Revisiting Ontologies of Units of Measure for Harmonising Quantity Values – A Use Case

Francisco Martín-Recuerda[1(✉)], Dirk Walther[1], Siegfried Eisinger[1], Graham Moore[2], Petter Andersen[3], Per-Olav Opdahl[3], and Lillian Hella[3]

[1] DNV GL, Oslo, Norway
{francisco.martin-recuerda,dirk.walther,siegfried.eisinger}@dnvgl.com
[2] Sesam.io, Oslo, Norway
graham.moore@sesam.io
[3] ix3 – an Aker Solutions Company, Fornebu, Norway
{petter.andersen,per-olav.opdahl,lillian.hella}@akersolutions.com

Abstract. Processing quantity values in industry applications is often arduous and costly due to different systems of units and countless naming and formatting conventions. As semantic technology promises to alleviate the situation, we consider using existing ontologies of units of measure to improve data processing capabilities of a cloud-based semantic platform for operating digital twins of real industry assets. We analyse two well-known ontologies: OM, the Ontology of units of Measure, and QUDT, the Quantities, Units, Dimensions and Types ontology. These ontologies are excellent resources, but do not meet all our requirements. We discuss suitable modelling choices for representing quantities, dimensions and units of measure and we outline the process we followed to adapt relevant definitions from OM and QUDT into a new ontology of units of measure that better meets our needs. Compared with the alternative of manually creating the ontology from scratch, the development and maintenance costs and duration were reduced significantly. We believe this approach will achieve similar benefits in other ontology engineering efforts.

Keywords: Digital twin · Ontology engineering · Units of measure

1 Introduction

Aker Solutions,[1] DNV GL[2] and Sesam[3] are three companies based in Norway that are collaborating on the development of services and IT infrastructure for the design, implementation, operation and quality assurance of digital twins

[1] https://www.akersolutions.com/.
[2] https://www.dnvgl.com/.
[3] https://sesam.io/.

This work has been supported by Aker Solutions, DNV GL, Sesam and SIRIUS labs. A special thanks to Kaia Means (DNV GL) for reviewing this document.

J. Z. Pan et al. (Eds.): ISWC 2020, LNCS 12507, pp. 551–567, 2020.
https://doi.org/10.1007/978-3-030-62466-8_34

(DTs).[4] To this end, Aker Solutions has founded a dedicated software company, ix3,[5] responsible for the implementation, management and commercialisation of the *ix3's digital twin platform Integral* (*Integral platform* for short), a cloud-based digital infrastructure for operating DTs. This platform facilitates the deployment, integration and orchestration of multiple types of digital services including data analytics and (multi-physics and process) simulation, which are essential when operating DTs. In particular, the Integral platform collects, harmonises, contextualises, analyses, visualises and distributes large amounts of heterogeneous data. This includes not only operational data from sensors and control systems but also engineering and enterprise data from production systems. In this paper, we define *data harmonisation* as the process of providing a common representation of quantities and units (including the use of unique identifiers), and *data contextualisation* as the process of enriching data with additional information that facilitates its understanding and processing.

Collecting, harmonising, contextualising and distributing data are tasks delegated to *Sesam Datahub*, enriched with an ontology library, developed by ix3 in cooperation with DNV GL, the *Information Model* (IM). The IM includes more than 100 OWL 2 (Direct Semantics) ontologies organised in a strict dependency hierarchy using the OWL 2 import mechanism. On top of the hierarchy are domain-independent ontologies, such as the ISO 15926-14 ontology,[6] SKOS[7] and PAV.[8] The next level of ontologies describes generic concepts in the engineering domain and mappings between system codes from Aker Solutions and from its customers. The lower level of ontologies represents oil and gas assets and related technical and commercial documentation. The IM provides Sesam with a controlled language with unique identifiers (IRIs), definitions for each term and mappings that explicitly state how terms defined by external sources are related to terms defined by the IM. Sesam uses this information to contextualise and harmonise the data propagated through *pipes*, which are internal processes defined using a declarative language and supported by specific built-in connectors and microservices for consuming, transforming and exporting data from/to external data-sources.

Data consumed and produced by DTs are often in the form of quantity values, which are products of numbers and units. Processing quantity values is still arduous and costly, mainly due to different systems of units and countless naming and formatting conventions. For instance, the following data items represent the same quantity value: '392.75 FAHR', '392,75 F', '39,275,231e−5 oF', '200,42 C', '200.416.667E−6 DegC', '473,57 K' and '4.7357e2 KLV'. A unit of measure, such as Fahrenheit, has several different abbreviations (i.e. FAHR, F and oF), punctuation may vary, or other units may be used, such as Celsius (i.e. C and DegC) or Kelvin (i.e. K and KLV). To support Sesam when harmonising these examples of quantity values, the IM provides a preferred label and a unique

[4] https://en.wikipedia.org/wiki/Digital_twin.
[5] https://www.ix3.com/.
[6] https://www.iso.org/standard/75949.html.
[7] https://www.w3.org/TR/skos-reference/.
[8] https://github.com/pav-ontology/pav/.

identifier for units of measure such as Fahrenheit, Celsius and Kelvin. The IM also includes the information to convert quantity values into different units of measure (i.e. from Fahrenheit to Kelvin). In addition, the IM maintains mappings between its terms and external terms using SKOS semantic properties (i.e. the terms FAHR, F and oF represents the unit Fahrenheit in the IM). Additional context of a quantity value, such as a related quantity kind (i.e. thermodynamic temperature), can be also obtained from the IM.

