ing.grid preprints

RESEARCH ARTICLE

Creating application-specific metadata profiles while improving interoperability and consistency of research data for the engineering sciences

Nils Preuß [1], Matthias Bodenbenner [2], Benedikt Heinrichs [3], Jürgen Windeck [1], Mario Moser [2], Marc Fuhrmans [1]

1. University and State Library Darmstadt, Technical University of Darmstadt, Darmstadt.
2. WZL, RWTH Aachen University, Aachen.
3. IT Center, RWTH Aachen University, Aachen.

Date Submitted:
2024-08-23

Keywords:
RDF, SHACL, application profiles, metadata, FAIR data, data modeling, mechanical engineering

Data availability:

Software availability:

Corresponding Author:
Nils Preuß
nils.preuss@tu-darmstadt.de

Abstract. Due to the heterogeneity of data, methods, experiments, and research questions and the necessity to describe flexible and short-lived setups, no widely used subject-specific metadata schemata or terminologies have been established for the field of engineering (as well as for other disciplines facing similar challenges). Nevertheless, it is highly desirable to realize consistent and machine-actionable documentation of research data via structured metadata.

In this article, we introduce a way to create subject specific RDF-compliant metadata profiles (in the sense of SHACL shapes) that allow precise and flexible documentation of research processes and data. We introduce a hierarchical inheritance concept for the profiles that we combine with a strategy that uses composition of relatively simple modular profiles to model complex setups. As a result, the individual profiles are highly reusable and can be applied in different contexts, which, in turn, increases the interoperability of the resulting data. We also demonstrate that it is possible to achieve a level of detail that is sufficiently specific for most applications, even when only general terms are available within existing terminologies, avoiding the need to create highly specific terminologies that would only have limited reusability.

1 Introduction

A lot of resources and effort is put into conducting scientific experiments in the lab or the field, generating large amounts of highly heterogeneous data. However, without adequate documentation of additional information, e.g., what the data represents and how it was obtained, the data can easily become useless. In this article, we present an approach to document research data in a flexible and precise way that is also highly interoperable and machine-actionable and suitable to embed data into semantic knowledge graphs in the sense of the resource description framework (RDF [1]).

Having structured and consistent metadata available is very beneficial in the earlier stages of the research data life cycle, while research data is still primarily stored locally and in active use within the project it was generated by. Structured metadata is a prerequisite for any automation attempts. Using a standardized language is key for automated validation and quality control. It supports the local data organization by allowing computerized workflows, and researchers also benefit from easier findability in large amounts of data. In addition, it enables machine learning approaches and is also one of the key factors in making research data FAIR [2] allowing the reuse of expensively produced data..

Since all of these goals can be accomplished best when metadata is highly interoperable and machine-actionable, semantic metadata, i.e., expressing information via well-defined, unambiguous terms represented by unique IDs, is considered most valuable [3]. The usage of such controlled vocabularies that themselves follow the FAIR-principles is paramount for the implementation of FAIR scientific data.

To this end, researchers, scientific communities and institutions are making ever-increasing efforts to leverage semantic web standards and ontologies to enable semantic, machine-actionable metadata describing the contents of their datasets.

Unfortunately, due to the heterogeneity of data, methods, experiments and research questions, and the necessity to describe flexible and short-lived setups [4], no widely used subject-specific metadata schemata or terminologies have been established for the field of engineering.

1.1 State of the art

A consensus on common standards for mechanical engineering vocabularies and information models remains elusive, even within less heterogeneous research communities. In many cases, these efforts even result in at least partially redundant vocabularies or ontologies. Typically designed for specific use cases, they feature a low degree of compatibility with other vocabularies or transferability onto similar use cases. This, of course, complicates the process of achieving a consensus regarding (quasi-) standards for the interoperable description of contents in scientific datasets.

Elaborate, well-designed vocabularies do exist, however, mostly in the form of (i) natural language texts like books and articles or reports or (ii) structured, therefore machine-readable data, but following custom or even proprietary schemas not trivially compatible with semantic web standards, imposing a high barrier to entry. This is a common occurrence with industry-standards such as eCl@ss, OPC UA, DEXPI, etc. [5], [6] As of the time of writing, although the organizations maintaining the aforementioned standards state they are committed to publish their standards using semantic web formats, none are available as such.

As a result, research data management in the field of mechanical engineering is typically based on simple file systems and relies on manual organization of directories, files, and metadata. Data and metadata are often created on a case-by-case basis and stored separately, inconsistently and untraceably [7] [8]. The created metadata are in many cases not even really metadata in the sense of being machine actionable auxiliary information about distinct datasets. These circumstances diminish the information value of research data and hinder the development of reusable tools for metadata creation or automation of workflow steps relying on metadata [9].

In Germany, these issues are currently being addressed by NFDI4Ing (National Research Data

Infrastructure for the Engineering Sciences), a consortium which provides engineers with research data management (RDM) services. Services are developed in a matrix organization with viewpoints of several engineering disciplines as well as research methods, both supported by overarching working groups. Within NFDI4Ing, efforts were also undertaken to create a basis for a semantic description of research in the engineering domain, resulting in the Metadata4Ing (m4i [10]) ontology, that aims at a process-based description of research activities and their results, focusing on the provenance of both data and material objects and provides highly applicable concepts like processing steps, in- and output, employed methods and tools, that we were able to reuse in our efforts.

