].

Figure 1 depicts the all the root concepts of the ontology-based metamodel.

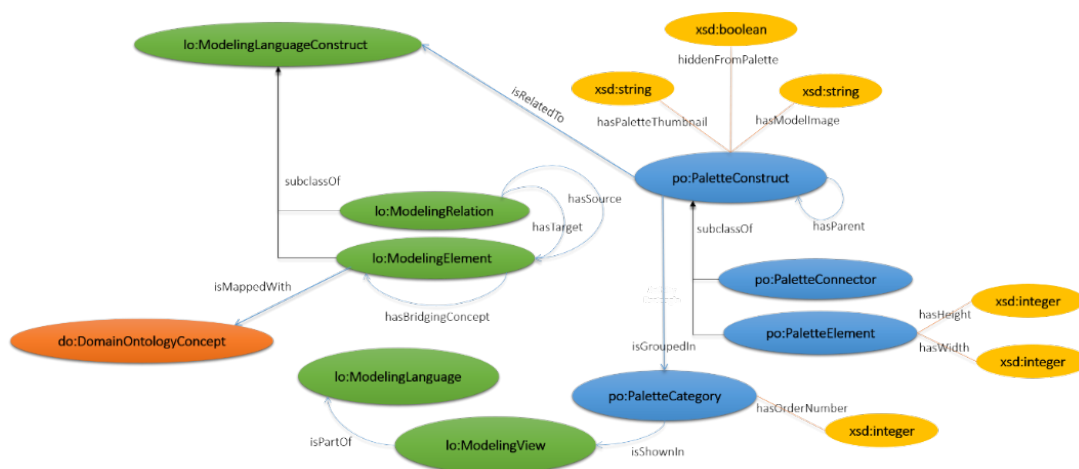


Figure 1: Root concepts and properties of the ontology-based metamodel [25]

The different prefixes indicate a different provenance of the terms. Specifically, “do” stands for domain ontology, “lo” stands for language ontology and “po” stands for palette ontology. Concepts from the language ontology can be associated with concepts from a domain ontology through the relation “lo:isMappedWith”. Concepts from the palette ontology are connected with concepts in the language ontology through the relation “po:isRelatedTo”.

Theoretically, the relation “lo:isMappedWith” is explained by the mathematical formula for the semantic mapping:

$$M: L \xrightarrow{\text{maps}} S [26]$$

where the semantic mapping (M) relates concepts from the abstract syntax (L) to the domain semantic (S). Similarly, the relation “po:isRelatedTo” is theoretically grounded in the relation between concept in the meta-model (or abstract syntax) and graphical notation, where notation visualizes concepts of the meta-model [27]. The palette ontology, in fact, contains all the graphical properties of concepts that are specified in the ontology-based meta-model. The self-relation “po:hasParent” is used to visualize the taxonomy among concepts in the modelling tool AOAME [28]. Similarly, the relation “po:isSchownIn” allows the visualization of graphical notation grouped by categories in the palette of AOAME.

Semantic rules ensure that the changes triggered by the operators are correctly propagated to the ontology. A semantic rule consists of one or more triples in the form of *subject-predicate-object* and *insert, delete* or *update* knowledge in the ontology-based meta-model. Each rule supports at least one operator. In this sense, if an operator is triggered by a domain-specific adaptation, the corresponding rule shall be instantiated. The instantiation contains the actual changes that need to be carried over to the ontology. For example, let us assume that an adaptation makes use of *Operator 1: Create sub-class*. The expected result is to create a sub-class relation between two modelling elements. To achieve that, the operator is supported by the two following semantic rules (1) insert the relation “po:hasParent” between the two palette element instances (in the Palette Ontology) that relate to the two modelling elements that wants to be connected; (2) insert the relation “rdfs:subClassOf” between the two modelling constructs (in the Language Ontology) that wants to be connected. See visually depiction in Figure 2. These semantic rules are, therefore, instantiated containing the actual instances of the palette element class and the two actual classes representing two modelling constructs. Thus, once rule instances are fired, the ontology-based meta-model are adapted with the new integration of the two modelling constructs.

