ETSI TR 103 781 V1.1.1 (2023-09)

SmartM2M;
Study for SAREF ontology patterns and usage guidelines

Reference
DTR/SmartM2M-103781

Keywords
interoperability, IoT, IoT platforms, oneM2M,
SAREF, semantic, smart lift

ETSI

650 Route des Lucioles
F-06921 Sophia Antipolis Cedex - FRANCE

Tel.: +33 4 92 94 42 00 Fax: +33 4 93 65 47 16

Siret N° 348 623 562 00017 - APE 7112B
Association à but non lucratif enregistrée à la
Sous-Préfecture de Grasse (06) N° w061004871

Contents

Intellectual Property Rights

Essential patents

IPRs essential or potentially essential to normative deliverables may have been declared to ETSI. The declarations pertaining to these essential IPRs, if any, are publicly available for **ETSI members and non-members**, and can be found in ETSI SR 000 314: *"Intellectual Property Rights (IPRs); Essential, or potentially Essential, IPRs notified to ETSI in respect of ETSI standards"*, which is available from the ETSI Secretariat. Latest updates are available on the ETSI Web server (https://ipr.etsi.org/).

Pursuant to the ETSI Directives including the ETSI IPR Policy, no investigation regarding the essentiality of IPRs, including IPR searches, has been carried out by ETSI. No guarantee can be given as to the existence of other IPRs not referenced in ETSI SR 000 314 (or the updates on the ETSI Web server) which are, or may be, or may become, essential to the present document.

Trademarks

The present document may include trademarks and/or tradenames which are asserted and/or registered by their owners. ETSI claims no ownership of these except for any which are indicated as being the property of ETSI, and conveys no right to use or reproduce any trademark and/or tradename. Mention of those trademarks in the present document does not constitute an endorsement by ETSI of products, services or organizations associated with those trademarks.

DECT™, **PLUGTESTS™**, **UMTS™** and the ETSI logo are trademarks of ETSI registered for the benefit of its Members. **3GPP™** and **LTE™** are trademarks of ETSI registered for the benefit of its Members and of the 3GPP Organizational Partners. **oneM2M™** logo is a trademark of ETSI registered for the benefit of its Members and of the oneM2M Partners. **GSM**® and the GSM logo are trademarks registered and owned by the GSM Association.

Foreword

This Technical Report (TR) has been produced by ETSI Technical Committee Smart Machine-to-Machine communications (SmartM2M).

Modal verbs terminology

In the present document "**should**", "**should not**", "**may**", "**need not**", "**will**", "**will not**", "**can**" and "**cannot**" are to be interpreted as described in clause 3.2 of the ETSI Drafting Rules (Verbal forms for the expression of provisions).

"**must**" and "**must not**" are **NOT** allowed in ETSI deliverables except when used in direct citation.

1 Scope

The present document reports on the modularization and factorization potential of the TC SmartM2M suite using reference ontology patterns. The present document lists identified modelling discrepancies in SAREF extensions, along with proposals to homogenise the modelling. As a result, the present document describes a set of core ontology patterns and how they can be used as a basis for future normative work in TC SmartM2M.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

References are either specific (identified by date of publication and/or edition number or version number) or non-specific. For specific references, only the cited version applies. For non-specific references, the latest version of the referenced document (including any amendments) applies.

NOTE: While any hyperlinks included in this clause were valid at the time of publication ETSI cannot guarantee their long term validity.

The following referenced documents are not necessary for the application of the present document but they assist the user with regard to a particular subject area.

[i.1] ETSI TS 103 548: "SmartM2M; SAREF reference ontology patterns".

[i.2] ETSI TR 103 549: "SmartM2M; Guidelines for consolidating SAREF with new reference ontology patterns, based on the experience from the ITEA SEAS project".

[i.3] ETSI TS 103 673: "SmartM2M; SAREF Development Framework and Workflow, Streamlining the Development of SAREF and its Extensions".

[i.4] ETSI TS 103 264 (V3.1.1): "SmartM2M; Smart Applications; Reference Ontology and oneM2M Mapping".

[i.5] ETSI TS 103 410-1 (V1.1.2): "SmartM2M; Extension to SAREF; Part 1: Energy Domain".

[i.6] ETSI TS 103 410-2 (V1.1.2): "SmartM2M; Extension to SAREF; Part 2: Environment Domain".

[i.7] ETSI TS 103 410-3 (V1.1.2): "SmartM2M; Extension to SAREF; Part 3: Building Domain".

[i.8] ETSI TS 103 410-4 (V1.1.2): "SmartM2M; Extension to SAREF; Part 4: Smart Cities Domain".

[i.9] ETSI TS 103 410-5 (V1.1.2): "SmartM2M; Extension to SAREF; Part 5: Industry and Manufacturing Domains".

[i.10] ETSI TS 103 410-6 (V1.1.2): "SmartM2M; Extension to SAREF; Part 6: Smart Agriculture and Food Chain Domain".

[i.11] ETSI TS 103 410-7 (V1.1.1): "SmartM2M; Extension to SAREF; Part 7: Automotive Domain".

[i.12] ETSI TS 103 410-8 (V1.1.1): "SmartM2M; Extension to SAREF; Part 8: eHealth/Ageing-well Domain".

[i.13] ETSI TS 103 410-9 (V1.1.1): "SmartM2M; Extension to SAREF; Part 9: Wearables Domain".

[i.14] ETSI TS 103 410-10 (V1.1.1): "SmartM2M; Extension to SAREF; Part 10: Water Domain".