Main Results and Organisation. In this paper, we describe how we extended the IM to better support Sesam when harmonising and contextualising quantity values in the Integral platform for digital twins. To minimise development costs, we tried to reuse existing ontologies of units of measure. In Sect. 2, we provide a summary of the analysis of two well-known such ontologies, the Ontology of units of Measure (OM)[9] and the Quantities, Units, Dimensions and Types ontology (QUDT).[10] Although both ontologies are excellent resources, we concluded that they must be revised to meet all of our requirements. Therefore, we built a new ontology of units of measure based on QUDT and OM that is better adapted to our needs. In Sect. 3, we discuss suitable modelling choices and ontology design patterns [2] that we considered for representing quantities, dimensions and units. We revisited the implementation and build processes of the IM to accommodate the new ontology of units of measure (cf. Sect. 4). In particular, we tried to reduce manual effort, facilitate collaborative development and ensure uniform modelling by implementing and applying ontology templates [6] and specific SPARQL queries. We conclude the paper in Sect. 5.

Notation and Terminology. In this paper, we use notation and terminology related to units of measure, ISO standards, W3C recommendations and ontologies. Terms such as dimensional analysis, units of measure, quantities (or quantity kinds), dimensions and systems of units are well introduced by Wikipedia[11] and the standard ISO/IEC 80000,[12] which describes the International System of Quantities (ISQ)[13] and also includes all units defined by the International System of Units (SI).[14] The various W3C recommendations for OWL 2, including syntaxes and semantics are accessible from the W3C website.[15] This is relevant when we refer to IRIs, named and anonymous individuals, classes, object and data properties, class expressions (or restrictions), assertions and punning. SKOS is a W3C recommendation that includes an OWL ontology. The notions of SKOS concepts, SKOS concept schemes and SKOS semantic and mapping properties are relevant for this paper. In addition to the ontologies OM, QUDT, SKOS, ISO 15926-14 and PAV we also refer to the ontologies DTYPE v1.2,[16]

[9] https://github.com/HajoRijgersberg/OM.
[10] https://github.com/qudt/qudt-public-repo/releases.
[11] https://en.wikipedia.org/wiki/Dimensional_analysis.
[12] https://www.iso.org/committee/46202/x/catalogue/.
[13] https://en.wikipedia.org/wiki/International_System_of_Quantities.
[14] https://en.wikipedia.org/wiki/International_System_of_Units.
[15] https://www.w3.org/TR/owl2-overview/.
[16] http://www.linkedmodel.org/doc/2015/SCHEMA_dtype-v1.2.

and VAEM v2.0.[17] When terms defined by these ontologies are explicitly mentioned, we use the following prefixes: om, qudt, skos, dtype, iso, pav, dtype and vaem, respectively. QUDT includes additional prefixes when referring to quantity kinds, dimensions and units such as quantitykind, qkdv and unit. Due to space limitations, we do not include the namespaces that correspond to these prefixes.

2 Assessing Ontologies of Units of Measure

In this section, we discuss the suitability of the ontologies OM and QUDT, based on a selection of the requirements that we considered in our use case, and we refrain from reexamining existing work on assessing ontologies of units of measure, such as Keil et al. [3]. OM and QUDT are outstanding candidates among the ontologies of units of measure that we considered. Both ontologies are prominent and explicitly referenced, for instance by Wikidata[18] and W3C SOSA/SSN.[19] In addition, these ontologies fulfil many of the requirements discussed in this section.

The selection of requirements includes a mix of functional and non-functional requirements that are also relevant for other use cases aiming to contextualise and harmonise quantity values using ontologies of units of measure. In fact, we noticed that many of the requirements of the Integral platform are similar to other projects where we applied ontologies to improve interoperability of applications. The key words *MUST* and *SHOULD* are to be interpreted as described in RFC 2119.[20] Table 1 summarises the assessment of the requirements considered. We use the following codes: +, - and *, which mean the requirement is satisfied, not satisfied, and nearly satisfied, respectively.

Table 1. Requirements assessment for the ontologies OM and QUDT

Requirement	Key words	OM	QUDT
R01	Public licence	+	+
R02	Active maintenance	+	+
R03	Coverage	+	+
R04	OWL 2 Direct Semantics	+	-
R05	(Web) Protégé & OWLAPI	+	-
R06	HermiT reasoner	-	-
R07	Ontology design patterns	*	*
R08	Dimensional analysis & SPARQL	-	+
R09	Compatibility with ISO 15926-14	+	*
R10	Compatibility with SKOS	-	-
R11	Modularity	-	+

[17] http://www.linkedmodel.org/doc/2015/SCHEMA_vaem-v2.0.
[18] https://www.wikidata.org/wiki/Wikidata:Main_Page.
[19] https://www.w3.org/TR/vocab-ssn/.
[20] https://tools.ietf.org/html/rfc2119.

It is obvious that QUDT and OM meet the requirements **R01**, **R02** and **R03**. Both ontologies are distributed under public licence CC BY 4.0[21], new releases of these ontologies are made available every few months, and they define a broad range of units, quantities and dimensions. For the rest of the requirements, we provide a more detailed discussion in the following paragraphs.