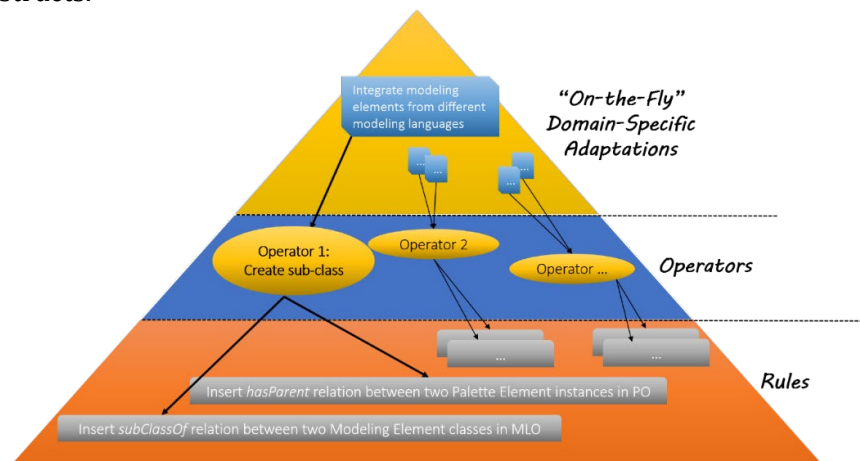


Figure 2: Semantic rules for the ontology-based meta-modelling [25]

4. Methodology for the Development of Semantic Rules

The design and evaluation of the semantic rules was supported by adapting the methodology proposed by Grüninger and Fox [29]. In the following, the general steps of the methodology are

described. Next, the adapted one is described, which is tailored to the development of semantic rules.

The methodology of Grüniger and Fox starts by explaining use case, from which informal competency questions (CQ) are derived and written in natural language.

Then, concepts and relations are extracted from the questions to conceptualise an ontology. Next, the ontology and competency questions are formalised through an ontology language and a rule language, respectively. Lastly, the evaluation of the semantic queries or rules is performed by executing them with respect to the ontology. In result, it should be shown that the expected results are met. Hence, the ontology is tested by proving completeness theorems with respect to the competency questions. The methodology has been successfully followed for the development and evaluation of several ontologies (e.g., [30]) and also used as a basis for the renowned Ontology Development 101 [31].

The purpose of the Grüniger and Fox methodology in our case is not of developing an ontology, but support the rigorous development of semantic rules. The latter aims to update the ontology of the meta-model that already exists. Thus, some part of the methodology is slightly adapted to fit our purpose (see Figure 3).

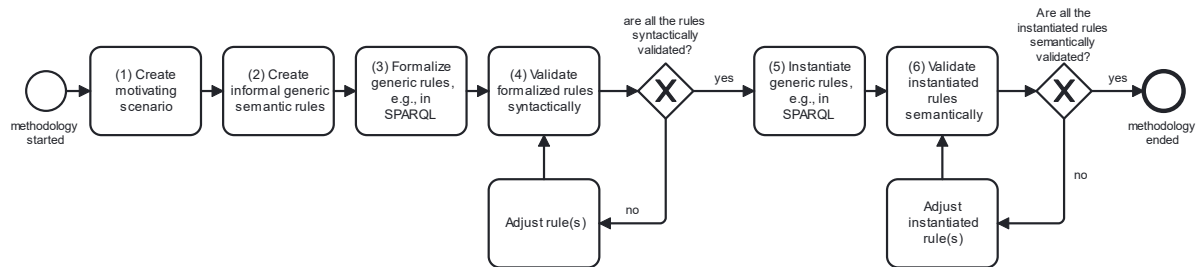


Figure 3: Methodology to design and evaluate semantic rules for the propagation domain-specific adaptations, in BPMN 2.0. Adapted from [29].

