Ontology Alignment and System Models

Yuanwei Qu
University of Oslo
quy@ifi.uio.no

Eduard Kamburjan
IT University of Copenhagen
University of Oslo
eduard.kamburjan@itu.dk

Martin G. Skjæveland
University of Oslo
martige@ifi.uio.no

Arild Waaler
University of Oslo
arild@ifi.uio.no

Abstract

Ontologies and knowledge graphs are increasingly used alongside system models in MBSE. For example, ontologies allow integrating the models with other knowledge sources, and knowledge graphs enable validating system models using graph queries, shapes and logical consistency checks. Currently, there is no clear best practice on how and when to connect a system model with an ontology. In this work, we give an overview of three different approaches: (1) The ontology can be directly referred to from the system model, (2) the ontology is only connected after the system model is serialized into a knowledge graph using a default mapping, and (3) the ontology is referred to in the mapping itself. We discuss the advantages and disadvantages of each approach and use a system model for an IoT device from the Oslo Fjord Digital Twin, originally modeled in IMF. The main contribution of this work is a comparison and discussion about possibilities for ontology alignment in system models.

Keywords

Model-based system engineering, ontologies

Introduction

Model-based system engineering (MBSE) is an increasingly common approach to systems engineering (SE), concerned with the use of (semi-)formal models for the core aspects of system engineering: integration and traceability of design and operational information throughout the overall life cycle of an asset. It is underlying further innovation drivers like digital twins, where MBSE can serve as a source for the models to integrate data, knowledge and information through the digital thread.

Ontologies and knowledge graphs are among the approaches used for integration and modeling, in particular model validation. Ontologies encode domain knowledge through a logic-based foundation and are supported by a rich ecosystem of tools and methodologies. They focus on conceptual modeling, aiming to reuse already existing ontologies as much as possible, and describe a system as a graph structure in terms of the use concepts. It is the reuse of other ontologies that enables integration – if a community agrees on an ontology to use, data can be exchanged using an agreed upon vocabulary and conceptual framework. A knowledge graph is a graph model, which uses terminology from an ontology, and is constrained by its axioms.

However, combining ontologies with the (structural) system models otherwise prevalent in MBSE poses challenges: Expertise in both SE and ontologies is rare, and the focus of ontologies to reuse concept conflicts with the focus on describing the structure of a specific system design. Thus, merely transforming the system model into a graph model does not suffice to exploit ontologies. In this work, we compare three different approaches to combine ontological and system modeling, especially on how to *align* a structural model with a preexisting ontology. They are illustrated in Fig. 1.

Language-Based. The language-based approaches align at language-level: A modeling language is designed that uses the vocabulary of the ontology to design the system model.

Graph-Based. The graph-based approaches are

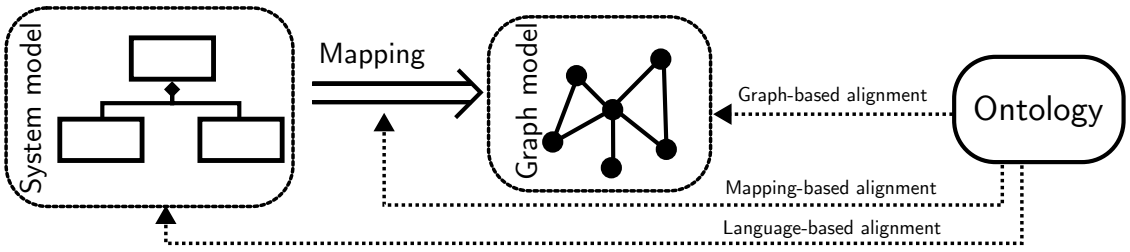


Figure 1. Comparison of three classes of approaches to integrate an ontology into a system model. Either this happens in the system modeling language itself (language-based approaches), in the mapping of the system model into a graph (mapping-based) or in the graph itself, based on a default mapping (graph-based).

based on classical ontology matching and alignment: The system model is designed in a non-ontological modeling language, e.g., SysML, and then translated into a graph model *using the terminology of the system modeling language*. In this form, either a knowledge graph or a language with language-based alignment, connection with the ontology is modelled.

Mapping-Based. The mapping-based approach translates a system model from the structural language *directly* in terms of the preexisting ontology. Here, the alignment is happening outside either modeling language.