[i.15] ETSI TS 103 410-11 (V1.1.1): "SmartM2M; Extension to SAREF; Part 11: Lift Domain".

[i.16] Ontology Design Patterns website.

[i.17] W3C® Recommendation 31 March 2013: "SPARQL 1.1 Query Language".

[i.18] W3C® Recommendation 20 July 2017: "Shapes Constraint Language (SHACL)".

[i.19] María Poveda-Villalón, Serge Chávez-Feria, & Raúl García-Castro. (2020): Chowlk visual notation (v0.1.0), Zenodo.

[i.20] María Poveda-Villalón, Serge Chávez-Feria, & Raúl García-Castro: Chowlk framework.

[i.21] Serge Chávez-Feria, María Poveda-Villalón, & Raúl García-Castro. (2020): Chowlk Converter (v0.0.1), Zenodo.

[i.22] Osumi-Sutherland, D., Courtot, M., Balhoff, J. P., & Mungall, C. (2017). Dead simple OWL design patterns. Journal of biomedical semantics, 8(1), 1-7.

[i.23] "Dead simple owl design pattern (DOS-DP) exchange format".

[i.24] CEUR Workshop Proceedings: "The Semantic Web takes wing: Programming ontologies with Tawny-OWL". Lord P. (2013). In: Rodriguez-Muro M, Jupp S, Srinivas K, editors. Proceedings of the 10th International Workshop on OWL: Experiences and Directions (OWLED 2013). Volume 1080. Aachen; 2013. /Vol-1080.

[i.25] Egana M, Stevens R, Antezana E. (2009): "Transforming the axiomisation of ontologies: The ontology pre-processor language". In: CEUR Workshop Proceedings. vol. 496; 2009. doi:10.1038/npre.2009.4006.1.

[i.26] Skjæveland, M. G., Lupp, D. P., Karlsen, L. H., & Forssell, H. (2018): "Practical ontology pattern instantiation, discovery, and maintenance with reasonable ontology templates". In The Semantic Web-ISWC 2018: 17th International Semantic Web Conference, Monterey, CA, USA, October 8-12, 2018, Proceedings, Part I 17 (pp. 477-494). Springer International Publishing.

[i.27] Reasonable Ontology Templates (OTTR).

[i.28] Shimizu, C., Hammar, K., & Hitzler, P. (2023). Modular ontology modeling. Semantic Web, 14(3), 459-489.

[i.29] CoModIDE: "The Comprehensive Modular Ontology IDE -- a Protégé plugin".

[i.30] W3C® Working Group Note 08 June 2017: " SHACL Advanced Features".

[i.31] IETF BCP 47 (RFC 5646) : "Tags for Identifying Languages", Phillips, A., Ed., and M. Davis, Ed., September 2009".

3 Definition of terms, symbols and abbreviations

3.1 Terms

For the purposes of the present document, the terms given in ETSI TS 103 673 [i.3] and the following apply:

basic graph pattern: set of generalized RDF triples

blank node: anonymous node with a label

NOTE: The label is local to the document where the blank node is used.

generalized RDF triple: triple of RDF terms

ontology design pattern: generic solution to a recurring ontology modelling problem

RDF Literal: combination of a Unicode string (the lexical form), a datatype IRI, and optionally, if the datatype is `rdf:langString`, a language tag following IETF BCP 47 [i.31]

RDF term: set of RDF terms is the union of the sets of IRIs, Blank nodes, Literals, and Variables

3.2 Symbols

Void.

3.3 Abbreviations

For the purposes of the present document, the following abbreviations apply:

BCP	Best Current Practice
BGP	Basic Graph Pattern
CoMoIDE	Comprehensive Modular Ontology IDE
DL	Description Logics
DOS-DP	Dead Simple OWL Design Patterns
DP	Datatype Property
DUL	Dulce Ultralite Ontology
FOAF	Friend Of A Friend
IETF	Internet Engineering Task Force
IoT	Internet of Things
IPR	Intellectual Property Right
IRI	Internationalized Resource Identifier
MOMo	Modular Ontology modeling
ODP	Ontology Design Pattern
OGC	Open Geospatial Consortium
OM	Ontology of Measurement
OP	Object Property
OTTR	Reasonable Ontology Templates
OWL	Web Ontology Language
RDF	Resource Description Framework
RDFS	RDF Schema
SAREF	Smart Applications REFerence ontology
SEAS	Smart Energy Aware Systems
SHACL	Shape and Constraint Language
SKOS	Simple Knowledge Organization System
SOSA	Sensor, Observation, Sample, and Actuator ontology
SPARQL	SPARQL Protocol and RDF Query Language
SSN	Semantic Sensor Network ontology
STF	Specialist Task Force
TR	Technical Report
TS	Technical Specification
UML	Unified Modeling Language
URL	Universal Resource Locator
W3C®	World Wide Web Consortium

4 Ontology Design Patterns and their usage in SAREF

4.1 Ontology Design Patterns

Ontology design patterns are generic solutions to recurring ontology modelling problems.

The taxonomy of ontology design pattern types is summarized in [i.16] as follows:

- Structural ODPs:
  - **Logical ODPs** are formal expressions, whose only parts are expressions from a logical vocabulary, e.g. OWL DL, that solves a problem of expressivity. Logical ODPs solve design problems where the primitives of the representation language do not directly support certain logical constructs.

  - **Architectural ODPs** affect the overall shape of the ontology: their aim is to constrain 'how the ontology should look like'.