In the first step, domain-specific adaptations are described as a motivating scenario to design semantic rules. Next, informal rules (instead of competency question) are described in natural language. Finally, informal rules are transformed into SPARQL rules (instead of SPARQL queries). The validation of the designed SPARQL rules is done with respect to its syntactic and semantic correctness. The syntactic correctness is validated by executing the SPARQL rules with the SPARQLer Update Validator³. To ensure that the SPARQL rule produces the expected outcome, it is instantiated. This activity is underpinned by the patient transferal management case, which contains typical domain-specific adaptations and covers all the 10 operators for ontology-based meta-modelling. The SPARQL rule instances are fired against the ontology-based meta-model⁴. If results match with the expected ones it is the proof that a SPARQL rule is semantically valid. SPARQL rules that are syntactically and semantically validated were considered for their implementation in the web service of AOAME. Specifically, the validated SPARQL rules were incorporated in Java methods for the dynamic generation of SPARQL rule instances. The complete list of methods is publicly available in a GitHub repository⁵.

4.1. Representation Language of the semantic rules

The SPARQL rules are represented in the W3C language specification SPARQL Update [32]. The SPARQL Update provides graph update operations such as “INSERT DATA” and “DELETE DATA” with which the 10 operators can be supported.

According to the W3C [33], an operation is defined as “an action to be performed that results in the modification of data [..]” in a triplestore expressible as a single command, e.g. INSERT DATA or DELETE DATA. A triplestore is a mutable container of ontologies, which in our case reflect the

³ <http://www.sparql.org/update-validator.html>

⁴ The tests are done in the ontology editor TopBraid: <https://allegrograph.com/topbraid-composer/>

⁵ <https://github.com/manulaur/OntologyBasedModellingEnvironment-WebService/blob/master/src/main/java/ch/fhnw/modeller/webservice/ModellingEnvironment.java>

afore-mentioned ontology-based meta-model. Operations on RDF(S) resources span classes, properties and instances. Namely, the “INSERT DATA” supports those operators that imply the creation of a resource, i.e., *Operator 1 – Create sub-class, Operator 4 – Create relation, Operator 7 – Create attribute and Operator 8 – Assign concept, attribute type or value*. The “DELETE DATA” statement supports *Operator 3: Delete sub-class, Operator 6: Delete relation, Operator 9: Delete Attribute*. The sequential use of both operations, first “DELETE DATA” and then “INSERT DATA”, supports those operators where an update is completed, i.e., *Operator 2: Update class, Operator 5: Update relation and Operator 10: Update attribute*.

5. Running Example for Semantic Rule 1 – Operator 1

In total, eleven semantic rules have been created with the proposed methodology. To avoid an unnecessary stretch of this paper, the syntactic and semantic validation of only the first semantic rule is reported. Therefore, the first rule is described with all the 6 steps of the proposed methodology.

5.1. Semantic Rule 1: Modelling Elements Integration from different Modelling Languages

(1) Create a motivating scenario.

Within the patient transferal management case, there are situations where particular activities shall be executed only if certain conditions hold to true, thus independently from a pre-defined process flow. For example, the acute hospital decides on whether to start performing the admission for the incoming patient, which includes preparing all the needed resources. This is only possible after the transferal date has been agreed upon between the acute hospital and rehabilitation clinic and after the cost reimbursement form has been sent to the health insurance. Another example is on decisions that need to be taken if the patient’s situation worsens.

Therefore, at run time, a discretionary task can be executed independently from a process flow, as soon as certain conditions evaluate to true, else they are skipped.

Such a scenario motivated the integration of the BPMN Manual Task with the CMMN Discretionary Task where the latter is sub-class of the former. Semantically, a discretionary task can be regarded as specialization of a manual task because the human actor can still decide whether to execute the task or not.

The first semantic rule, thus, aims to support the integration of two modelling elements from different modelling languages (Discretionary Task of CMMN and BPMN Manual Task) with the relation *rdfs:subClassOf*.

(2) Create informal semantic rules (to Integrate Modelling Elements).