1.2 Our approach

In order to facilitate use of semantic metadata within engineering, we have developed an approach to define flexible and specific metadata schemata that are nevertheless highly interoperable and reusable. The metadata schemata are realized on the basis of so-called application profiles or SHACL shapes [11], and will be referred to as metadata profiles in this article.

Researchers can utilize such metadata profiles as a target format to guide the creation of metadata in their research workflows, as well as to validate the conformity of generated metadata. Our approach allows researchers to create metadata specific to their use case, while maintaining conformity to existing standards and vocabularies, as well as reusing and extending those profiles for similar applications.

The main contribution of this article is a set of best practices and modeling techniques which

- allow the implementation of metadata schemata as application profiles
- support a modular and hierarchical design
- maximize the potential of achieving metadata interoperability through the reuse of existing terms and controlled vocabularies
- avoid ad hoc definition of poorly designed custom terms or new vocabularies

Our approach to achieve those objectives is based on four underlying concepts partially borrowed from object-oriented programming: inheritance, composition of modularly designed elements, combination of existing sources, and specificity via restriction of general concepts [12]. It relies heavily on SHACL [11] features to implement dependencies in profiles instead of vocabularies, avoiding any need for creation or adaptation of vocabulary or ontology graphs.

The article is structured as follows: We start describing a typical application scenario, background and resulting challenges (Section 2), discuss the modeling approach in general, relevant standards and our design choices (Section 3), introduce our developed modeling techniques and qualitative validation (Sections 3.1 - 3.5) and conclude with a discussion of our solution, open issues and future developments (Section 4).

2 Application scenario

Consider the following application example of experimental research in engineering sciences: Figure 1 shows a typical experimental setup, i.e., a technical system equipped with additional sensors, actuators and other components to induce a specified operational state and to study the resulting behavior of the system. In this case, *the system under test*, a hydraulic circuit, is used to operate different pumps (*units under test*) and investigate their operating behavior and ultimately, their efficiency.

Along with the acquired raw signal data, various descriptive metadata elements must be present to document the experiments carried out, as well as their results, so that the created data remains findable and interpretable. This includes, but is not limited to, measured and actuated quantities, as well as the utilized equipment and its properties. As stated in the introduction, a recurring challenge to documenting this metadata in a machine-actionable and interoperable way is that researchers need the ability to describe very heterogeneous setups and a large range of hard- and software components. However, the metadata profiles that formalize those individual combinations and allow the standardization and validation of the corresponding metadata must be as reusable as possible, since reuse establishes consistency and interoperability. Typical avenues for facilitating reuse are i) inheritance and ii) composition. Furthermore, the metadata profile concept must allow for iii) combination and alignment of both common and more specific terms stemming from different, often non-aligned terminology sources. Lastly, they must provide a means to achieve iv) specificity for their respective application targets, despite the widespread lack of suitable terms.

The following sections introduce the overall modeling approach, as well as the developed modeling techniques for each of those four core concepts, using a simplified subset of the application scenario outlined in Fig. 1, consisting of measurements using up to two of the deployed sensors, one temperature sensor and one pressure sensor, as well as the respective observed quantities.

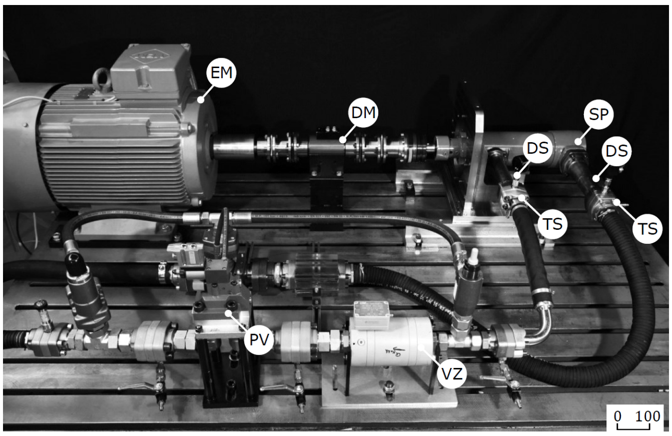

Figure 1: Test rig for the experimental investigation of the efficiency of screw pumps (positive displacement pumps). EM: electric motor, DM: torque- and rotational speed measuring shaft, SP: screw pump, DS: pressure sensor, TS: temperature sensor, VZ: volume flow sensor, PV: proportional pressure valve.

3 Modeling approach

Our modeling approach relies on semantic technologies, specifically on the Resource Description Framework (RDF [1]), that expresses information by subject-predicate-object triples assembled from controlled terms taken from ontologies, and the Shapes Constraint Language (SHACL [11]), that allows defining metadata profiles by placing requirements and restrictions on the triples for the entity that is supposed to be described. Such a metadata profile could, e.g., state that an entity of the kind *Sensor* must have a *serialnumber* attribute of the type *string*, and may have one or more *observes* attributes that are only allowed to refer to entities that satisfy the metadata profile defined for *Property*.

The newly proposed modeling approach is based on four underlying concepts: inheritance, composition of modularly designed elements, combination of existing sources, and specificity via restriction of general concepts, which will be presented in Sections 3.1 - 3.4. For each of the four concepts, we describe the goal we want to achieve, the challenges one faces when trying to reach the goal with existing methods, and our solution accompanied by an example for each of the concepts. Fig. 2 gives an overview of the simplified application example as well as the proposed use of the core concepts, i.e., the respective modeling techniques.