- Correspondence ODPs:
  - **Alignment ODPs** refer to correspondences between ontologies. Each pattern models a relation between two entities or sets of entities in two ontologies. Instantiation of an Alignment OP results in a correspondence between elements of two given ontologies.
  - **Reengineering ODPs** are transformation rules applied in order to create a new ontology (target model) starting from elements of a source model.

- **Content ODPs** are distinguished networked ontologies and have their own namespace. They cover a specific set of competency questions (requirements), which represent the problem they provide a solution for. Furthermore, Content ODPs show certain characteristics, i.e. they are: computational, small, autonomous, hierarchical, cognitively relevant, linguistically relevant, and best practices.
- **Reasoning ODPs** are applications of Logical ODPs oriented to obtain certain reasoning results, based on the behavior implemented in a reasoning engine.
- **Presentation ODPs** deal with usability and readability of ontologies from a user perspective:
  - **Naming ODPs** are conventions on how to create names for namespaces, files, and ontology elements in general (classes, properties, etc.).
  - **Annotation ODPs** provide annotation properties or annotation property schemas that are meant to improve the understandability of ontologies and their elements.
- **Lexico-Syntactic ODPs** are linguistic structures or schemas that consist of certain types of words following a specific order, and that permit to generalize and extract some conclusions about the meaning they express.

4.2 Ontology Design Patterns in SAREF

SAREF already adopts certain ODPs from the categories listed in clause 4.1. This clause provides some examples.

ETSI TS 103 673 [i.3] specifies naming ODPs for namespaces and ontology elements in general. For example, all SAREF extension project namespaces look the same with a four-letter code, SAREF versions follow the same format, SAREF class local names are Camel Case.

ETSI TS 103 673 [i.3] also specifies annotation ODPs for the annotation property schemas that should be used in SAREF ontologies. For example all SAREF entities have a `rdfs:label` and a `rdfs:comment` annotation property.

The SAREF Documentation generation tool in the SAREF Pipeline uses Lexico-Syntactic ODPs as human-readable descriptions of the axioms in the ontology.

SAREF Core [i.4] specifies content ODPs, and provides some extensions for the top-level classes. For example, it specifies how devices and services are described, and illustrates how the top-level classes `saref:Device` and `saref:Service` can be specialized as `saref:SmokeSensor` and `saref:SwitchOnService`.

Implicit architectural patterns in SAREF Core can be made explicit. For example, axioms of type `MinCardinality 1` are avoided, and instead axioms of type `SomeValuesFrom` are used.

Some implicit Naming ODPs are used to name inverse properties in SAREF Core. For example:

- `saref:accomplishes - saref:isAccomplishedBy ; saref:offers – saref:isOfferedBy`
- `saref:hasMeasurement – saref:isMeasurementOf, saref:hasCommand – saref:isCommandOf`

SAREF Extensions specify additional content ODPs, and provide some extensions for the top-level classes in SAREF.

Some implicit Naming ODPs are used to name subclasses of top-level classes `saref:Measurement` and `saref:Property`. In general, their local name ends with "Measurement" or "Property".

SAREF4ENVI [i.6] defines architectural ODPs that, given a property (Height, Frequency, Period), defines a sub-class of `saref:Property`, a sub-class of `saref:Measurement`, and a sub-class of `saref:UnitOfMeasure` of it. The definitions always look parallel:

- `s4envi:FrequencyMeasurement`: "Represents the measured value made over a frequency property. It is also linked to the frequency unit of measure in which the value is expressed and the timestamp of the measurement".
- `s4envi:HeightMeasurement`: "Represents the measured value made over a height property. It is also linked to the height unit of measure in which the value is expressed and the timestamp of the measurement".
- `s4envi:PeriodMeasurement`: "Represents the measured value over a period property. It is also linked to the period unit of measure in which the value is expressed and the timestamp of the measurement".

In addition, SAREF4ENVI uses a Naming ODP: the local name for these three measurement sub-classes end with string "Measurement" (`s4envi:HeightMeasurement`, `s4envi:FrequencyMeasurement`, `s4envi:PeriodMeasurement`).

SAREF4SYST [i.1] defines content ODPs, and further specifies architectural ODPs for how these content ODPs may be specialized.

4.3 Degrees of Applicability of Content ODPs

Content ODPs may apply more or less for a specific domain and use-case:

- *Exact match*: the pattern function and terminology fit the use-case exactly; the pattern has been applied in a SAREF extension with the same domain as the project.
- *Incomplete match*: the pattern function and terminology fit the use-case very well, but some other restrictions or properties are needed to complete the pattern; the pattern have to be extended or specialized.
- *Similar match*: the pattern function fits the use-case well enough, but the terminology is a little different or the domain is not the same or the pattern is not intuitive for the domain.
- *Composite pattern needed*: a combination of two or more existing patterns is needed.
- *No pattern match*: nothing seems to fit.

For example in SAREF4ENVI, the unit for `s4envi:PeriodMeasurement` seems to be an exception. It uses existing entity `time:TemporalEntity` instead of defining a `s4envi:PeriodUnit` class.

4.4 Types of Logical axioms in SAREF

Which type of logical axiom is used, and how they are typically used, can be considered architectural ODPs.