The integration between modelling elements from different modelling languages requires a rule that creates the following two properties (1) Create one object property *po:hasParentPaletteConstruct* in the Palette Ontology between the instance of the class *po:PaletteElement* (*instance son*) and the instance of the class *po:PaletteElement* (*instance parent*). *Instance son* and *instance parent* relate to the modelling construct that is to be extended and the modelling construct that extends, respectively. Both modelling constructs are classes in the Language Ontology. The object property *po:hasParentPaletteConstruct* ensures a taxonomy among the instances that contain the graphical notations. The taxonomy is then graphically displayed in the palette of the modelling tool AOAME. (2) Next, create one property *rdfs:subClassOf* in the Language Ontology between the modelling construct chosen for extending a class (*source class*) and the one that is being extended (*target class*). This action creates the desired integration between the two modelling constructs in the Language Ontology. It also ensures that the class taxonomy in the Language Ontology is consistent with the above-mentioned graphically displayed taxonomy of the notation.

(3) Formalize generic rules (SPARQL Rule 1 to Integrate Modelling Elements).

The informal rule for modelling elements integration is implemented by the SPARQL operation “INSERT DATA”. The latter fits the purpose of inserting one property *po:hasParentPaletteConstruct* between two instances and one property *rdfs:subClassOf*, between two classes.

SPARQL Rule 1 is shown in Table 1 and contains (1) the prefixes *rdfs*, *po* and *lo* that are used in the two statements. The prefix *rdfs* refers to the syntax of the RDF(S) ontology language. The prefix *po* refers to the Palette Ontology and *lo* to the Language Ontology. (2) the operation “INSERT DATA”, the two statements that allow the creation of the two properties. The two properties are shown in bold. (3) two comments above each statement.

All the variables written in italics *po:InstanceParent*, *po:InstanceSon*, *lo:sourceClass* and *lo:targetClass* will be replaced with concrete resources when the SPARQL Rule 1 is instantiated.

The rest of the SPARQL rules are shown with the same look as in Table 1.

Table 1. SPARQL Rule 1 – Integrate modelling elements from different modelling

<pre>PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#> PREFIX po:<http://fhnw.ch/modellingEnvironment/PaletteOntology#> PREFIX lo:<http://fhnw.ch/modellingEnvironment/LanguageOntology#> INSERT DATA { //enter parent relation between two palette Element instances (triple 1) <i>po:InstanceSon</i> po:paletteConstructHasParentPaletteConstruct <i>po:InstanceParent</i>. //enter subClassOf relation between two Modelling Element classes (triple 2) <i>lo:sourceClass</i> rdfs:subClassOf <i>lo:targetClass</i> . }</pre>
--

(4) Validate SPARQL rules syntactically.

The syntactic validation of the SPARQL rules were performed through the SPARQLer Update Validator. Figure 4 shows the screenshot that proves the syntactic validation of SPARQL Rule 1. The upper part of the figure contains the SPARQL rule described in Table 1, while the bottom of the figure shows the output. The output does not contain any errors, which means that SPARQL Rule 1 is syntactically correct.

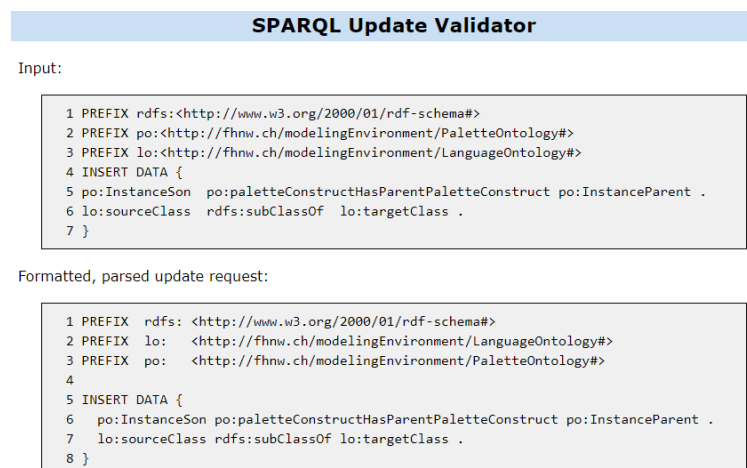


Figure 4: Syntactic Validation of SPARQL Rule 1

(5) Instantiate SPARQL rules.