In either case, a second challenge arises in aligning structural and ontological modeling: as MBSE aims to provide interdisciplinary designs, a structural model may need to be interpreted in different contexts, and consequently aligned with different ontologies: the alignment itself must be *configured*. For language-based approaches, this reduces to variability modeling in the language, while for alignment- and mapping-based approaches the variability management has to be handled by the used technologies.

In the following, we introduce the needed background in alignment and modeling, discuss the three aforementioned approaches to ontological alignment of structural system models and the consequences for variability management and configuration of alignments. We present one case study to illustrate the novel case of configurable mappings and present related work before we conclude.

Background

Ontologies, Matching and Alignment

An ontology is a formal, explicit specification of a shared conceptualization (Studer, Benjamins, & Fensel, 1998). More technically, it is a conceptual model describing the main entities of a domain, as well as their relation to each other, in a machine-readable format. Ontologies are used as a way to interoperate between different models and data within a domain, where all stakeholders agreed on an ontology.

Ontologies are artifacts that are independent of data and software, they are not data models, even though they can be used as a kind of data model for graph data. Designing ontologies is a collaborative task called *ontology engineering* (Poveda-Villalón, Fernández-Izquierdo, Fernández-López, & García-Castro, 2022; Tudorache, 2020), which has given rise to numerous software tools and ontology engineering methodologies. One task during ontology engineering is ontology alignment[1]. If a part of the domain, or an adjacent domain, is already expressed in an ontology, then the terms of the ontology-in-development must be related, or *aligned* to the existing ontology.

In this work, we do not touch on aligning an ontology and metamodels (Henderson-Sellers, 2011) directly, but on aligning system models and ontologies in the sense that the structural system model must be expressed both in terms of its metamodel and a given ontology. We call this task *structure alignment*, as it can be understood as an instance of ontology alignment in some cases: Which patterns and classes/blocks in the model correspond to a certain concept in the ontology. In other cases, it maps to other ideas in ontology engineering. We return to this discussion in the next chapter.

In this work, we restrict ourselves to the semantic web (Hitzler, Krötzsch, & Rudolph, 2010) technology stack, in particular RDF and OWL: The Resource Description Framework (RDF) is a model for graph models, which can be used to serialize other data formats, and in particular, ontologies. At its core, it defines a metamodel over triples, which together form a graph.

[1]Also called ontology matching. In this work, we use these terms interchangeably for the same task, regardless of its automation.

If the triples are used according to an ontology, it is a *knowledge graph*. The Web Ontology Language (OWL) is a logical language to express axioms over classes and their properties. If used over RDF, it can be used to define classes and their membership, using a special predicate, and to restrict valid graph models.

System Modeling Languages

System modeling language expresses different aspects of a system, and in the following we focus only on structural modeling, such as SysML's block definition diagrams.

Information Modeling Framework (IMF)

For examples we use the Information Modeling Framework (IMF) (Fjøsna & Waaler, 2021), a logic-based framework for describing how engineered systems appear under different contextual conditions. IMF provides the following structural primitives: (1) blocks representing system boundaries, (2) terminals representing interface points, and (3) connectors representing connections. Additionally, specialization and breakdown relations allow to describe a system decomposition, and attributes to annotate concrete data. A central feature of IMF is *aspect modeling*, inspired by ISO/IEC 81346: every element belongs to an aspect comprising *modality*, *information domain*, and *interest* parameters. This allows functional, physical, and spatial views of the same system to be expressed and related within a single formalism. Elements carry *classifiers* whose values may be drawn from external reference data libraries such as DEXPI, CFIHOS, or ISO 15926, or ontologies, which we discuss in detail below. While it does have a metamodel, the semantics of IMF is expressed in a mapping into a graph model. Thus, the meaning of an IMF model is a set of triple. This is in contrast to, e.g., SysMLv2, whose semantics are defined in terms of a meta-modeling language KerML. However, we point out that this knowledge graph is *not* connected to any ontology, but a serialization with constraints on patterns that may occur. Fig. 2 shows an example.

Ontological Modeling Language (OML)

The Ontological Modeling Language (OML) is a ontology and system modeling language with a textual syntax. It is also translated into an ontology, but is more expressive than IMF and covers most of OWL2-DL. Indeed, it can be used for both systems modeling and ontology engineering. Fig. 3 shows an example.