Using `saref:Property` as an illustrative example, a class can serve:

- In the definition of a subclass (e.g. `:MyNewClass rdf:type owl:Class; rdfs:subClassOf saref:Property`); that is, when the new class exactly matches the pattern or will extend or specialize the pattern.
- In an **AllValuesFrom** restriction (e.g. `[] rdf:type owl:Restriction; owl:onProperty saref:controlsProperty; owl:allValuesFrom saref:Property .`); that is, to describe a class of all individuals for which all values of the property under consideration are either members of the class extension of the class description or are data values within the specified data range.
- In the definition of an **NamedIndividual** (e.g. `s4agri:SoilMoisture rdf:type owl:NamedIndividual, saref:Property`); that is, to declare a instance of that type (`saref:Property`).
- As a **Domain** or **Range** of an Object Property (e.g. `rdfs:domain saref:FeatureOfInterest; rdfs:range saref:Property`). Note that if multiple domains or ranges are specified, the domain or range is the intersection of them -- the items in the list are combined with the AND logical operator.

- Or even as in local restrictions such as **AllValuesFrom**, **SomeValuesFrom**, **QualifiedCardinalityRestriction**.

Table 1 below summarizes the types of OWL axioms used by SAREF Core and each of the SAREF extensions.

Table 1: Type of axioms used in each SAREF project

Type of axiom	Core [i.4]	ENER [i.5]	ENVI [i.6]	BLDG [i.7]	CITY [i.8]	INMA [i.9]	AGRI [i.10]	AUTO [i.11]	EHAW [i.12]	WEAR [i.13]	WATR [i.14]	LIFT [i.15]	SYST [i.1]
Datatype oneOf		X							X				
Class IntersectionOf										X			
Class UnionOf	X	X		X		X							
Class ComplementOf													
Class oneOf		X											
Class DisjointFrom	X		X *also on classes from other onto logies*	X *on classes from other onto logies*	X *on classes from other onto logies*	X	X *on classes from other onto logies*	X *on classes from other onto logies*			X *on classes from other onto logies*		X
OP DisjointFrom													
OP InverseOf	X		X	X	X	X	X	X		X			X
OP Domain	X				X *on classes from other onto logies*		X *on classes from other onto logies*	X *on classes from other onto logies*	X	X *hijacking saref:hasFunction*	X	X	X
OP Range	X					X		X	X	X	X	X	X
OP Functional			X										X
OP InverseFunctional			X										
OP Reflexive													
OP Irreflexive													
OP Symmetric			X										X
OP Asymmetric													
OP Transitive			X	X									X
OP SomeValuesFrom	X		X			X	X	X		X	X	X	X
OP AllValuesFrom	X	X	X	X	X	X	X	X	X	X	X		
OP HasValue			X										
OP HasSelf													

Type of axiom	Core [i.4]	ENER [i.5]	ENVI [i.6]	BLDG [i.7]	CITY [i.8]	INMA [i.9]	AGRI [i.10]	AUTO [i.11]	EHAW [i.12]	WEAR [i.13]	WATR [i.14]	LIFT [i.15]	SYST [i.1]
OP MinCardinality													
OP MinQualifiedCardinality	X	X				X							
OP MaxCardinality													
OP MaxQualifiedCardinality		X					X	X					
OP Cardinality	X	X			X								X
DP SomeValuesFrom						X	X	X				X	
DP AllValuesFrom	X	X	X	X	X	X	X	X	X	X			
DP HasValue													
DP MinCardinality		X (min 0)										X (min 1)	
DP MinQualifiedCardinality		X											
DP MaxCardinality	X	X					X				X		
DP MaxQualifiedCardinality			X				X						
DP Range		X	X	X	X	X	X	X	X	X	X	X	
DP Functional			X										
Keys													

Some of these logical axioms used by extensions are in fact bad practices and should be avoided. For example, SAREF4ENER V1.1.2 and SAREF4EHAW V1.1.1 use the value partitions & sets Logical ODPs on datatype properties. `s4ener:valueSource`, s4ener:valueTendency, s4ener:powerSource, s4ener:messagingType, s4ehaw:banTopology, s4ehaw:hasGender, all define a range that is a OneOf anonymous class (a class composed of a set of individuals or literals). This forbids extensions to define more possible values as the object of these properties.

From Table 1, one may also conclude the following:

- SAREF4ENER V1.1.2 uses different modelling choices than most other extensions.
- SAREF4ENVI provides the richest set of OWL axioms.
- Some ontologies redefine entities from external ontologies, and add axioms to these entities. This should be discouraged as it is error prone, and may lead to incoherence if the reused ontology evolves. For example the range of `saref:hasValue`, which was defined as `xsd:float` in V2.1.1, has been removed in V3.1.1 to support other datatypes for measurements. Extensions that copy-pasted parts of SAREF still define the old range for `saref:hasValue`.

In general, some further harmonization across the different SAREF projects would be useful.

5 Specification Frameworks for Ontology Patterns

5.1 Introduction

The present clause presents several approaches for specifying ontology patterns through specification frameworks.

In general, ontology pattern specification frameworks help to factorize the description of ontologies, and contribute to satisfy the "Don't repeat yourself" (DRY) principle of software development. This principle aims at reducing repetition of information which is likely to change, replacing it with abstractions that are less likely to change.

Ontology pattern specification frameworks may simplify the development of SAREF extension ontologies and consolidate them into a more homogeneous and predictable structure. Applying these ontology patterns also enhances semantic interoperability and facilitates inference. These patterns expedite ontology development through pattern modularity, composability, reuse, and extendibility.

5.2 RDF Basic Graph Patterns

An RDF Basic Graph Pattern (BGP) is an RDF graph where variables can be used in any position of any triple.

By substituting variables with IRI, Blank Nodes, or Literals, a BGP can be transformed into an RDF graph.