To instantiate SPARQL Rule 1, the expected results from the use case concerning the integration between the BPMN Manual Task and the CMMN Discretionary Task is considered. That is, the Discretionary Task is added as a sub-class of the Manual Task. Figure 5 shows the conceptualisation of this use case. The two arrows in Figure 5 indicate the two properties that are expected to be added by the two statements (or triple) of the SPARQL rule instance. These

properties are the *rdfs:subClassOf* relation between the Discretionary Task and the Manual Task and the *po:hasParent* object property between the two instances of *po:PaletteElement*.

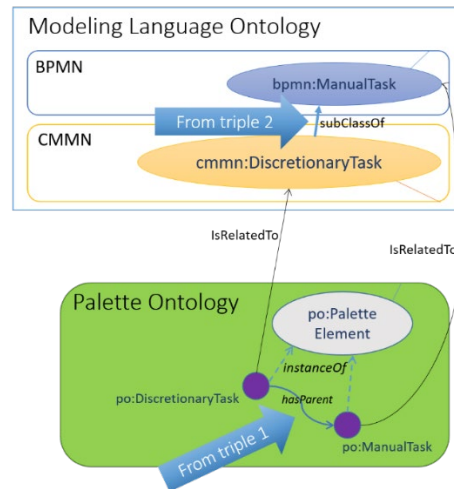


Figure 5: Conceptualisation of the use case for the integration of CMMN Discretionary Task with BPMN Manual Task

The instance of SPARQL Rule 1 is described in Table 3. The prefixes *bpmn* and *cmmn* on top of the rule instance replace the generic *lo* prefix seen in SPARQL Rule 1 (in Table 1). Both prefixes refer to two ontologies, reflecting the linguistic view of the BPMN and CMMN modelling languages, respectively.

Table 2. An instance of SPARQL Rule 1

```

PREFIX rdfs:<http://www.w3.org/2000/01/rdf-schema#>
PREFIX po:<http://fhnw.ch/modellingEnvironment/PaletteOntology#>
PREFIX bpmn:< http://ikm-group.ch/archimeo/BPMN#>
PREFIX cmmn:< http://ikm-group.ch/archimeo/CMMN#>
INSERT DATA {
//enter parent relation between the two palette Element instances (triple 1)
po:DiscretionaryTask po:hasParentPaletteConstruct po:ManualTask .
//enter subClassOf relation between the two Modelling Element classes (triple 2)
cmmn:DiscretionaryTask rdfs:subClassOf bpmn:ManualTask .
}

```

(6) Validate SPARQL rules semantically.

To validate the semantic correctness of SPARQL Rule 1, its instance in Table 2 was fired against the ontology. For this, the ontology editor TopBraid Composer⁶ was used. Figure 6 shows the result of the rule instance execution. As shown by the two arrows, the two triples contain the two new properties that have been added to the ontology. Hence, the Discretionary Task and the Manual Task are integrated as expected (consistent to the use case in Figure 5), which proves that SPARQL Rule 1 is semantically correct.

⁶ <https://allegrograph.com/topbraid-composer/>



Figure 6: Semantic validation of SPARQL Rule 1 - modelling elements integration

6. Conclusion

This paper presented a methodology for the ontology-based meta-modelling. The methodology of Grüninger and Fox [29] was adapted for the rigorous development and evaluation of semantic rules. The latter ensure the correct propagations of the language adaptations. In total, *11 SPARQL rules* were developed, each of which implements one or more operators. For space reasons, in this paper only the first semantic rule has been shown. The rest have already been implemented in the form of Java methods and are publicly available in a GitHub repository⁷.

As future work, I am investigating the use of the proposed methodology for the creation and formalization of user-friendly visual constraints to a SHACL [34] language. The aim is to enable domain experts to visually create, constrain and validate ontology-based models.