R04: *A candidate ontology MUST be specified using OWL 2 language under direct, model-theoretic semantics.* This requirement is a consequence of implementing the Information Model in OWL 2 and applying OWL 2 (DL) reasoners such as HermiT[22] to verify the consistency of the ontologies and to compute their class and property hierarchies (which would be impractical when using OWL 2 RDF-based Semantics[23]) during the build process (cf. Sect. 4). The requirement is fulfilled by OM but not by QUDT. We identified several issues in the QUDT ontologies, as well as imported ontologies such as DTYPE v1.2. These issues are mostly related to the fact that DTYPE and QUDT follow OWL 2 RDF-based Semantics instead of OWL 2 Direct Semantics. For instance, under OWL 2 Direct Semantics, it is not possible to implement cardinality restrictions on transitive properties.[24] The property qudt:isScalingOf is defined as being transitive and it is used in the definition of the class qudt:ScaledUnit together with a cardinality restriction. Additional typing restrictions defined under OWL 2 Direct Semantics[25] are not followed by QUDT and DTYPE. For instance, a property cannot be an object and a data property in the same set of axioms. We observed that DTYPE and QUDT ontologies define several properties of type rdf:Property which is a superclass of owl:ObjectProperty and owl:DataProperty. In addition to the issues regarding semantics, we also detected several syntactic errors. For instance, the definition of the class qudt:PhysicalConstant includes a cardinality restriction over the annotation property qudt:latexDefinition, which is not valid OWL 2 syntax.

R05: *A candidate ontology MUST be compliant with "de facto" reference tools for OWL 2 such as (Web-)Protégé and OWLAPI.* The requirement is needed to support the team responsible of developing and maintaining the Information Model. They use (Web-)Protégé[26] to manually inspect ontologies and the OWLAPI[27] to support some of the services responsible for building and validating new releases of the Information Model (cf. Sect. 4). The requirement is fulfilled by OM but not by QUDT. In the case of QUDT, it is not a surprise given the problems we reported with requirement R04.

R06: *A candidate ontology MUST ensure acceptable reasoning performance using the state-of-the-art reasoner HermiT.* The requirement is relevant for releasing, (manually) inspecting and applying our ontologies (cf. requirement

[21] https://creativecommons.org/licences/by/4.0/legalcode.
[22] http://www.hermit-reasoner.com/.
[23] https://www.w3.org/TR/owl2-primer/#OWL_2_DL_and_OWL_2_Full.
[24] https://www.w3.org/TR/owl-ref/#TransitiveProperty-def.
[25] https://www.w3.org/TR/owl2-syntax/#Typing_Constraints_of_OWL_2_DL.
[26] https://protege.stanford.edu/.
[27] https://github.com/owlcs/owlapi.

R05 and Sect. 4). Acceptable reasoning performance should be achieved on a commodity PC. Reasoning tasks are not limited to consistency checking and classification, they also include materialisation of inferred class and property assertions. Since QUDT does not fulfil requirements R04 and R05, it does not meet this requirement. Neither does OM. Classifying OM takes more than 25 min for HermiT (v1.4.3.456) running on a Dell Latitude laptop with i5-6300U CPU (2.50 Ghz), 16 GB RAM and Windows 10 (64 bit). Materialisation of class and property assertions is even more time consuming. After more than five hours, HermiT could not complete these reasoning tasks. The problem is related to the definition of classes for units. Instead of explicitly asserting units (represented by named individuals) into classes of units, this is done by defining complex class restrictions involving nominals and disjunctions. In particular, we noticed that it is very difficult to reason with subclasses of compound and prefixed units due to the definition of specific equivalent-classes restrictions. After simplifying the definitions of the unit classes in OM, Hermit completed classification and materialisation of class and property assertions in less than one minute.

R07: *A candidate ontology MUST provide well-defined ontology design patterns for quantity values, quantity kinds and units.* Well-defined ontology design patterns [2] are instrumental when implementing and maintaining large industrial ontologies such as the Information Model. Definition of classes for quantity values, quantity kinds, units and dimensions should include class restrictions that clearly state the expected class and property assertions for individuals of these classes. OM and QUDT nearly achieve this requirement. In the case of OM, it includes well-defined ontology design patterns for quantity values, quantity kinds and dimension vectors. However, we missed the expected class restrictions in the classes om:Measure, om:Quantity, and om:Dimensions. Class restrictions can be found in the definition of the subclasses of om:Quantity and om:Dimensions. The subclasses of om:Quantity also provide the necessary class restrictions for individuals of the class om:Measure. The definition of the class om:Unit does not include any class restrictions either. This is due to the different nature of the subclasses of units defined by the designers of OM.

QUDT includes well-defined ontology design patterns for quantity values, units, quantity kinds and dimensions. These design patterns are clearly specified by the respective classes qudt:QuantityValue, qudt:Unit as well as the classes qudt:QuantityKind and qudt:QuantityKindDimensionVector. QUDT also includes the class qudt:Quantity that represents a relation between a phenomenon (object or event), a quantity kind and a collection of quantity values of the same kind (similar to individuals of type om:Quantity). The specification is not clear about which property relates a phenomenon with a quantity, but we suspect it is qudt:hasQuantity.