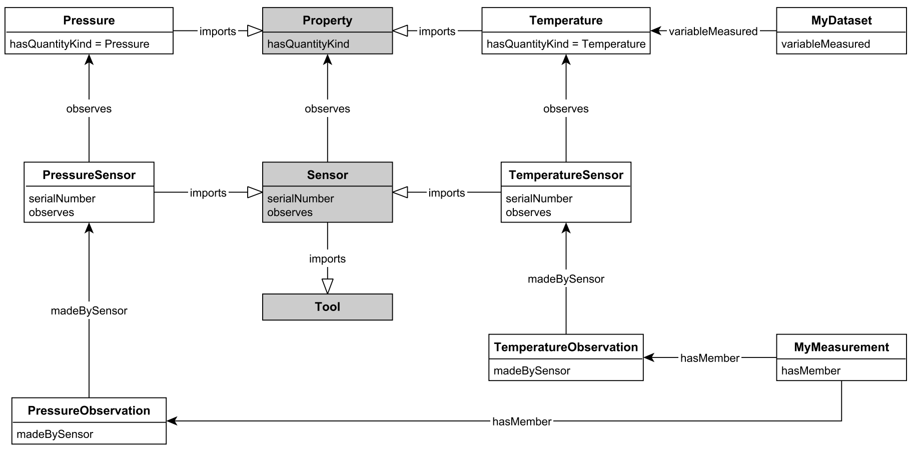

Figure 2: Diagram of metadata profiles for the simplified application scenario, on which the proposed core concepts and developed modeling techniques are demonstrated

In contrast to rule- or reasoning-based formalizations of information models, the proposed approach is based on metadata profiles, formalized as SHACL shapes. As described above, this allows the definition of restrictions and constraints for select parts of an RDF-based data graph.

One of the overarching challenges imposed by the goal to avoid definition of new vocabulary terms as much as possible, is the problem of targeting, i.e. controlling which of the nodes in a data graph a metadata profile is applied to. Throughout the following sections presenting the core concepts of the modeling approach, appropriate options to solve this are discussed. In doing so, we avoid the pragmatic but semantically meaningless approach of defining "hybrid" profile-classes that simultaneously serve as `sh:NodeShape` and `rdfs:Class` for the sole purpose of being able to declare nodes in the data graph as instances of the hybrid profile-class so that the it targets the node (implicit targeting). In general, we propose to rely on relationships on the

prefix	namespace
sh:	http://www.w3.org/ns/shacl#
rdfs:	http://www.w3.org/2000/01/rdf-schema#
dcterms:	http://purl.org/dc/terms/
owl:	http://www.w3.org/2002/07/owl#
schema:	http://schema.org/
sosa:	http://www.w3.org/ns/sosa/
qudt:	http://qudt.org/schema/qudt/
quantitykind:	http://qudt.org/vocab/quantitykind/
m4i:	http://w3id.org/nfdi4ing/metadata4ing#
ex:	http://www.example.org/

Table 1: Namespace prefix bindings for vocabularies used throughout this article.

profile level for targeting because those relationships are part of the metadata profiles we define and therefore under our direct control, whereas sufficient relationships on the level of existing vocabularies are often missing and not easily defined.

Table 1 states the utilized existing vocabularies, their namespaces as well as the prefixes used throughout this article.

3.1 Implementing inheritance between application profiles

In order to document research data precisely, a metadata profile must include all relevant information. Nevertheless, we want to avoid creating idiosyncratic profiles for each new research method or setup. Related metadata profiles need to be compatible and interoperable with each other to foster reusability of common specifications, which at the same time reduces redundancy between separate, but related metadata profiles.

A way to accomplish this is implementing inheritance, i.e., a hierarchical modeling approach in which a parent metadata profile contains common requirements and a child metadata profile contains only more specific requirements to avoid redundancy and enable reusability. An instance of the child profile has to fulfill all requirements, the common ones of the parent and the specific ones of the child. This way, general metadata profiles form the basis for more specific derived children, all of which are compatible to each other on the level of their closest shared parent.

In addition, duplicate definition of requirements shared by related metadata profiles is avoided. Profiles for a *temperature sensor* and a *pressure sensor* can be modeled as children of a more general *sensor* containing common requirements. At the same time, the reusability of metadata profiles is maximized, as researchers can always select the most fitting existing profile and either reuse it as is, or use it as a basis to derive a child profile according to their more specific needs.

On the technical side, this means enabling researchers to define additional requirements for already existing more general metadata profiles (potentially created by other researchers) or refine existing requirements, typically making them more narrow.

However, trying to accomplish this with the mechanisms provided by RDF, RDFS, and SHACL is not as straightforward as one might expect. The native approach would be to use the built-in inheritance mechanism of RDFS [13] to create child metadata profiles. This approach is not feasible in practice, as it requires a one-to-one correspondence between metadata profiles and target classes, which, given that we are defining metadata profiles with the intention of specifically defining a method or tool used in an engineering setup, cannot be expected. To illustrate this, consider the following example: There is a general metadata profile `ex:SensorProfile` which is targeting the RDFS-class `ex:Sensor`. Following the built-in approach, a more specific metadata profile `ex:TemperatureSensorProfile` targets a more specific RDFS-class `ex:TemperatureSensor`, as illustrated in listing 1. Doing so, at some point of desired specificity (which due to a lack of subject-specific terminologies will be very early for many entities at the time of writing this article) no suitable term exists that could be reused as a target class, which would require to introduce a new term, which would then either be an uncurated user-defined custom term (which should be avoided) or require a complex and slow curation process, which defeats our purpose of giving researchers an option to quickly define profiles to create consistent and quality-checked metadata for documenting their research data. In addition, this approach would require the classes and the hierarchical relations between them. It becomes very challenging if classes from multiple vocabularies must be combined. In the latter case, one has to do a manual alignment, which must be explicitly defined as an RDF vocabulary.

```
ex:TemperatureSensorProfile
    a sh:NodeShape ;
    sh:targetClass ex:TemperatureSensor ;
    sh:property [
        # constraints for temperature sensors
    ] .
```