For example, the following BGP can be used to partially factorize SAREF4ENVI:

```
?property rdf:type owl:NamedIndividual, saref:Property ;
    rdfs:comment ?propertyComment ;
    rdfs:label ?propertyLabel .
?measurement rdf:type owl:Class ;
    rdfs:subClassOf saref:Measurement ,
        [ rdf:type owl:Restriction ;
          owl:onProperty saref:relatesToProperty ;
          owl:hasValue ?property
        ] ,
        [ rdf:type owl:Restriction ;
          owl:onProperty saref:isMeasuredIn ;
          owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
          owl:onClass ?unit
        ] ;
    rdfs:comment ?measurementComment ;
    rdfs:label ?measurementLabel.
?unit rdf:type owl:Class ;
    rdfs:subClassOf saref:UnitOfMeasure ;
    rdfs:comment ?unitComment ;
    rdfs:label ?unitLabel .
```

SAREF4ENVI contains the result of substituting variables in this BGP with each of the three mappings below:

```
?property = s4envi:Height
?propertyComment = "An individual representing the property height."@en
?propertyLabel = "Height"@en
?measurement = s4envi:HeightMeasurement
?measurementComment = "Represents the measured value made over a height property. It is also linked
to the height unit of measure in which the value is expressed and the timestamp of the
measurement."@en
?measurementLabel = "Height measurement"@en
?unit = :LengthUnit
?unitComment = "Unit of measure for the quantity length."@en
?unitLabel = "Length unit"@en

?property = s4envi:Period
?propertyComment = "An individual representing the property period."@en
?propertyLabel = "Period"@en
?measurement = s4envi:PeriodMeasurement
?measurementComment = "Represents the measured value made over a period property. It is also linked
to the period unit of measure in which the value is expressed and the timestamp of the
measurement."@en
?measurementLabel = "Period measurement"@en
?unit = time:TemporalUnit
?unitComment = "A temporal unit of measure, which provides a scale factor for a time quantity."@en
?unitLabel = "Temporal unit"@en .
```

```
?property = s4envi:Frequency
?propertyComment = "An individual representing the property frequency."@en
?propertyLabel = "Frequency"@en
?measurement = s4envi:FrequencyMeasurement
?measurementComment = "Represents the measured value made over a frequency property. It is also
linked to the frequency unit of measure in which the value is expressed and the timestamp of the
measurement."@en
?measurementLabel = "Frequency measurement"@en
?unit = s4envi:FrequencyUnit
?unitComment = "Unit of measure for the quantity frequency.@en
?unitLabel = "Frequency unit"@en .
```

BGPs help to reduce repetitive information in the ontology, but there are still many repetitions that could be avoided, which provoke mistakes and typos (see "length" in ?unitComment of the first mapping).

5.3 RDF Basic Graph Patterns with Templates

To further reduce the redundancy, many variables in BGPs could be mapped to the result of applying transformation functions on a reduced set of variables.

The snippet below assumes that the set of RDF Terms is augmented with a set of IRI templates such as f<{expr}>, a set of literal templates such as f"{expr}", expression nodes ?{expr}, where expr is an expression over RDF Terms:

```
f<{ns}{transformForIRI(p)}> rdf:type owl:NamedIndividual , saref:Property ;
    rdfs:comment "An individual representing the property {transformForComment(p)}."@en ;
    rdfs:label "{transformForLabel(p)}"@en.
f<{ns}{transformForIRI(p)}Measurement> rdf:type owl:Class ;
    rdfs:subClassOf saref:Measurement ,
        [ rdf:type owl:Restriction ;
          owl:onProperty saref:relatesToProperty ;
          owl:hasValue <{ns}{transformForIRI(p)}>
        ] ,
        [ rdf:type owl:Restriction ;
          owl:onProperty saref:isMeasuredIn ;
          owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
          owl:onClass ?unit
        ] ;
    rdfs:comment "Represents the measured value made over a {transformForComment(p)} property. It is
also linked to the {transformForComment(p)} unit of measure in which the value is expressed and the
timestamp of the measurement."@en ;
    rdfs:label "{transformForLabel(p)} measurement"@en.
?unit rdf:type owl:Class ;
    rdfs:subClassOf saref:UnitOfMeasure ;
    rdfs:comment ?{commentForUnit(?unit)};
    rdfs:label "{transformForLabel(p)} unit"@en.
```

Then SAREF4ENVI would contains the result of substituting variables in these BGP with Templates with each of the three reduced mappings below:

```
?ns = "https://saref.etsi.org/saref4envi/"
?p = "Height"
?unit = :LengthUnit

?ns = "https://saref.etsi.org/saref4envi/"
?p = "Period"
?unit = time:TemporalUnit
?unitComment = "A temporal unit of measure, which provides a scale factor for a time quantity."@en

?ns = "https://saref.etsi.org/saref4envi/"
?p = "Frequency"
?unit = f<{ns}{transformForIRI(p)}Unit>
```

This solution requires the definition of a few transformation functions transformForIRI transformForLabel transformForComment, and a function commentForUnit that returns ?unitComment if it is bound, else it returns "Unit of measure for the quantity {transformForComment(p)}.@en.

The BGP with Template approach is bound to domain-specific use-cases. Their general reuse is thus tied to a certain level of abstraction.

5.4 SPARQL SELECT Queries

BGPs can be wrapped within a SPARQL query [i.17]. The query selects for triples where the variables match the defined pattern. The results are variable bindings which are defined by value: URIs, literals, and FILTER rules. Thus, each non-trivial SPARQL query can be thought of as a design pattern specification that, when executed on a RDF graph, identifies the occurrences of that pattern. Note that SPARQL SELECT queries does not return a RDF graph like CONSTRUCT and DESCRIBE do.