References

- [1] OMG, “Business Process Model and Notation (BPMN), Version 2.0.” Object Management Group OMG, 2011.
- [2] U. Frank, “Domain-specific modeling languages: Requirements analysis and design guidelines,” in *Domain Engineering*, I. . Reinhartz-Berger, A. . Sturm, T. . Clark, S. . Cohen, and J. Bettin, Eds. Berlin Heidelberg: Springer , 2013, pp. 133–157.
- [3] C. Natschläger, “Towards a BPMN 2.0 ontology,” in *Lecture Notes in Business Information Processing*, Nov. 2011, pp. 1–15, doi: 10.1007/978-3-642-25160-3_1.
- [4] K. Hinkelmann, E. Laurenzi, A. Martin, and B. Thönssen, “Ontology-Based Metamodeling,” in *Business Information Systems and Technology 4.0. Studies in Systems, Decision and Control*, Dornberger R., Ed. Springer, Cham, 2018, pp. 177–194.
- [5] K. Hinkelmann, A. Gerber, D. Karagiannis, B. Thoenssen, A. van der Merwe, and R. Woitsch, “A new paradigm for the continuous alignment of business and IT: Combining enterprise architecture modelling and enterprise ontology,” *Comput. Ind.*, vol. 79, pp. 77–86, 2016, doi: 10.1016/j.compind.2015.07.009.
- [6] P. Haase, D. M. Herzig, A. Kozlov, A. Nikolov, and J. Trame, “Semantic Web 0 (0) 1 metaphactory: A Platform for Knowledge Graph Management,” *Semant. Web J.*, 2019, Accessed: Sep. 07, 2022. [Online]. Available: <http://www.knime.com>.
- [7] E. Laurenzi, K. Hinkelmann, S. Izzo, U. Reimer, and A. van der Merwe, “Towards an Agile and Ontology-Aided Modeling Environment for DSML Adaptation,” in *Advanced Information Systems Engineering Workshops. CAiSE 2018. Lecture Notes in Business Information Processing*, Jun. 2018, pp. 222–234, doi: 10.1007/978-3-319-92898-2_19.
- [8] D. Karagiannis, “Conceptual Modelling Methods: The AMME Agile Engineering Approach,” in *Informatics in Economy*, G. Silaghi, R. Buchmann, and C. Boja, Eds. Springer, Cham, 2018, pp. 3–19.
- [9] U. Frank, “Domain-Specific Modeling Languages: Requirements Analysis and Design Guidelines,” in *Domain Engineering*, Berlin, Heidelberg: Springer Berlin Heidelberg, 2013, pp. 133–157.
- [10] D. Karagiannis, M. Lee, K. Hinkelmann, and W. Utz, *Domain-Specific Conceptual Modeling*. Springer

⁷<https://github.com/manulaur/OntologyBasedModellingEnvironment-WebService/blob/master/src/main/java/ch/fhnw/modeller/webService/ModellingEnvironment.java>