Listing 1: Metadata profile illustrating direct targeting: The profile is applied to all entities that claim membership of the class `ex:TemperatureSensor`. Validation example available at https://doi.org/10.48328/tudatalib-1163, https://s.zazuko.com/usyRnn.

Proposed solution

Instead of defining a specific target class, i.e., modeling the hierarchy in the data graph, we represent inheritance by importing the parent metadata profile with a common target class into the child profile via `owl:imports` and the node constraint `sh:node`, i.e., we model the hierarchy in the SHACL shape graph. Thus, inheritance can be modeled even if this relationship has not been explicitly defined elsewhere or if the parent class is not available or stated.

```
ex:SensorProfile
    a sh:NodeShape ;
    sh:targetClass sosa:Sensor ;
    sh:property [
        sh:path schema:serialNumber ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [
        sh:path sosa:observes ;
        sh:class qudt:Quantity ;
    ] .
```

```
ex:TemperatureSensorProfile
    a sh:NodeShape ;
    sh:node ex:SensorProfile ;
    owl:imports ex:SensorProfile ;
    sh:property [
        sh:path sosa:observes ;
        sh:node [
            sh:property [
                sh:path qudt:hasQuantityKind ;
                sh:hasValue quantitykind:Temperature ;
            ] ;
        ] ;
    ] .

ex:PressureSensorProfile
    a sh:NodeShape ;
    sh:node ex:SensorProfile ;
    owl:imports ex:SensorProfile ;
    sh:property [
        sh:path sosa:observes ;
        sh:node [
            sh:property [
                sh:path qudt:hasQuantityKind ;
                sh:hasValue quantitykind:Pressure ;
            ] ;
        ] ;
    ] .
```

Listing 2: Metadata profile illustrating inheritance. Validation example using direct targeting available at `https://doi.org/10.48328/tudatalib-1164`, `https://s.zazuko.com/xTQM4G`.

Listing 2 illustrates this approach. The `sh:node` statement causes all property restrictions in the parent metadata profile to also be included in the child profile. The `owl:imports` statements has no direct effect by itself, but is required to tell any applications using the metadata profiles that `SensorProfile` needs to be loaded into the graph, whenever `TemperatureSensorProfile` is loaded.

Note that the `sh:targetClass` statement is not inherited by the child metadata profile, which is beneficial if there are different instances of the class in the data, not all of which are supposed to be validated against the child metadata profile. Referring back to the example, there might also be non-temperature sensors present in the data (e.g. `ex:PressureSensorProfile`) that are of the `sosa:Sensor` class but should not be validated against the `ex:TemperatureSensorProfile` profile.

However, this requires targeting of the `ex:TemperatureSensorProfile` to be determined by alternative, more indirect ways than using sh:targetClass as demonstrated in listing 1. In conjunction with node constraints defined by a *wrapper profile*, metadata profiles can be applied to data without specifying an explicit target class. Listing 3 shows an example where a metadata profile for a temperature sensor is applied without relying on a corresponding target class. Instead, a sosa:Observation's sosa:madeBySensor attribute receives a node constraint that restricts the attribute's target to satisfy the ex:TemperatureSensorProfil profile, which is thereby applied to it without stating a target class of its own. It is important to note, however, that the initial application of the (composite) metadata profile that contains the node constraints still requires a target class. In our experience, this requirement can realistically be met by using a class that is unique within the scope of data the metadata profiles are used on. Listing 3, e.g., assumes that there is only one kind of observation present in the data, which is often the case. If this is not true, the problem can be solved by introducing another layer in the data that includes the different kinds of observations (or other classes for which instances with different restrictions are present in the data) via the help of sh:qualifiedValueShape, as discussed in Section 3.4.

```
ex:TemperatureObservationProfile
    a sh:NodeShape ;
    sh:targetClass sosa:Observation ;
    sh:property [
        sh:path sosa:madeBySensor ;
        sh:node ex:TemperatureSensorProfile ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    owl:imports ex:TemperatureSensorProfile .
```

Listing 3: Metadata profile illustrating indirect targeting: Each sosa:Observation in the data has to have exactly one sosa:madeBySensor attribute pointing to a node that fulfills all requirements specified in the ex:TemperatureSensorProfile. Validation example available at https://doi.org/10.48328/tudatalib-1165, https://s.zazuko.com/kkpL8N.

3.2 Implementing modularity and composition

Another means of increasing reusability of metadata profiles is to use a modular modeling approach, in which separable aspects of a setup are modeled via separate, modular metadata profiles. The modular metadata profiles have a higher reusability than metadata profiles that simultaneously model multiple aspects within a single profile, even when they are highly specific. The modular metadata profiles can be reused in different contexts, which again avoids duplicate definition of restrictions, and can be flexibly combined in different ways to represent even complex and highly specific setups.