Here is an example that selects all the sub-classes of `saref:Measurement`:

```
PREFIX saref: <https://saref.etsi.org/core/>
PREFIX owl: <http://www.w3.org/2002/07/owl#>
SELECT ?measurement WHERE {
  ?measurement rdf:type owl:Class ;
    rdfs:subClassOf saref:Measurement ,
        [ rdf:type owl:Restriction ;
          owl:onProperty saref:relatesToProperty ;
          owl:hasValue ?property
        ] ,
        [ rdf:type owl:Restriction ;
          owl:onProperty saref:isMeasuredIn ;
          owl:qualifiedCardinality "1"^^xsd:nonNegativeInteger ;
          owl:onClass ?unit
    ].
}
```

measurement	property	unit
s4envi:HeightMeasurement	s4envi:Height	s4envi:LengthUnit
s4envi:PeriodMeasurement	s4envi:Period	time:TemporalUnit
s4envi:FrequencyMeasurement	s4envi:Frequency	s4envi:FrequencyUnit

SPARQL Select queries does not allow to generate ontologies, but can be used to identify the occurrences of patterns in ontologies. SPARQL Construct queries could be used to generate ontologies.

5.5 SPARQL Rules

The SPARQL entailment rules approach defines a domain-specific ontology pattern that returns RDF subgraphs that are not explicitly defined within the RDF triple-set, but which can be inferred based upon IF … THEN conditional rules. These rules can use any one or more of a number of logical relationships to further structure the ontology dataset. Common semantic web entailment rule-sets include: RDF entailment, RDF Schema entailment, D-Entailment, OWL 2 RDF-Based Semantics entailment, OWL 2 Direct Semantics entailment, SHACL rules [i.29] and RIF-Simple entailment: https://www.w3.org/TR/sparql11-entailment/diff.

Inference is often based on the forward chaining of entailment rules defined using RDF triple patterns with variables. A ruleset is a set of axiomatic triples, consistency checks and entailment rules which determine the applied semantics. Thus, most ruleset files have three distinct sections: Prefixes, Axioms, and Rules. In a forward chaining inference step, a rule is interpreted as meaning that for all possible ways of satisfying the premises, the bindings for the variables are used to populate the consequences of the rule. Like the Basic Graph Patterns and for the same reasons, it is difficult to generalize, compose, or modularize these rules. But a well-tuned rule could be worth the development time and testing effort.

Consider the declaration of the SAREF4WEAR *isLocatedIn* ObjectProperty:

```
###  https://saref.etsi.org/saref4wear/isLocatedIn
s4wear:isLocatedIn rdf:type owl:ObjectProperty ;
    rdfs:subPropertyOf s4wear:isLocated ;
    rdfs:domain s4wear:InBodyWearable ;
    rdfs:range s4wear:Wearer ;
    rdfs:comment "A relationship specifying the location of a wearer with respect to an in-body
wearable."@en ;
    rdfs:label "is located in"@en .
```

Here is an example of a file (CustomRule.pie) that uses the OWL-MAX-optimized ruleset in GraphDB v10 to compute the transitive closure of isLocatedIn:

```
Prefices
{
```

```
rdf : http://www.w3.org/1999/02/22-rdf-syntax-ns#
owl : http://www.w3.org/2002/07/owl#
abc : http://www.xyzabc.com/schema/abcentity#
}
Axioms
{
<abc:isLocatedIn> <rdf:type> <owl:ObjectProperty>
}
Rules
{
Id: isLocatedIn
a <abc:isLocatedIn> b [Constraint a != b]
b <abc:isLocatedIn> c [Constraint b != c]
}
```

5.6 SHACL Shapes

The W3C Shape and Constraint Language (SHACL) specification [i.18] defines a language for validating RDF graphs against a set of conditions or constraints similar to SQL table constraints -- SHACL shapes formally define a static ontology pattern. A SHACL "shape graph" is typically used to validate whether RDF data graphs satisfy the set of described constraints. A shape graph can also be used to validate whether the RDF ontology graph conforms to a set of guidelines formatted as SHACL shapes.

Any Class that is bound to this SHACL shape is (statically) type-checked. SHACL shapes can be used in the SAREF verification / validation pipeline. This type-checking works on named as well as unnamed nodes and can be configured to work alongside any defined entailment rules.

The following SHACL shape validates whether a saref:Property that may be defined in an extension conforms to the structure identified in clause 5.2. This is an instance of using SHACL for automatically checking the conformance of an ontology to a set of guidelines.

The following two SHACL shape describe best practices that could be put on the ontology. The first specifies that every element should have a string as rdfs:comment and a language string as rdfs:label. The second follows the format of clause 5.2 by specifying that each instance of saref:Property should explicitly be defined as an owl:NamedIndividual. Additionally, any SPARQL query (e.g. the example from clause 5.2) can be included in a shape:

```
ex:SarefGuidelines
    a sh:NodeShape ;
    sh:property [
        sh:path rdfs:comment ; # every class should have a comment
        sh:minCount 1 ;
        sh:datatype xsd:string ;
    ] ;
    sh:property [ # _:b2
        sh:path rdfs:label ;
        # every class should have label in a particular language
        sh:minCount 1 ;
        sh:datatype xsd:langString ;
    ] ;
.
ex:PropertyGuidelines
    a sh:NodeShape ;
    sh:targetClass saref:Property ;
    sh:property [
    # a property should also be defined as named individual
      sh:path rdf:type ;
      sh:hasValue owl:NamedIndividual
.
```

The following SHACL shape describes that a SAREFised measurement should contain at least one saref:Property and exactly one unit of measure:

```
ex:MeasurementShape
  a sh:NodeShape ;
  sh:targetClass saref:Measurement ;
  sh:property [
    sh:path saref:relatesToProperty ;
    sh:minCount 1 ;
    sh:class saref:Property ;
  ] ,
  [ sh:path saref:isMeasuredIn ;
    sh:minCount 1;
```

```
    sh:maxCount 1;
    sh:class saref:UnitOfMeasure ;
  ]
```

SHACL Playgrounds and validation tools can be found online. Defining SHACL shapes alongside OWL axioms for SAREF and its extensions would be an interesting future direction of work.