R08: *A candidate ontology MUST be optimised for dimensional analysis using SPARQL.* The requirement is essential for practical applications based on semantic technologies. This includes conversion of units, verification of dimensional homogeneity and discovery of base quantities (and units) from derived quantities (and units). QUDT fulfils this requirement, OM does not (not entirely). OM and QUDT provide an excellent support for the verification of dimensional

homogeneity and the identification of base quantities (and units) from derived quantities (and units). This is possible thanks to the notion of *dimension vector*, representing a derived dimension, defined by ISQ as the product of powers of the seven base dimensions. Both ontologies define each dimension vector as a named individual asserted to the exponents of each base dimension. OM and QUDT define seven data properties, each related to one base dimension. By adding the exponents of each base dimension, it is possible to verify if two derived quantity kinds or two complex expressions involving powers of products of derived quantity kinds are commensurable. This can be done with a relatively simple ASK SPARQL query that will return true if the base dimensions of two expressions are the same.

Unit conversion using SPARQL is relatively simple in QUDT but not in OM. Computing the conversion of two commensurable units defined by QUDT is done by using the values of two data properties: qudt:conversionMultiplier and qudt:conversionOffset. The values of these properties determine how the magnitude of a quantity value can be converted to a base (or reference derived) unit. For instance, the conversion multiplier and offset of the unit kilowatt hour (unit:KiloW-HR) are $3.6e^6$ and 0.0. These values are defined with respect to the reference derived unit Joule (unit:J), which is of the same kind, and has as conversion multiplier 1.0 and offset 0.0. Therefore, converting a magnitude defined in kilowatt hour into joules is as simple as multiplying by $3.6e^6$.

The case for unit conversion using SPARQL on OM is not as simple, for two main reasons. The first is that the conversion factor (or multiplier) and offset are not available for all derived units. Consequently, to obtain the necessary conversion factors and offsets for a given unit, we must traverse the RDF graph until we find a unit that has one or two of these properties. The second reason is that OM defines several types of units with different properties. This means that we must add several optional graph patterns to deal with all these cases in a SPARQL query. This can be easily verified by trying to convert again a magnitude defined in kilowatt hour (om:kilowattHour) into joules (om:joule). The unit om:kilowattHour does not include any conversion factor in its definition. Therefore, we need to find the information in the definition of its constituent units starting from om:kilowatt and om:hour.

R09: *A candidate ontology MUST be compatible with ISO 15926-14 upper ontology.* The requirement is the result of using ISO 15926-14 as our reference upper ontology, which is influenced by BFO. The ontologies OM and QUDT are both compatible with ISO 15926-14. The ontology design patterns defined by OM for representing quantity values, quantity kinds and units fit well to ISO 15926-14. Dimensions are not explicitly covered by the ISO 15926-14 ontology, but they can be represented as individuals of a subclass of iso:InformationObject. QUDT does not meet the requirement as well as OM, because it uses a different ontology design pattern for representing quantity kinds. Contrary to ISO 15926-14 and OM, QUDT models quantity kinds only as individuals and not as classes. These individuals do not represent a relation between a particular phenomenon (such as an object or event) and a collection of quantity values, as in OM and ISO 15926-14.

R10: *A candidate ontology SHOULD enable mapping definitions with related terms defined by external data sources using SKOS.* We often use SKOS to define mappings between locally defined terms and related terms defined by external sources. Neither OM, nor QUDT provide appropriate support for SKOS. However, it is possible to adapt these ontologies to fulfil the requirement. In fact, QUDT uses SKOS properties to define semantic relations between QUDT terms and to specify mappings with terms defined by Wikipedia and DBpedia. However, we observed that these properties are not always consistently applied and, in general, the use of SKOS must be revised to make QUDT fully compliant with the W3C recommendation. For instance, the quantity kind quantitykind:LuminousEnergy is related to the quantity kind quantitykind:RadiantEnergy, using the SKOS mapping property skos:closeMatch. None of these terms are defined as SKOS concepts and are not grouped in specific SKOS concept schemes, which is expected when using SKOS properties.

R11: *A candidate ontology SHOULD be developed as a coherent collection of modules.* The requirement is related to well-known best practices when developing large industrial ontologies such as the Information Model. QUDT meets this requirement, but OM does not. The latter is released as a single ontology, whereas QUDT is released as a collection of 52 ontologies. We appreciate the strict separation between the QUDT upper ontology, SCHEMA_QUDT-v2.1.ttl, and the remaining files. Units, quantity kinds, dimensions and physical constants are also defined in separate ontologies.

The result of our analysis of OM and QUDT shows that both ontologies are, in principle, suitable for the harmonisation and contextualisation of quantity values and the implementation of services for dimensional analysis. However, we also concluded that neither of these ontologies fulfil all of our requirements. This appears to be a common situation that private and public organisations may frequently face when attempting to incorporate external resources into their ontology libraries. This usually ends up in the decision to build a new ontology from scratch, where certain term definitions are manually copied from existent ontologies, edited and finally inserted into the new ontology. As reported by Skjæveland et al. [6], the manual creation and maintenance of ontologies is expensive and error-prone. The problem is exacerbated by the fact that the ontologies QUDT and OM are updated often. To reduce time and development costs of creating and maintaining a new ontology of units of measure based on QUDT and OM that better suits our needs, and at the same time, while preserving interoperability, we must consider carefully the design, implementation and build phases. In the design phase, we must identify, which modelling choices and ontology design patterns better suit our needs. This is the aim of the next section.