Combining the modular metadata profiles can be accomplished via composite metadata profiles that use the modular metadata profiles as node constraints via sh:node. Same as described for inheritance in Section 3.1, the most straightforward approach for applying these composite metadata profiles to data-graphs would be to use sh:targetClass. Unfortunately, this approach, again, is not feasible, as we cannot assume that suitable target classes exist for each of our composite metadata profiles.

Proposed solution

We propose to model the relationship between composite and component on the metadata profile level without relying on classes. Specifically, the component resource related to by the composite resource is not constrained to be a member of any specific component class, but to conform to a component profile. In that way, the component profile does not need to target any class and the component resource does not need to correspond to a class at all.

```
ex:SensorProfile
    a sh:NodeShape ;
    sh:targetClass sosa:Sensor ;
    sh:property [
        sh:path schema:serialNumber ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [ # common requirement
        sh:path sosa:observes ;
        sh:node ex:PropertyProfile ;
    ] ;
    owl:imports ex:PropertyProfile .

ex:PropertyProfile
    a sh:NodeShape ;
    sh:property [ # common requirement
        sh:path qudt:hasQuantityKind ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] .

ex:TemperatureSensorProfile
    a sh:NodeShape ;
    sh:node ex:SensorProfile ;
    owl:imports ex:SensorProfile ;
    sh:property [ # more specific requirement
        sh:path sosa:observes ;
        sh:node ex:TemperatureProfile ;
    ] ;
    owl:imports ex:TemperatureProfile .

ex:TemperatureProfile
    a sh:NodeShape ;
    sh:property [ # more specific requirement
        sh:path qudt:hasQuantityKind ;
        sh:hasValue quantitykind:Temperature ;
```

```
            sh:minCount 1 ;
            sh:maxCount 1 ;
        ] .

ex:DatasetProfile
        a sh:NodeShape ;
        sh:targetClass schema:Dataset ;
        sh:property [
            sh:path schema:variableMeasured ;
            sh:node ex:TemperatureProfile ;
            sh:minCount 1 ;
        ] .
```

Listing 4: Metadata profile illustrating modularity. Validation example using direct targeting available at https://doi.org/10.48328/tudatalib-1166, https://s.zazuko.com/2JgBTZy.

The example shown in listing 4 illustrates a combination of hierarchical and modular modeling, in which a highly modular metadata profile for a single temperature value (`ex:TemperatureProfile`) is defined as a child of a general parent profile representing properties. The modular metadata profile is then (re)used by two composite profiles (`ex:TemperatureSensorProfile` and `ex:DatasetProfile`). An example where a single composite metadata profile imports multiple component profiles is shown in listing 6.

3.3 Combining terms from different terminology sources

When creating metadata, it is desirable to use the most fitting terms defined within existing established terminologies. This often means combining terms from different sources, especially when working in a domain like engineering that is characterized by a lack of subject-specific ontologies.

This poses a challenge when working on the ontology level, since there are no relations between terms unless explicitly introduced via attributes like `owl:equivalentClass`, `owl:equivalentProperty`, `rdfs:subClassOf` or `rdfs:subPropertyOf`. On the level of metadata profiles, this is also challenging, since restrictions or target classes that specify a class from one ontology are not satisfied by instances from different classes, unless some sort of equivalence relation has been stated. A `sosa:Sensor` would, e.g., not count as `m4i:Tool`, even though one would naturally assume that a sensor is a tool.

Proposed solution

Within our hierarchical and modular approach, however, we can combine terms from different ontologies using the modeling techniques described above.

Using the inheritance mechanism described in Section 3.1, i.e., importing a profile which targets a term in one vocabulary into another profile which targets another term from another vocabulary, causes no conflicts, as target classes are not inherited to the child class, leaving it free to specify a narrower target class (assuming such class exists) than its parent, regardless of whether that class has an explicitly stated relation to the parent's target class. Listing 5 illustrates this approach. A narrower metadata profile for sensors targets the `sosa:Sensor` class, whereas its parent profile targets the more general `m4i:Tool` class. The relation is realized purely on the metadata profile level and does not require relations defined on the ontology level.

Similarly, restrictions can be set to accept a list of alternative terms via `sh:or`, which allows including terms from different source ontologies without introducing conflicts.

Such use of ontology classes within metadata profiles contains information that could be used to deduce semantic relationships between the classes, e.g., that a child-profile's target class needs to be equivalent to or narrower than its parent-profile's target class, or that classes combined via `sh:or` are equivalent, have a common ancestor or have at least common meaning because they are interchangeable in the application scenario modeled. Within the article's focus on the metadata profile level, we have, however, not explored this option further.

```
ex:SensorProfile
    a sh:NodeShape ;
    sh:targetClass sosa:Sensor ;
    sh:node ex:ToolProfile ;
    owl:imports ex:ToolProfile ;
    sh:property [
        sh:path schema:serialNumber ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [ # common requirement
        sh:path sosa:observes ;
        sh:node ex:PropertyProfile ;
    ] ;
    owl:imports ex:PropertyProfile .

ex:ToolProfile
    a sh:NodeShape ;
    sh:targetClass m4i:Tool .
```

Listing 5: Metadata profile illustrating simultaneous use and alignment of topically related classes from different unaligned ontology sources.