Declarative Mapping Languages

Declarative mapping languages describe mappings from structured data into graphs. For example, they describe how to translate tabular data, XML or a similar format into RDF. There are several languages available, e.g., OTTR (Skjæveland & Karlsen, 2024), and we use the RDF mapping language (RML) (Dimou et al., 2014) as a representative. RML defines mappings from tabular data or XML to knowledge graphs. A mapping is defined by an iterator over the input data, e.g., an XPath expression for XML, and a pattern that is instantiated for each element.

Structure Alignment

We define *structure alignment* as the following task:

> Given (1) a structural system model and (2) an ontology describing the concepts of the domain of this model, produce a semantically equivalent graph model in terms of the ontology.

Where semantically equivalent denotes that it has the same meaning to the users. We do not understand this equivalence in terms of a *formal* equivalence.

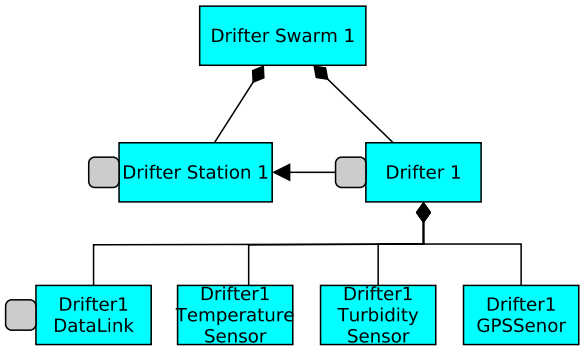


Figure 2. A drifter swarm in the OFDT consists of a station that serves as a gateway to relay communication, and at least one drifter, which has 3 sensors (temperature, turbidity, GPS) and a communication channel to the station.

As a running example, we use a *drifter* from the Oslo Fjord Digital Twin (OFDT) (Kamburjan et al., 2025) for which a simple IMF model is available and shown in Fig. 2. A drifter is a floating device to measure water temperature and turbidity in water bodies. This information is tagged with the current GPS location of the floater, and transmitted to a drifter station,

which acts as an Internet-of-Things (IoT) gateway. A drifter is also an IoT device. The structure alignment task is to produce a graph model in terms of the Smart Applications REFerence (SAREF) ontology (García-Castro et al., 2023; Lefrançois, 2023).

Language-Based

The language-based approach is to provide a language where the ontology can be imported and afterwards extended and referred to in the structural system model itself. The structure alignment task is reduced to a modeling task that combines structural and ontological modeling in one. OML exemplifies this approach: ontologies are explicitly referred to, and every relation and concept must be ontological.

The clear advantage of this approach is that it provides one language for both structure modeling and structure alignment. It indeed even provides the same language for ontological modeling. As OML and openCaesar demonstrate, monotlithic tooling can be avoided: existing semantic technologies can be reused, and meta-models can still be used to create and maintain extensive tooling.

OML

```
1 description <.../swarm1#> as swarm1 {
2   uses <...> as saref
3   uses <...> as s4sys
4   instance swarm : s4sys:System
5   instance station : s4sys:System [
6     s4sys:subSystemOf swarm ]
7   instance connection : s4sys:Connection [
8     s4sys:connectsSystemAt link1
9     s4sys:connectsSystemAt link2 ]
10  instance drift1 : saref:Device, s4sys:System [
11    s4sys:connectsAt link1
12    s4sys:subSystemOf swarm ]
13  instance link1 : s4sys:ConnectionPoint [
14    s4sys:subSystemOf drifter1  ]
15  instance link2 : s4sys:ConnectionPoint [
16    s4sys:subSystemOf station  ]
17  instance temp1 : saref:Sensor[
18    s4sys:subSystemOf drifter1
19    s4sys:FeatureOfInterest saref:Temperature ]
20  ...
21 }
```

RDF

```
1 @prefix swarm1: <...> . @prefix s4sys: <...> .
2 swarm1:swarm a s4sys:System.
3 swarm1:station a s4sys:System;
4                s4sys:subSystemOf swarm1:swarm.
```

Figure 3. The above OML model is a shortened representation of Fig. 2. The below are some of the triples generated for the `swarm` and `station` instances.