SHACL Playgrounds https://shacl.org/playground/ and https://shacl-playground.zazuko.com/. SHACL validation tool https://shacl-play.sparna.fr/play/.

5.7 Chowlk visual notation templates

The Chowlk visual notation [i.19] provides a visual syntax for the conceptualization of OWL ontologies based on UML notation and is based on the NeOn UML onto profile. The notation specifies a set of graphical blocks that represent elements from the OWL language. Additionally, as part of the whole framework [i.20], Chowlk provides an online converter [i.21], that is a system that takes the conceptualizations elaborated with an online diagram creation tool following Chowlk notation, and generates the corresponding OWL implementation. The Chowlk framework also provides two libraries to help users adopting the notation. This feature of generating templates can be used to generate SAREF ontology design patterns to facilitate, not only the reuse of SAREF, but also the extension of some parts of the SAREF ontologies when developing new extension or generating data examples.

5.8 Dead Simple OWL Design Patterns (DOS-DP)

The Dead Simple OWL Design Patterns (DOS-DP) framework [i.22] and [i.23] uses Class management techniques to modify and automatically update large bio-ontologies.

DOS-DP rightly points out that the best alternative to manually asserting classification is to use OWL inference to automate it, in which OWL equivalence axioms can be used to specify necessary and sufficient conditions for class membership. In this regard, their approach is similar to the Tawny OWL language [i.24] and the Ontology Pre-Processing Language [i.25], the other major mechanisms for specifying design patterns by programmatic use. However, DOS-DP by design lacks a mechanism for relating patterns to each other via inheritance or composition. It also lacks a system for specifying the optional clauses necessary for a subsumption hierarchy. Since manual maintenance of design pattern hierarchies' risks re-creating the maintenance problem that these patterns are meant to solve, the DOS-DP approach is best used for the management of existing large ontologies such as the Gene Ontology.

The authors mention *TermGenie*, a well-used web-application for pattern-based ontology class generation. TermGenie allows biocurators to generate new classes based on formally specified design patterns or templates. Automated rules and reasoning engines are used to ensure validity, uniqueness and relationship to pre-existing classes. All classes added through pre-defined templates are guaranteed to have OWL equivalence axioms that are used for automatic classification. Automated rules and reasoning engines are used to ensure validity, uniqueness and relationship to pre-existing classes.

Below is a DOS-DP template which may be used to describe the properties of the SAREF core class 'Command':

```
{
  "classes": {
    "Command pattern": null
  },
  "def": {
    "text": " A directive that a device has to support to perform a certain function. A command may
act upon a state but does not necessarily act upon a state. For example, the ON command acts upon
the ON/OFF state, but the GET command does not act upon any state, it simply gives a directive to
retrieve a certain value. We propose here a list of commands that are relevant for the purpose of
SAREF, but this list can be extended. ",
    "vars": [
      "actsUpon",
      "isCommandOf"
    ]
  },
  "equivalentTo": {
  "text": "",
  "vars": [
    "actsUpon",
    "isCommandOf"
  ]
  },
  "name": {
```

```
  "text": "%s1 actsUpon, %s2 isCommandOf",
  "vars": [
    "actsUpon",
    "isCommandOf"
  ]
  },
  "pattern_name": "Command",
  "relations": {
  },
  "vars": {
    "actsUpon": "State",
    "isCommandOf": "Function"
  }
}
```

5.9 Reasonable Ontology Templates (OTTR)

The Reasonable Ontology Templates (OTTR) library framework [i.26] and [i.27] address the ontology modelling issues of using parameterized templates to construct, interact with, and maintain ontology patterns; formulating abstractions in a Description Logic; promoting tool-supported best practices; and bridging the gap between the natural language of domain facts and their representation in OWL. A library http://tpl.ottr.xyz/ registers 196 well-designed OTTR templates available that are designed to be reused, are good for organizing abstractions and use-cases, managing abstraction granularity and conceptual clarity. This approach clearly helps to avoid design conflicts and modelling discrepancies, though less so for existing ontologies. OTTR integrates with existing toolchains (except for Protégé!) and provides good process support. The central output artifact that the framework manages is an OTTR template document, which can then be serialized into multiple output formats.

OTTR detects two types of redundancy:

i) a lack of reuse of existing templates; and

ii) recurring patterns not captured by templates within the library.

Thus, OTTR automatically detects and identifies ontology similarities Templates can be used directly as queries and are serialized into multiple output formats including SPARQL SELECT, CONSTRUCT, and UPDATE. The macro expander mechanism provides semantic consistency checks. OTTR is described and implemented formally, yet still integrates with many online tools. The framework provides model, data, and query tests by comparing the various serialized outputs to verify ontology consistency.