3.4 Achieving specificity despite lack of suitable terms

The more specific something is, the more specific terms must be used to describe it in order to distinguish it from other things. However, sufficiently specific terms are rarely available, and even when they are, they limit the reusability of metadata profiles that rely on them, since specificity and reusability are conflicting goals that need to be carefully balanced to allow interoperability of the profiles while still meeting the specific needs of the particular research.

On the other hand, metadata profiles attempting to achieve specificity while using only general terms tend to become convoluted or abstract, which is also not desirable.

Proposed solution

Our approach is to achieve specificity while making do with existing, relatively general but therefore widely applicable terms that enable a high level of interoperability.

Specificity is accomplished via composition of modular metadata profiles. Instead of defining numerous metadata profiles representing very specific properties of an entity, existing more general properties are used, and the data nodes they point to are restricted to conform to modular metadata profiles, that are specific, but nevertheless highly reusable due to their modularity (c.f. Section 3.2). In this approach, it is even possible to include the same property multiple times within the same metadata profile, using a different metadata profile as restriction each time.

Listing 6 illustrates a metadata profile for a highly specific measurement setup that includes restrictions that both temperature and pressure measurements must be specified in the data. The profile does not rely on multiple distinct properties with node constraints for each of the measurement types to be included, but rather only uses the existing property `sosa:hasMember` for all measurement kinds, restricting the data nodes it refers to via `sh:qualifiedValueShape` to different metadata profiles representing the measurement types (`ex:TemperatureObservationProfile` and `ex:PressureObservationProfile`).[1]

```
ex:SensorProfile
    a sh:NodeShape ;
    sh:targetClass sosa:Sensor ;
        sh:property [
        sh:path schema:serialNumber ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    sh:property [ # common requirement
        sh:path sosa:observes ;
        sh:node ex:PropertyProfile ;
    ] ;
    owl:imports ex:PropertyProfile .

ex:PropertyProfile
    a sh:NodeShape ;
    sh:property [ # common requirement
        sh:path qudt:hasQuantityKind ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] .

ex:TemperatureSensorProfile
```

1. Note that one cannot simply use multiple properties with the same `sh:path` using normal node constraints, as restrictions via `sh:node` always need to be satisfied, even if defined within separate property constraints. The latter would lead to contradictions if more than one set of node constraints are defined for distinct properties with the same `sh:path`.

```
    a sh:NodeShape ;
    sh:node ex:SensorProfile ;
    owl:imports ex:SensorProfile ;
    sh:property [ # more specific requirement
        sh:path sosa:observes ;
        sh:node ex:TemperatureProfile ;
    ] ;
    owl:imports ex:TemperatureProfile .

ex:TemperatureProfile
    a sh:NodeShape ;
    sh:property [ # more specific requirement
        sh:path qudt:hasQuantityKind ;
        sh:hasValue quantitykind:Temperature ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] .

ex:PressureSensorProfile
    a sh:NodeShape ;
    sh:node ex:SensorProfile ;
    owl:imports ex:SensorProfile ;
    sh:property [ # more specific requirement
        sh:path sosa:observes ;
        sh:node ex:PressureProfile ;
    ] ;
    owl:imports ex:PressureProfile .

ex:PressureProfile
    a sh:NodeShape ;
    sh:property [ # more specific requirement
        sh:path qudt:hasQuantityKind ;
        sh:hasValue quantitykind:Pressure ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] .

ex:TemperatureObservationProfile
    a sh:NodeShape ;
    sh:property [
        sh:path sosa:madeBySensor ;
        sh:node ex:TemperatureSensorProfile ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
```

```
    ] ;
    owl:imports ex:TemperatureSensorProfile .

ex:PressureObservationProfile
    a sh:NodeShape ;
    sh:property [
        sh:path sosa:madeBySensor ;
        sh:node ex:PressureSensorProfile ;
        sh:minCount 1 ;
        sh:maxCount 1 ;
    ] ;
    owl:imports ex:PressureSensorProfile .

ex:MyMeasurementProfile
    a sh:NodeShape ;
    sh:targetClass sosa:ObservationCollection ;
    sh:property [
        sh:path sosa:hasMember ;
        sh:qualifiedValueShape ex:TemperatureObservationProfile ;
        sh:qualifiedMinCount 1 ;
    ] ;
    owl:imports ex:TemperatureObservationProfile ;
    sh:property [
        sh:path sosa:hasMember ;
        sh:qualifiedValueShape ex:PressureObservationProfile ;
        sh:qualifiedMinCount 1 ;
    ] ;
    owl:imports ex:PressureObservationProfile .
```

Listing 6: Metadata profile illustrating multiple specific occurrences of the same property. Validation example available at https://doi.org/10.48328/tudatalib-1167, https://s.zazuko.com/R4GZdm.

3.5 Direct targeting

Generally, we recommend using the techniques described above for targeting, i.e., stating a suitable unique target class in a high level metadata profile to start the application of metadata profiles to the data graph, and using relations introduced between metadata profiles to make sure that each profile targets the intended nodes in the data graph. However, in rare cases where no unique class is available to provide a suitable starting point for this kind of targeting chain, it is also possible to use targeting based on SPARQL rules to explicitly mark nodes in the data graph as target of a specified metadata profile. This has the disadvantage that information needs to be included in the data graph that mainly serves for targeting and would otherwise not be considered as “native” part of the metadata, and that it relies on advanced SHACL features that are currently not necessarily supported by validators, but can be used as a last resort for scenarios in which no data-intrinsic pattern can be used for targeting.