On the downside is that users have several roles with different requirements on their expertise and tooling: they must not only be experts in both systems engineering and ontology engineering, but they must also be trained in structural alignment and use a tool that is neither common in SE nor OE.

Figure 3 shows a, for brevity's sake slightly condensed, OML model for the OFDT drifter. Note that this model is *explicitly* referring to the SAREF ontology by importing it and declaring the instances used to describe the system. The model is formally equivalent to a graph model using the external OML semantics. An excerpt is shown at the bottom of Fig. 3. Another possibility is shown in line 24: the ontology is used directly and we easily add information to the system model without extending the domain of the system model. In this line, we add the information that the temperature sensor is indeed measuring temperature, reusing the feature of interest subontology of SAREF.

Graph-Based

The graph-based approach is the opposite of the language-based: instead of creating one language for all tasks (structural modeling, limited ontology design, structural alignment), it aims to reuse the available tooling for either. To this end, an existing structural modeling language, say SysML, is *serialized* into a knowledge graph. It is important that it is not in an ontological form yet, it still uses the concepts and vocabulary of the *structural modeling language*. Afterwards, this knowledge graph and the ontology are aligned using existing ontology engineering tools. Examples that use the graph-based approach are IMF and its toolkit, as well as some approaches for SysML.

The advantages of this approach are that it is easy to implement, as only a serialization must be provided, and that the split of the tasks maps to separate roles with distinct tool ecosystems and available training.

The main disadvantage is that the alignment task is overloaded: If it is performed by an ontology engineer, then it still requires expertise in systems engineering to interpret the serialized system model. Additionally, matching the serialized structural model is not the same task as classical ontology matching: its vocabulary is the one of its meta-model, not a different conceptualization of the same domain.

RDF

```
@prefix swarm1: <...> .
@prefix imf: <...> .
swarm1:swarm a imf:Block;
             imf:hasAspect imf:Product;
             rdfs:hasLabel "Drifter Swarm 1".
swarm1:station a imfBlock;
               imf:hasAspect imf:Product;
               imf:partOf swarm1:swarm;
               rdfs:hasLabel "Station 1".
```

OWL

```
s4sys:System SubClassOf:
   imf:Block and imf:hasAspect imf:Product
s4sys:subSystemOf SubPropertyOf: imf:partOf
```

RDF

```
swarm1:swarm a s4sys:System.
swarm1:station a s4sys:System;
               s4sys:subSystemOf swarm1:swarm.
```

Figure 4. The top RDF triples are a direct serialization of the IMF model in Fig. 2. The middle is the OWL alignment, i.e., the OWL axioms that allow to derive the bottom RDF triples that connect the IMF model serialization to the SAREF ontology.

Figure 4 shows the RDF resulting from serializing the IMF model of the OFDT drifter. Note that this graph model is completely independent of the SAREF ontology. The bottom of Fig. 4 shows the additional OWL axioms that are added to align the serialized IMF model with SAREF. We point out that the axioms are a general alignment, i.e., they identify *sets* of individuals with the SAREF classes, they do not just add information for this concrete system model – the alignment axioms are reusable. However, their formulation requires expertise in IMF.

Mapping-Based

Mapping-based approaches align during translation, i.e., they take a structural system modeling language and translate it into a knowledge graph using the domain ontology, or an ontological modeling language. This is exemplified by parts of the IMF toolbench and some approaches for SysML. There are two subvariants of mapping-based approaches. One can either map directly into a knowledge graph, or into a language with language-based alignment. In the former case, it is an instance of *knowledge graph construction*. In the latter case, it is an instance of *model-to-model transformations*. We focus on the knowledge graph construction case in the following, for the transformation we refer to Nakajima et al. (2025).

A mapping-based approach can be exogenous or endogenous. An exogenous mapping is a mapping that is outside the model. An endogenous mapping is described inside the model.