3 Modelling Choices and Design Patterns

Ontology design patterns (ODPs) [2] define relevant and best-practice conceptual building blocks for implementing high-quality ontologies. ODPs ensure consistent and uniform modelling of similar terms and facilitate understanding of large

ontologies, which is instrumental for reducing maintenance costs. As reported during the assessment of requirement R07, QUDT and OM provide well-defined modelling patterns (with few exceptions) for each of their fundamental concepts (i.e. quantity values, dimensions, quantity kinds and units of measure). Therefore, QUDT and OM modelling patterns represent a good starting point when defining the modelling patterns for the new ontology of units of measure. The assessment of requirement R08 indicates that the modelling pattern for representing dimensions (very similar in both ontologies) facilitates the verification of dimensional homogeneity using SPARQL. The same is not true when computing unit conversion using SPARQL, where the modelling pattern of QUDT for representing units of measure has a clear advantage. In addition, the definition of classes of units in OM, particularly subclasses of prefixed and compound units must be updated to avoid inefficient reasoning (cf. assessment of requirement R06).

Before discussing specific modelling patterns for representing quantity values (Sect. 3.2), dimensions (Sect. 3.3), quantity kinds (Sect. 3.4) and units of measure (Sect. 3.5), we will briefly introduce some general design decisions regarding modular structure and the use of upper ontologies (Sect. 3.1).

3.1 General Considerations

The Information Model follows a strict dependency hierarchy, where the ontologies ISO 15926-14 and SKOS are at the top. It is important to take this into consideration when defining the modular structure of the new ontology of units of measure and how it is integrated with the upper and domain independent ontologies. These decisions will contribute to a better fulfilment of the requirements R09, R10 and R11 discussed in the previous section.

Modular Structure. As with QUDT, we stored the definitions of quantity kinds, (types of) quantity values, dimensions and units of measure in different ontologies (with different namespaces). Moreover, we placed common term definitions (of classes and properties) in a dedicated *core* ontology, that is imported by other ontologies. These ontologies were released as an ontology library (UoM library, for short) organised following a strict dependency hierarchy, where ontologies can only refer to terms defined in ontologies located in upper levels. Respect a strict dependency hierarchy in the UoM library may be difficult because a term in one ontology, such as a specific unit, may refer to terms in different ontologies, such as a particular quantity kind or dimension. To avoid cross references between ontologies at the same level, we separate term declarations from term definitions and we store them in different files. Files containing the term declarations import the core ontology and they are, in turn, imported by the ontologies that include the definitions of these terms. In the remaining document, we refer to defined terms using the following prefixes: uomcore (common terms), uomqd (types of quantity values), uomqk (quantity kinds), uomqkdv (dimensions) and uomunit (units).

ISO 15926-14 and SKOS Ontologies. It is important that the ontologies of the UoM library are well aligned with the ISO 15926-14 and SKOS ontologies. The alignment with the ISO 15926-14 ontology is relatively easy to achieve, because QUDT and OM are already quite compatible. For instance, the ISO 15926-14 classes iso:PhysicalQuantity, iso:QuantityDatum and iso:UnitOfMeasure correspond well with the classes in QUDT and OM that define quantity kinds, (types of) quantity values (or measurements) and units of measure. The case of quantity kinds is somewhat more challenging as the design pattern adopted for the ISO 15926-14 ontology is similar to its counterpart in OM but not to the one in QUDT. The class for defining dimension vectors (defined similarly by QUDT and OM) is not supported by the ISO 15926-14 ontology, but it can be specified as a subclass of iso:InformationObject.

Just below the ISO 15926-14 ontology, the Information Model includes the ontology SKOS as part of the ontology (import) hierarchy. SKOS has been adopted by Aker Solutions, ix3, Sesam and DNV GL to facilitate interoperability with external resources. To define SKOS mappings between terms in the ontology library (local terms) and terms used by external resources (external terms), we apply SKOS semantic properties such as skos:Broader. This requires that all mapped terms must be defined as SKOS concepts and associated to one or more SKOS concept schemes, which implies that these terms are being declared as OWL individuals. When they are not, we apply *punning* to produce the necessary OWL individuals.

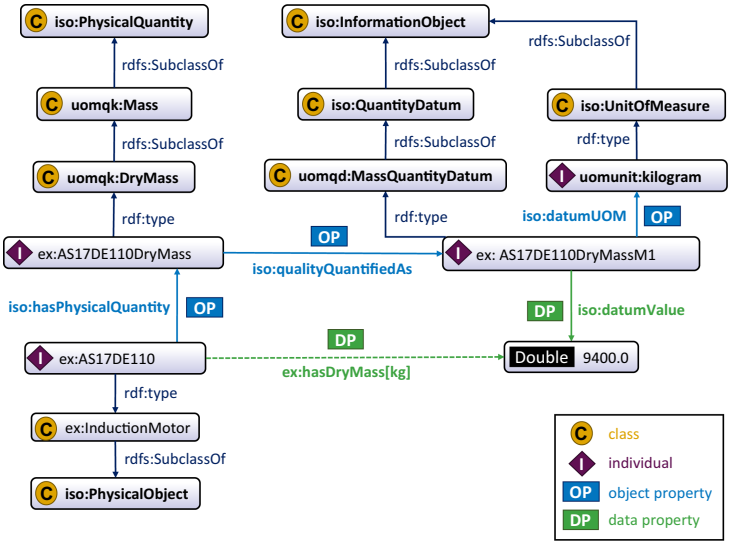

Fig. 1. Modelling example

The modelling example in Fig. 1 describes a simplified scenario illustrating some of the modelling choices we considered when building a new ontology of