In those cases, the metadata profile to be applied can be declared directly within the entities in the data graph, for example via `dcterms:conformsTo`, and subsequently targets those entities.[2] Listing 7 shows how the targeting rule proposed above can be implemented, as well as an adjusted implementation of the `ex:MyMeasurementProfile`. Listing 8 shows a minimal example of a `sosa:ObservationCollection` present in a data graph that declares conformity to `ex:MyMeasurementProfile` and therefore is considered its target via the profile's `sh:target` condition.

```
ex:ConformsToShapeTarget
    a sh:SPARQLTargetType ;
    rdfs:subClassOf sh:Target ;
    sh:labelTemplate "All subjects that conform to {$conformsTo}" ;
    sh:parameter [
        sh:path dcterms:conformsTo ;
        sh:description "The shape that the focus nodes claim to conform to." ;
        sh:class sh:NodeShape ;
        sh:nodeKind sh:IRI ;
    ] ;
    sh:select """
        SELECT ?this
        WHERE {
            ?this <http://purl.org/dc/terms/conformsTo> $conformsTo .
        }
        """ .
ex:MyMeasurementProfile
    a sh:NodeShape ;
    sh:target [
        a ex:ConformsToShapeTarget ;
        dcterms:conformsTo ex:MyMeasurementProfile ;
    ] ;
    sh:property [
        sh:path sosa:hasMember ;
        sh:qualifiedValueShape ex:TemperatureObservationProfile ;
        sh:qualifiedMinCount 1 ;
    ] ;
    owl:imports ex:TemperatureObservationProfile ;
    sh:property [
        sh:path sosa:hasMember ;
        sh:qualifiedValueShape ex:PressureObservationProfile ;
```

2. Note that there are other targeting mechanisms of the SHACL language like `sh:targetNode`, `sh:targetSubjectsOf`, or `sh:targetObjectsOf`, that are not discussed in detail in this article. `sh:targetNode` is not recommended since it requires declaring individual nodes instead of relying on some pattern for matching, whereas `sh:targetSubjectsOf` and `sh:targetObjectsOf` suffer the same problem as using `sh:targetClass` in that they would require ontologies that provide properties that are specific enough to be matched to metadata profiles, which is usually not the case.

```
        sh:qualifiedMinCount 1 ;
    ] ;
    owl:imports ex:PressureObservationProfile .
```

Listing 7: Metadata profile illustrating rule-based targeting: The profile is applied to all entities that adhere to the pattern specified by a SPARQL based custom target. Validation requires SHACL processors to support advanced SHACL features. Validation example available at https://doi.org/10.48328/tudatalib-1168.

```
ex:SomeMeasurement
    a sosa:ObservationCollection ;
    dcterms:conformsTo ex:MyMeasurementProfile ;
    sosa:hasMember ex:SomeTemperatureObservation ;
    sosa:hasMember ex:SomePressureObservation .
```

Listing 8: Minimal example of a data graph containing a `sosa:ObservationCollection` declaring conformance to the `ex:MyMeasurementProfile` via `dcterms:conformsTo`, which can be used for rule-based targeting as illustrated by listing 7. A validation example is available at https://doi.org/10.48328/tudatalib-1168.

4 Summary and outlook

In conclusion, we have demonstrated a way to create subject specific RDF-compliant metadata profiles (in the sense of SHACL shapes) that allow precise and flexible documentation of research processes and data. We have implemented a hierarchical inheritance concept for the profiles, which we combine with a strategy that uses the composition of relatively simple modular profiles to model complex setups. As a result, the individual profiles are highly reusable and can be applied in different contexts, which in turn increases the interoperability of the resulting data. We also demonstrated that it is possible to achieve specificity even when only general terms are available within existing terminologies. We do this by relying on existing relatively unspecific properties that we make more specific by restricting the nodes to which they refer to conform to metadata profiles that convey the desired level of specificity.

While we have demonstrated our approach using examples from the domain of mechanical engineering, our modeling technique is subject-independent and also applicable to other disciplines. In fact, the approach resonates well with domain-agnostic guidelines for metadata profiles brought forth by W3C's Dataset Exchange Working Group [14].

To facilitate the modeling process and make it available to users with only very little knowledge of RDF, we are currently developing a web service providing a graphical user interface for creating metadata profiles within the AIMS project (cf. [12]). The web service supports searching for suitable terms from existing terminologies and assembling them into profiles via drag-and-drop. Profiles created on the service will be shared via a publicly available search function so that other scientists can discover exiting profiles and reuse or extend them for their research. In addition, the service supports curation of the profiles by existing scientific communities.

The service will fully support the modeling techniques described above. An example of a graphical representation of interacting modular metadata profiles as rendered by the service is shown in Fig. 3. An instance of the web service will soon be available as a metadata profile

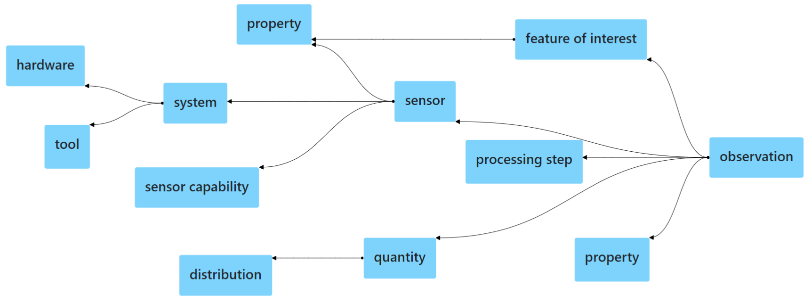

Figure 3: Example representation of a more complex metadata profile

service within the German National Research Data Infrastructure for Engineering Sciences (NFDI4Ing). Related news and updates can be found via the NFDI4Ing homepage [15].