RML

```
mappings:
  product-blocks:
    sources:
      - access: OFDT.graphml
        referenceFormulation: xpath
        iterator: "xpath/for/product/blocks"
    s: https://...$(@id)
    po:
      - [a, s4sys:System]
      - p: rdfs:label
        o:
          value: $(data/ShapeNode/NodeLabel)
          datatype: xsd:string

  partOf:
   sources:
     - access: OFDT.graphml
       referenceFormulation: xpath
       iterator: "xpath/for/partOf/between/
product/blocks"
   s: https://...$(@source)
   po:
     - p: s4sys:subSystemOf
       o:
         value: https://...$(@target)
         type: iri
```

RDF

```
swarm1:xy12 a s4sys:System;
            rdfs:hasLabel "Drifter Swarm 1".
swarm1:xy23 a s4sys:System;
            s4sys:subSystemOf swarm1:xy12;
            rdfs:hasLabel "Station 1".
```

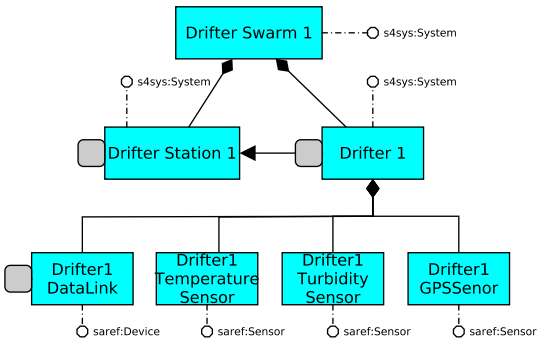
Figure 5. The top code are two RML mapping in YARRRML syntax that operate on the XML serialization of IMF in GraphML. The bottom RDF are some of the triples generated for the swarm and station blocks.

Exogenous Approaches. Figure 5 shows the RML mapping that maps the IMF model of the OFDT drifter from XML to directly aligned. Note that this mapping is using the SAREF ontology directly, it does not use any IMF vocabulary. The bottom of Fig. 5 shows an excerpt of the resulting graph model.

The advantages and disadvantages are similar to graph-based approaches: System engineers are freed from ontological concerns, at the cost of needing to do alignment in a non-standard setting. However, we argue that declarative mapping languages are easier to configure than alignment based on a

given knowledge graph. In particular, they are easier to scope. While an alignment using OWL axioms expresses general knowledge, a mapping can be fine-tuned more easily for the specifics of a model.

Endogenous Approaches. An endogenous approach, in contrast, uses special model elements to configure the mapping. For example, consider the IMF model in Fig. 6. It contains additional elements, so-called classifiers, that during generation automatically add the alignment axioms.For a product block, IMF interprets the annotated classes as new *membership* predicates. For functional blocks, they are interpreted as *superclasses*. Note that the axioms connecting the IMF `partOf` relation with SAREF's `subSystemOf` must still be added manually, as these annotations are not supported so far in IMF for edges.



RDF

```
swarm1:xy12 a s4sys:System;
            rdfs:hasLabel "Drifter Swarm 1".
swarm1:xy23 a s4sys:System;
            rdfs:hasLabel "Station 1".
```

Figure 6. The example from Fig. 2 (top), annotated with classifiers to configure the mapping process, and the resulting RDF (bottom).

In this form, mapping-based approaches lean towards language-based approaches, and have the advantage that they can be used on top of established system modeling languages. Their disadvantage is that they are less flexible and expressive — the above example works well for elements that have a singular edge connecting them with the ontology, but cannot express more complex patterns like exogenous approaches. If it supported them, the additional training connected with a new declarative mapping language would have to be added.

Configuring Semantics

MBSE is an inheritably interdisciplinary field and as such each system model, especially in the early stages of design where early verification and validation (V&V) is a main advantage of using model, can be interpreted in different ways. This requires not only aligning structures, but also managing the variability between different structure alignments. We define *structure alignment configuration* as follows:

> Given (1) a structural system model and (2) a set of ontology describing the concepts of different domains where this model can be interpreted, produce a semantically equivalent system model for each of the ontologies.

For language-based approaches, this is reduced to standard variability management, which can be either internally or externally. To the best of our knowledge, no such language implements internal variability management such as delta-oriented programming or variability modules, but we see no specific challenge for such techniques either. Note that as such language do include modularity and reuse mechanisms, variability can already be managed using available language features.

For alignment-based approaches, variability management has not been considered to the best of our knowledge, even though it is common to maintain and export different versions of an ontology, and tools like the ontology development kit Matentzoglu et al. (2022) can be the basis for external variability management systems.

For mapping-based approaches, variability management has also not considered. However, in contrast to alignment-based approaches, the used languages already include reuse and dependency mechanisms which can be used to internally manage variability even in the absence of, e.g., feature models.