units of measure. The individual ex:AS17DE110 represents an induction motor that has the physical quantity of type uomqk:DryMass associated with it (the prefix ex: refers to a namespace for examples). As an example of the modelling pattern for representing quantity kinds as classes (cf. Sect. 3.4), the relation is defined using the ISO 15926-14 object property iso:hasPhysicalQuantity and the individual ex:AS17DE110DryMass. A quantity value representing a measurement of the dry mass of the induction motor is represented by the individual ex:AS17DE110DryMassM1. Notice that we used named individuals instead of anonymous individuals for physical qualities and values to make the example easier to understand. We will discuss this example in the following subsections.

3.2 Quantity Values

A quantity value is represented as a relation that contains at least one numerical value (or magnitude) and one unit of measure. This is illustrated by the individual ex:AS17DE110DryMassM1 in Fig. 1, where the data property iso:datumValue states the numerical value and the object property iso:datumUOM indicates the unit of measure. In addition, we extended this relation to include a timestamp that indicates when the quantity value was generated and the uncertainty of the quantity value commonly represented by the (relatively) standard uncertainty (both properties are defined in the core ontology and are not included in Fig. 1 for the sake of simplicity). Therefore, a suitable approach for representing n-ary relations in OWL 2 was required. Among the different available modelling approaches, we followed Noy et al. [4] and defined quantity values as individuals. Notice that QUDT and OM represent quantity values using the same modelling approach.

There are two issues to note. First, the question of whether individuals representing quantity values should be defined as named or anonymous individuals. This is not clearly stated in ISO 15926-14, OM and QUDT, and the decision may depend on the application. In the case of the Integral platform, named individuals make sense, because the application Sesam is responsible for detecting changes in the external data sources and updating the data accordingly. Therefore, the numerical value does not determine the identity of a quantity value. However, tracking quantity values is not a simple task, and it requires substantial contextual information around each quantity value. This can be costly in terms of storage and computing resources. The second issue is determining the suitability of this modelling approach, i.e. modelling quantity values as individuals, when there are many values to be represented or when values stem from time series data. In the former case, there are simpler modelling approaches, where quantity values are represented using data property assertions (cf. Fig. 1, ex:hasDryMas[kg]) as done similarly in the SOSA/SSN ontology using the data property sosa:hasSimpleResult.[28]. We also considered the latter case, time series data, where different modelling patterns can be applied, but this is not discussed here.

[28] https://www.w3.org/TR/vocab-ssn/#SOSAhasSimpleResult.

3.3 Dimension Vectors

A dimension vector, representing the product of powers of the seven base dimensions, is defined using an individual and seven data property assertions, where each data property corresponds to one base dimension. This is similar to the way dimension vectors are represented in OM and QUDT. This representation facilitates the verification of dimensional homogeneity using SPARQL queries, as discussed earlier (cf. requirement R08).

We decided to adapt the naming conventions specified by QUDT for labelling dimension vectors. For instance, the acceleration dimension, defined by the expression LT^{-2}, is represented by the label 'N0I0L1J0M0H0T-2', where the numbers correspond to the powers of each base dimension. For the data properties stating the power of each base dimension, we followed the naming convention adopted by OM. All individuals representing dimension vectors are asserted to the class uomcore:Dimension which is defined as a subclass of iso:InformationObject. For reasons of simplicity, we do not include dimensions in the modelling example depicted in Fig. 1.

3.4 Quantity Kinds

The question of how to model quantity kinds has created a significant controversy between the authors of this work. This is also a source of disagreement for the designers of OM and QUDT, as they followed different design patterns for representing quantity kinds. We considered the following design patterns: *quantity kinds as individuals*, *quantity kinds as classes* and *quantity kinds as properties*. For a more in-depth discussion about these three design patterns, we recommend interested readers the overview given by Rijgersberg et al. [5].

Quantity Kinds as Individuals. This is the preferable modelling choice in QUDT. In our ontology, we also defined quantity kinds as individuals, as in QUDT, but these individuals are defined as SKOS concepts and related to specific SKOS concept schemes. The purpose of this approach is to map our quantity kinds to similar terms defined by external data sources using SKOS semantic properties (cf. requirement R10). This is also expected by ontologies following ISO 15926-14 (cf. requirement R09).

All quantity kinds in QUDT (e.g. quantitykind:DRY-MASS) are defined using named individuals and asserted to the class qudt:QuantityKind. To indicate that a quantity kind, such as quantitykind:DRY-MASS, is more general (or more specific) than other quantity kinds, such as quantitykind:Mass, QUDT uses the properties qudt:generalization and skos:broader (or qudt:specialization and skos:narrower). In our example, the individual ex:AS17DE110DryMass would be asserted to the class qudt:Quantity and it would be related to the quantity kind quantitykind:DRY-MASS using the object property qudt:hasQuantityKind.