5 Acknowledgements

Funded by the Deutsche Forschungsgemeinschaft (DFG, German Research Foundation) under Project-ID 432233186 – AIMS. The authors would like to thank the principal investigators of this project, Peter F. Pelz, and Thomas Stäcker of Technische Universität Darmstadt, Robert H. Schmitt and Matthias S. Müller of RWTH Aachen University, for supervising and supporting this research. In addition, the authors would like to thank the Federal Government and the Heads of Government of the Länder, as well as the Joint Science Conference (GWK), for their funding and support within the framework of the NFDI4Ing consortium; funded by the German Research Foundation (DFG) - project number 442146713.

6 Roles and contributions

Nils Preuß: Conceptualization, Methodology, Writing – original draft

Matthias Bodenbenner: Conceptualization, Methodology, Writing – original draft

Benedikt Heinrichs: Conceptualization, Methodology, Writing – original draft, Software

Jürgen Windeck: Conceptualization, Writing – original draft

Mario Moser: Conceptualization, Writing – original draft

Marc Fuhrmans: Conceptualization, Writing – original draft, Project administration

References

[1] D. Wood, M. Lanthaler, and R. Cyganiak, "RDF 1.1 Concepts and Abstract Syntax," W3C, W3C Recommendation, Feb. 2014, https://www.w3.org/TR/2014/REC-rdf11-concepts-20140225/.

[2] M. D. Wilkinson, M. Dumontier, I. J. J. Aalbersberg, *et al.*, "The FAIR Guiding Principles for scientific data management and stewardship," *Scientific data*, vol. 3, p. 160 018, 2016. DOI: 10.1038/sdata.2016.18.

[3] C. Bizer, T. Heath, and T. Berners-Lee, "Linked data: The story so far," in *Semantic services, interoperability and web applications: emerging concepts*, IGI global, 2011, pp. 205–227.

[4] Z. Chen, D. Wu, J. Lu, and Y. Chen, "Metadata-based information resource integration for research management," *Procedia Computer Science*, vol. 17, pp. 54–61, 2013, First International Conference on Information Technology and Quantitative Management, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.05.009.

[5] OPC Foundation. "OPC UA (opc unified architecture)." (2009), [Online]. Available: https://opcfoundation.org/about/opc-technologies/opc-ua/ (visited on 05/31/2023).

[6] Y. Filke, L. Gomez, L. Hanke, *et al.*, *DEXPI - P&ID Specification*, 2021. [Online]. Available: https://dexpi.org/wp-content/uploads/2020/09/DEXPI-PID-Specification-1.3.pdf.

[7] R. R. Panko and S. Aurigemma, "Revising the panko–halverson taxonomy of spreadsheet errors," *Decision Support Systems*, vol. 49, no. 2, pp. 235–244, 2010, ISSN: 0167-9236. DOI: https://doi.org/10.1016/j.dss.2010.02.009.

[8] D. Smith, "Appendix 6 - human error probabilities a2," *Reliability, Maintainability and Risk (Eighth Edition)*, vol. 8, pp. 395–397, 2011, ISSN: 1877-0509. DOI: https://doi.org/10.1016/j.procs.2013.05.009.

[9] J. Gray, D. T. Liu, M. Nieto-Santisteban, A. Szalay, D. J. DeWitt, and G. Heber, "Scientific data management in the coming decade," *Acm Sigmod Record*, vol. 34, no. 4, pp. 34–41, 2005.

[10] S. Arndt, B. Farnbacher, M. Fuhrmans, *et al.*, *Metadata4Ing: An ontology for describing the generation of research data within a scientific activity.* Version 1.1.0, Feb. 2022. DOI: 10.5281/zenodo.7706017.

[11] D. Kontokostas and H. Knublauch, "Shapes Constraint Language (SHACL)," W3C, W3C Recommendation, Jul. 2017, https://www.w3.org/TR/2017/REC-shacl-20170720/. (visited on 05/31/2023).

[12] M. Grönewald, P. Mund, M. S. Bodenbenner, *et al.*, "Mit AIMS zu einem Metadatenmanagement 4.0 : FAIRe Forschungsdaten benötigen interoperable Metadaten," in *E-Science-Tage 2021 : share your research data*, V. Heuveline and N. Bisheh, Eds., Heidelberg: Universitätsbibliothek Heidelberg, 2022, pp. 91–104. DOI: 10.11588/HEIBOOKS.979.C13721.

[13] D. Brickley and R. Guha, "RDF schema 1.1," W3C, W3C Recommendation, Feb. 2014, https://www.w3.org/TR/2014/REC-rdf-schema-20140225/. (visited on 05/31/2023).

[14] W. D. E. W. Group. "Profile Guidance." (2023), [Online]. Available: https://w3c.github.io/dxwg/profiles/ (visited on 05/31/2023).

[15] NFDI4Ing. "National Research Data Infrastructure for Engineering Sciences." (2023), [Online]. Available: https://nfdi4ing.de (visited on 05/31/2023).