Mapping-based approaches combine the separation-of-concerns in modeling from alignment-based approaches with the ability to perform internal variability management with existing languages and language features.

For exogenous approaches, consider the case if the OFDT model has to be aligned with another ontology, for example, the Industrial Data Ontology (IDO).[2] In this case, a second mapping, as shown in Fig. 7

[2]Industrial Data Ontology is part of ISO/DIS 23726-3 Automation systems and integration — Ontology based interoperability,

RML

```
mappings:
  product-blocks:
    sources: ...
    s: https://...$(@id)
    po:
      - [a, ido:System]
      - p: rdfs:label
        o: ...
  partOf:
   sources: ...
   s: https://...$(@target)
   po:
     - p: ido:hasPart
       o:
         value: https://...$(@source)
         type: iri
```

RDF

```
swarm1:xy12 a ido:System;
            ido:hasPart swarm1:xy23;
            rdfs:hasLabel "Drifter Swarm 1".
swarm1:xy23 a ido:System;
            rdfs:hasLabel "Drifter Station 1".
```

Figure 7. A second RML mapping for Fig. 2, which maps its serialization into terms of IDO.

For endogenous approaches, IMF supports to use multiple classifiers on each structural element. Generation of the knowledge graph then either generates all triples, or only those for a chosen ontology, resp., reference data library.

Discussion and Take Aways

We have compared and discussed different approaches to align a system model with an ontology. Each of the discussed approaches is viable, has been used in practice, and is supported by tooling. The main difference is indeed the nature of the tooling and how it can be integrated into the workflow of the engineers for a given project.

The main take-away for practitioners, beyond the viability of all approaches, is that the trade-off is between *integration of ontology engineering practices into systems engineering* and *availability of tooling*.

The richest tooling is available for graph-based approaches and exogenous mapping-based approaches, but ontology engineering practices are disjoint from the system modeling itself. This is not necessarily a disadvantage, as alignment and modeling may be two different roles with certainly different requirements on training, but requires additional coordination between the two tasks.

Language-based approaches and endogenous mapping-based approaches rely on novel languages or language extensions. Both languages we are aware of that realize this natively, IMF and OML, are supported by a toolkit and are actively maintained, but are too novel to have a big support in terms of community effort at the time of writing.

We also hope that our case study helps to illustrate the difference on a practical level.

It is worth pointing out that structural alignment shares similarities with ontology alignment, but is not a strict subset. For example, the language-based approach of OML directly relates ontological classes, and the graph-based alignment uses OWL axioms to express relations. However, mapping-based approaches have the relation between system model-classes and ontology-classes implicit as part of the mapping – in the endogenous approach of IMF, the individuals of product blocks are related to OWL classes, with is a membership relation, not one between different classes. The reason is that the mapping has an implicit relation between the patterns where the classifiers are added, which is not explicit in the generated graph.

Related Works

Mens and Gorp (2005) give a taxonomy of model transformation. Alignment can be seen as a model transformation, and as discussed for mapping-based approaches is also realized as such. Ontologies and metamodels are closely related, see e.g., the work of Henderson-Sellers (2011), but subtle differences in their purpose and usage has an effect of the tools and methodologies in the respective field: metamodels focus on domain modeling for data in a software system, while ontologies are software-independent, conceptual models (Gray & Rumpe, 2022). Ontology alignment is an inherently *conceptual* task that is concerned with *domain analysis*, not modeling *per se*. Consequently, model transformations can be used to realize and automate system alignment, but they cannot be the sole tool to do so and must be connected to an overall methodology. Such tools and methodologies are surveyed by Portisch et al. (2024) and Thiéblin et al. (2020), and we refer to their works for an overview. Similarly, there are numerous works on the use of ontologies in systems engineering, especially in the recent paradigms on digital twins and digital engineering, where we refer to the survey of

Karabulut et al. (2024) and the recent work of Gregory et al. (2025) and Pigazzi et al. (2025).

Conclusion

We have presented the different possibilities and choices for connecting system models with ontologies, and gave additional case studies for the mapping-based approaches, which are explored less than language- and graph-based approaches. We hope that this work can contribute to the increasing interconnection of systems engineering and knowledge representation. For future work, it remains to conduct user studies to gain empirical data on the perceived differences between the three approaches by practitioners.