- International Publishing, 2022.
- [11] R. T. Burlton, R. G. Ross, and J. A. Zachman, "The Business Agility Manifesto Building for Change," 2017.
 - [12] OMG, "Meta Object Facility (MOF) Core Specification," Object Management Group, 2016.
 - [13] N. Guarino, G. Guizzardi, and J. Mylopoulos, "On the Philosophical Foundations of Conceptual Models," *Front. Artif. Intell. Appl.*, vol. 321, pp. 1–15, Dec. 2020, doi: 10.3233/FAIA200002.
 - [14] G. Guizzardi and J. Mylopoulos, "Taking It to the Next Level: Nicola Guarino, Formal Ontology and Conceptual Modeling," *Front. Artif. Intell. Appl.*, vol. 316, pp. 223–241, 2019, doi: 10.3233/978-1-61499-955-3-223.
 - [15] L. M. L. Delcambre, S. W. Liddle, O. Pastor, and V. C. Storey, "Articulating Conceptual Modeling Research Contributions," in *Advances in Conceptual Modeling. ER 2021. Lecture Notes in Computer Science*, vol. 13012, I. Reinhardt-Berger and S. Sadiq, Eds. Cham: Springer, 2021, pp. 45–60.
 - [16] W3C, "RDF Schema 1.1." <https://www.w3.org/TR/rdf-schema/> (accessed Apr. 13, 2023).
 - [17] R. A. Buchmann and D. Karagiannis, "Enriching linked data with semantics from domain-specific diagrammatic models," *Bus. Inf. Syst. Eng.*, vol. 58, no. 5, pp. 341–353, Oct. 2016, doi: 10.1007/S12599-016-0445-1/TABLES/1.
 - [18] M. Smajevic and D. Bork, "From Conceptual Models to Knowledge Graphs: A Generic Model Transformation Platform," in *MoDELS'21: ACM/IEEE 24th International Conference on Model Driven Engineering Languages and Systems (MODELS) – Tools & Demonstrations*, 2021, pp. 610–614, doi: 10.1109/MODELS-C53483.2021.00093.
 - [19] S. Bachhofner, E. Kiesling, K. Revoredo, P. Waibel, and A. Polleres, "Automated Process Knowledge Graph Construction from BPMN Models," in *Database and Expert Systems Applications. DEXA 2022. Lecture Notes in Computer Science*, 2022, vol. 13426 LNCS, pp. 32–47, doi: 10.1007/978-3-031-12423-5_3/FIGURES/5.
 - [20] U. Reimer and E. Laurenzi, "Creating and maintaining a collaboration platform via domain-specific reference modelling," in *EChallenges e-2014 Conference : 29-30 October 2014, Belfast, Ireland.*, 2014, pp. 1–9.
 - [21] M. Dumas, M. La Rosa, J. Mendling, and H. Reijers, *Fundamentals of Business Process Management*, 2nd ed. Springer-Verlag Berlin Heidelberg, 2018.
 - [22] C. Parry, E. Mahoney, S. A. Chalmers, and E. A. Coleman, "Assessing the Quality of Transitional Care," *Med. Care*, vol. 46, no. 3, pp. 317–322, Mar. 2008, doi: 10.1097/MLR.0b013e3181589bdc.
 - [23] R. Lenz and M. Reichert, "IT support for healthcare processes – premises, challenges, perspectives," *Data Knowl. Eng.*, vol. 61, no. 1, pp. 39–58, Apr. 2007, doi: 10.1016/j.datak.2006.04.007.
 - [24] E. Laurenzi, K. Hinkelmann, U. Reimer, A. Van Der Merwe, P. Sibold, and R. Endl, "DSML4PTM: A Domain-Specific Modelling Language for Patient Transferal Management," in *ICEIS 2017 - Proceedings of the 19th International Conference on Enterprise Information Systems*, 2017, vol. 3, pp. 520–531, doi: 10.5220/0006388505200531.
 - [25] E. Laurenzi, "An Agile and Ontology-Aided Approach for Domain-Specific Adaptations of Modelling Languages," University of Pretoria, 2020.
 - [26] D. Harel and B. Rumpe, "Modeling Languages: Syntax, Semantics and All That Stuff, Part I: The Basic Stuff," Weizmann Science Press of Israel, 2000.
 - [27] D. Karagiannis and H. Kühn, "Metamodelling Platforms," in *In Proceedings of the 3rd International Conference EC-Web 2002 -- Dexa 2002, Aix-en-Provence, France, 2002, LNCS 2455*, 2002, p. 182.
 - [28] E. Laurenzi, K. Hinkelmann, and A. van der Merwe, "An Agile and Ontology-Aided Modeling Environment," in *The Practice of Enterprise Modeling. PoEM 2018.*, Oct. 2018, pp. 221–237, doi: 10.1007/978-3-030-02302-7_14.
 - [29] M. Grüniger and M. S. Fox, "Methodology for the Design and Evaluation of Ontologies," *Ind. Eng.*, vol. 95, pp. 1–10, 1995.
 - [30] K. Hinkelmann, E. Laurenzi, A. Martin, D. Montecchiari, M. Spahic, and B. Thönssen, "ArchiMEO: A Standardized Enterprise Ontology based on the ArchiMate Conceptual Model," in *Proceedings of the 8th International Conference on Model-Driven Engineering and Software Development*, Mar. 2020, pp. 417–424.
 - [31] N. F. Noy and D. L. McGuinness, "Ontology Development 101: A Guide to Creating Your First Ontology," Palo Alto, 2001.
 - [32] W3C, "SPARQL Update - A language for updating RDF graphs," 2008.
 - [33] W3C, "SPARQL Query Language for RDF," 2008. .
 - [34] W3C, "Shapes Constraint Language (SHACL)," 2017. .