Quantity Kinds as Classes. This is the preferable modelling choice in OM and also implemented in our ontology of units of measure. This is also the recommended design pattern for representing quantities in ISO 15926-14 and BFO

upper ontologies (cf. requirement R09). Both upper ontologies define qualities as classes representing *dependent continuants* [1], meaning that the existence of a quality or a physical quantity depends on the existence of its bearer. For instance, in the modelling example depicted in Fig. 1, the induction motor represented by the individual ex:AS17DE110 has the physical quantity dry mass and this is stated by a property assertion with the individual ex:AS17DE110DryMass.

As a result of this modelling approach, we created a hierarchy of classes representing quantity kinds, where iso:PhysicalQuantity is the top class. Subclass relations for quantity kinds are determined by the notion of dimensional homogeneity. For instance, the class uomqk:DryMass is defined as a subclass of uomqk:Mass. This is because both have the same dimension, represented by the individual (not included in Fig. 1), and uomqk:DryMass is more specific than uomqk:Mass.

Quantity Kinds as Properties. This design pattern was supported in earlier releases of OM [5] and is the preferred representation in the Sesam datahub. As in the case of quantity kinds represented as individuals, this design pattern does not follow ISO 15926-14 (or BFO) but it does not prevent integration with this upper ontology (cf. requirement R09). As with quantity kinds implemented as classes, we can create a hierarchy of properties representing quantity kinds, where iso:hasPhysicalQuantity is the top property. Subproperty relations for quantity kinds are also determined by the notion of dimensional homogeneity. For instance, the object property uomqk:hasDryMass is defined as a subproperty of uomqk:hasMass, as both have the same dimension. A clear advantage of this pattern is that many users are more comfortable working with properties than classes, where properties define meaningful relations between objects (or events) and quantity values. For instance, an object representing a particular induction motor is related to a quantity value representing a measurement of its dry mass using the property uomqk:hasDryMass.

3.5 Units of Measure

As discussed earlier, unit conversion using SPARQL is easier in QUDT than in OM (cf. requirement R08). Therefore, we decided to define units of measure using the same conversion multipliers and offsets. We explicitly indicated which unit is related to the conversion values in the definition of each unit, using the object property uomcore:hasReferenceUnit. QUDT is also used as the main reference for the identifiers of the units even though OM defines more intuitive identifiers, especially for simple units. This is because the identifiers defined in QUDT are more compact, particularly for compound units.

The designers of QUDT made a reasonable attempt at defining a generic class of units of measure, qudt:Unit. We adapted this by improving or removing some class restrictions. This includes, for instance, deleting unneeded references to latex or MathML expressions, simplifying the definition of units. In addition, we included all the subclasses of qudt:Unit defined by QUDT and we extended this class hierarchy with the OM class om:CompoundUnit and its direct subclasses.

The definitions have been simplified to improve reasoning performance, as discussed (cf. requirement R06).

To facilitate the verification of dimensional homogeneity (i.e. if two units are commensurable) using SPARQL (cf. requirement R08), we adopted the approach implemented by OM that relates each unit to a suitable dimension vector. In addition, we related each unit of measure to quantity kinds of the same dimensions and to relevant base quantities and units (when defining derived units only).

4 Implementation and Build Process

After analysing and selecting suitable modelling choices for the design of the UoM library, we focus on how to optimise its implementation by reducing manual efforts, facilitating collaborative development and ensuring uniform modelling. One of the key ingredients to achieving this is the use of ontology templates for representing ontology design patterns in the form of parameterised ontologies [6]. In this section, we discuss how we produced a tailor-made UoM library for the Integral platform, based on QUDT and OM ontologies, where it is instrumental to fulfil requirements R04, R05 and R06. Figure 2 outlines the main phases of the process, including *Preprocessing*, *Creation* and *Build & Deployment*. The input and output (results) of each phase are represented in the upper part and the tools used to produce the output in the lower part.

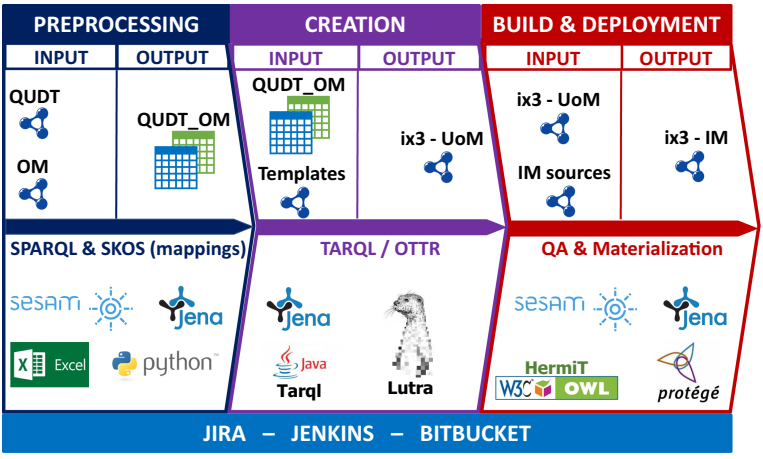

Fig. 2. Implementation and build process

Pre-processing. The assessment of the requirement R07 helped us to identify the relevant design patterns in QUDT and OM. These patterns provide a useful guideline when defining the graph patterns of the SELECT SPARQL queries we

executed to extract information from QUDT and OM. The new design patterns, defined for the UoM Library, also helped us to establish the output of these queries.