Acknowledgments

This work is partially supported by the EU Commission funded projects SM4RTENANCE (101123490) and Tec4MaaSEs (101138517).

AI Assistance Disclosure

ChatGPT (GPT-5.2) assisted with grammar correction and minor textual editing. All modeling, analysis and conclusions are original work by the authors.

References

Dimou, A., et al. (2014). RML: A generic language for integrated RDF mappings of heterogeneous data. In *LDOW* (Vol. 1184). CEUR-WS.org.

Fjøsna, E., & Waaler, A. (2021). *READI Information modelling framework (IMF). Asset Information Modelling Framework* (Tech. Rep.). READI.

García-Castro, R., et al. (2023). The ETSI SAREF ontology for smart applications. *Energy Smart Appliances*, 183–215.

Gray, J., & Rumpe, B. (2022). On the relationship between models and ontologies. *SoSyM*, *21*(4). doi: 10.1007/S10270-022-01021-0

Gregory, J., et al. (2025). Towards a digital engineering ontology to support information exchange. *INCOSE International Symposium*, *35*(1), 37-52. doi: https://doi.org/10.1002/iis2.70073

Henderson-Sellers, B. (2011). Bridging metamodels and ontologies in software engineering. *J. Syst. Softw.*, *84*(2). doi: 10.1016/J.JSS.2010.10.025

Hitzler, P., Krötzsch, M., & Rudolph, S. (2010). *Foundations of semantic web technologies*. Chapman and Hall/CRC Press.

Kamburjan, E., et al. (2025). Towards self-adaptive data management in digital twins for biodiversity monitoring. In *EDTConf*. ACM.

Karabulut, E., et al. (2024). Ontologies in digital twins: A systematic literature review. *Future Gener. Comput. Syst.*, *153*, 442–456. doi: 10.1016/J.FUTURE.2023.12.013

Lefrançois, M. (2023). SAREF4SYST: a SAREF reference ontology pattern for representing systems and their interconnections. In *WOP* (Vol. 3636). CEUR-WS.org. Retrieved from `https://ceur-ws.org/Vol-3636/paper10.pdf`

Matentzoglu, N., Goutte-Gattat, D., Tan, S. Z. K., Balhoff, J. P., Carbon, S., Caron, A. R., … Osumi-Sutherland, D. (2022). Ontology development kit: a toolkit for building, maintaining and standardizing biomedical ontologies. *Database J. Biol. Databases Curation*, *2022*(2022).

Mens, T., & Gorp, P. V. (2005). A taxonomy of model transformation. In *GRaMoT*. Elsevier.

Nakajima, Y., et al. (2025, October). Power of a Reasoner: Model Validation for SysML Model using openCAESAR . In *onto:NEXUS@models*. IEEE. doi: 10.1109/MODELS-C68889.2025.00112

Pigazzi, R., et al. (2025). Ontological approaches to model engineering qualities and values in dolce owl. *Applied Ontology*, *20*(3), 226-253. doi: 10.1177/15705838251367131

Portisch, J., et al. (2024). Background knowledge in ontology matching: A survey. *Semantic Web*, *15*(6), 2639-2693. doi: 10.3233/SW-223085

Poveda-Villalón, M., Fernández-Izquierdo, A., Fernández-López, M., & García-Castro, R. (2022). LOT: an industrial oriented ontology engineering framework. *Eng. Appl. Artif. Intell.*, *111*, 104755. doi: 10.1016/J.ENGAPPAI.2022.104755

Skjæveland, M. G., & Karlsen, L. H. (2024). The reasonable ontology templates framework. *TGDK*, *2*(2), 5:1–5:54.

Studer, R., Benjamins, V. R., & Fensel, D. (1998). Knowledge engineering: Principles and methods. *Data Knowl. Eng.*, *25*(1-2), 161–197. doi: 10.1016/S0169-023X(97)00056-6

Thiéblin, E., et al. (2020). Survey on complex ontology matching. *Semantic Web*, *11*(4), 689-727. doi: 10.3233/SW-190366

Tudorache, T. (2020). Ontology engineering: Current state, challenges, and future directions. *Semantic Web*, *11*(1). doi: 10.3233/SW-190382