The results of the execution of these SELECT SPARQL queries were stored in several Comma-separated values (CSV) files (one for each design pattern and represented in Fig. 2 by the tables *QUDT_OM*). These files were processed further to make them suitable for populating the UoM ontology library using ontology templates. For instance, each term stored in the CSV files received an identifier, an IRI. Then new columns were added to enrich the information and to state, for instance, that each relevant term is defined as an OWL named individual and as a SKOS concept. This process was done using common tools such as Microsoft Excel, Python and Jena.

The main challenge was, and still is, to efficiently align and combine information from the ontologies OM and QUDT. To do this, we created a specific CSV file where similar terms in OM and QUDT had been identified and mapped using SKOS semantic properties. We applied algorithms (in Python) for detecting similar strings in the labels of each term. The resulting mapping list was manually refined, which is an expensive process. We expect that new versions of Sesam data hub can soon take care of this process and significantly reduce manual work.

Creation (from Tarql to OTTR). After we collected, refined and stored QUDT and OM data in CSV files, we generated the various ontologies of the UoM library by instantiating ontology templates. Currently, we are using Tarql,[29] to define templates (as SPARQL queries) and instantiate these templates (by running these queries). We adopted Tarql due to its simplicity, but discovered several limitations when building a library or reusable templates. Therefore, we turned our attention to Reasonable Ontology Templates (OTTR) [6], a language for representing ontology templates. In combination with the tool Lutra,[30] it is possible to instantiate OTTR templates from tabular data and produce RDF graphs and OWL ontologies. In the future, it may be possible to define OTTR templates for querying data from QUDT and OM ontologies and avoid the use of CSV files. We defined several OTTR templates to represent the design patterns associated to quantity kinds, units, dimension vectors and quantity values. We are currently testing these OTTR templates with data from QUDT and OM, and the preliminary results look promising. We hope we can shortly introduce OTTR templates when building the Information Model at ix3.

Build and Deployment. ix3 has designed and implemented an infrastructure and a build process to deliver and deploy new releases of the Information Model that includes the UoM library. Changes in the source files of the Information Model are requested and discussed in Jira,[31] a tool for issue tracking and project

[29] https://tarql.github.io/.
[30] https://gitlab.com/ottr/lutra/lutra.
[31] https://www.atlassian.com/software/jira.

management. Jira is integrated with Jenkins[32] and Bitbucket,[33]. Jenkins takes care of the distribution and deployment of new releases of the Information Model, and Bitbucket hosts the different files needed to build the Information Model and facilitating the collaborative edition of these files.

During the build process of a new release of the Information Model, several quality assurance and refining tasks are executed. This includes, for instance, the validation of the syntax of the different RDF files produced using the tool Jena. Consistency of the OWL ontologies delivered in the release is tested using the HermiT reasoner, and the classification of the ontology (class and property hierarchies) is computed and materialised (cf. requirements R04, R05 and R06). Materialisation of subclass and subproperty relations becomes valuable when executing SPARQL in the triple stores where the Information Model resides. In addition, class and property assertions are inferred and materialised to speed up SPARQL queries. Manual inspection using (Web) Protégé is also done as part of the quality assurance process (cf. requirement R05).

5 Conclusion

In this paper, we discuss how we extended the Information Model, the ontology library supporting the ix3 Integral platform for digital twins, to better support the contextualisation and harmonisation of quantity values and units of measures. To minimise development costs, we tried to reuse the ontologies QUDT and OM. Although they are excellent resources, they did not meet all our requirements. Therefore, we decided to create a new ontology based on QUDT and OM. We started by analysing suitable ontology design patterns and we implemented the selected modelling patterns using ontology templates. By applying these templates and specific SPARQL queries as part of the implementation and build processes at ix3, we reduced manual effort and we ensured uniform modelling of the new ontology of units of measure. The latest release of this ontology is currently integrated and deployed as part of the Integral platform. Despite the resulting ontology not being released under a public licence, we can share our experience gained with this work. This is to help practitioners who aim to incorporate and adapt external resources to their ontology libraries, and who seek to benefit from using ontologies for contextualisation and harmonisation of quantities and units of measure.

References

1. Arp, R., Smith, B., Spear, A.D.: Building Ontologies with Basic Formal Ontology. MIT Press, Cambridge (2015)
2. Gangemi, A., Presutti, V.: Ontology design patterns. In: Staab, S., Studer, R. (eds.) Handbook on Ontologies. IHIS, pp. 221–243. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-540-92673-3_10

[32] https://www.jenkins.io/.
[33] https://bitbucket.org/.

3. Keil, J.M., Schindler, S.: Comparison and evaluation of ontologies for units of measurement. Semant. Web J. **10**(1), 33–51 (2019)
4. Noy, N., Rector, A., Hayes, P., Welty, C.: Defining N-ary relations on the semantic web (W3C working note). Technical report, World Wide Web Consortium (W3C) (2006). https://www.w3.org/TR/swbp-n-aryRelations/
5. Rijgersberg, H., van Assem, M., Top, J.: Ontology of units of measure and related concepts. Semant. Web J. **4**(1), 3–13 (2013)
6. Skjæveland, M.G., Lupp, D.P., Karlsen, L.H., Forssell, H.: Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates. In: Vrandečić, D., et al. (eds.) ISWC 2018. LNCS, vol. 11136, pp. 477–494. Springer, Cham (2018). https://doi.org/10.1007/978-3-030-00671-6_28