The OTTR framework would most likely consolidate and harmonize the SAREF ontologies, but it would take a lot of work because the existing ontologies have to first be transformed into wOTTR templates, a special-purpose OWL/RDF vocabulary. It is also difficult to predict how much of SAREF could be transformed and aligned in this manner. The templates would then be run through https://weblutra.ottr.xyz/ and the redundancies resolved. Naming conventions and a standardized vocabulary would still need more work.

Below is an OTTR example that declares a *SoilTemperature* named individual from the SAREF core class Measurement:

```
@prefix ottr:      <http://ns.ottr.xyz/0.4/>.
@prefix o-docttr:  <http://tpl.ottr.xyz/p/docttr/0.1/>.
@prefix o-owl-ax:  <http://tpl.ottr.xyz/owl/axiom/0.1/>.
@prefix o-owl-re:  <http://tpl.ottr.xyz/owl/restriction/0.1/>.
@prefix o-owl-ut:  <http://tpl.ottr.xyz/owl/util/0.1/>.
@prefix o-rdf:     <http://tpl.ottr.xyz/rdf/0.1/>.
@prefix o-rdfs:    <http://tpl.ottr.xyz/rdfs/0.1/>.
@prefix owl:       <http://www.w3.org/2002/07/owl#> .
@prefix rdf:       <http://www.w3.org/1999/02/22-rdf-syntax-ns#>.
@prefix rdfs:      <http://www.w3.org/2000/01/rdf-schema#>.
@prefix skos:      <http://www.w3.org/2004/02/skos/core#>.
@prefix xsd:       <http://www.w3.org/2001/XMLSchema#>.
@prefix saref:     <https://saref.etsi.org/core/>.
@prefix temp:      <https://saref.etsi.org/examples/SoilTemperature#>.
@prefix s4agri:    <https://saref.etsi.org/saref4agri/>.

<<https://saref.etsi.org/ottr/core/Temperature>[
    LUB<ottr:IRI> ?blank512,
    NEList<xsd:string> ?blank513,
    NEList<xsd:string> ?blank514,
    xsd:string ?blank515,
    NEList<xsd:string> ?blank516
```

```
]
@@o-docttr:Version(
    <https://saref.etsi.org/core/v3.1.1/Temperature>,
    ottr:draft,
    "3.1.1",
    saref:Annotation,
    none)
:: {
    ottr:Triple(?blank512, skos:prefLabel, ?blank515),
    cross | ottr:Triple(?blank512, skos:altLabel, ++?blank514),
    cross | ottr:Triple(?blank512, rdfs:comment, ++?blank516),
    cross | ottr:Triple(?blank512, rdfs:label, ++?blank513)
} .

<https://saref.etsi.org/saref4agri/SoilTemperature>[
    owl:Class ?blank508,
    ? owl:NamedIndividual ?blank509,
    NEList<owl:Class> ?blank510
]
@@o-docttr:Version(
    <https://saref.etsi.org/saref4agri/SoilTemperature>,
    ottr:draft,
    "0.2.2",
    saref4agri:SoilTemperature,
    none)
:: {
    o-owl-ax:SubClassOf(?blank508, temp:Temperature),
    cross | o-owl-ax:SubObjectSomeValuesFrom(?blank508, temp:hasMeasurement, ++?blank510),
    o-owl-re:ObjectUnionOf(_:blank511, ?blank510)
} .
```

5.10 The Modular Ontology Modeling (MOMo) methodology

The Modular Ontology Modeling (MOMo) methodology [i.28] is built over the CoMoIDE Protégé plugin [i.29]. It addresses the ontology modelling issues of differing levels of abstraction/detail representational granularity, the lack of conceptual clarity, poor modelling principles, and poor process support.

The framework is simple to understand and use by a domain expert for ontology construction. It is not as useful for deconstruction because loading an existing (SAREF) ontology displays a graphical mess that is impossible to analyse. Their use of modules organizes the development process as a divide-and-conquer approach for both modelling and use-cases. These well-designed modules are often reused since modules wrap other modules, but the number of modules multiplies quickly. The authors admit this process takes time for people to agree on how to best connect modules together. The development process is well-supported as long as the domain, data, and ontology experts are gathered together. The framework creates and manages schema diagrams to produce OWL axiom files as both the central artifact and the final output.

The CoMoIDE library provides a set of 17 built-in ontology patterns. It is a Protégé plugin, and the MOMo workflow is both structured and intuitive. However, someone on the team has to know OWL very well since functionality restrictions, domain-range axioms, and logical connectives are not indicated. Nor does the framework provide model, data, or query tests. There is also no automatic detection or identification of existing ontology similarities or redundancies, no formulation of abstractions in Description Logic, and no detection or reconciliation of bad instance data.

Migrating the existing SAREF ontologies to the 17 CoMoIDE/MOMo patterns would provide limited benefit given the amount of work needed.

Consider the SAREF4CITY declaration of an Agent:

```
S4CITY:Agent a owl:Class ;
  rdfs:label "Agent"@en ;
  rdfs:comment "An agent making an action in the context of a city. An agent could be a person,
software, etc."@en ;
  rdfs:subClassOf <http://xmlns.com/foaf/0.1/Agent> ,
  [ a owl:Restriction ;
    owl:onProperty cpsv:provides ;
    owl:allValuesFrom s4city:PublicService
  ] ,
  [ a owl:Restriction ;
    owl:onProperty cpsv:uses ;
    owl:allValuesFrom s4city:PublicService
  ] .
```

The corresponding CoMoIDE pattern uses different assumptions (e.g. TemporalExtent) from the SAREF Agent class.

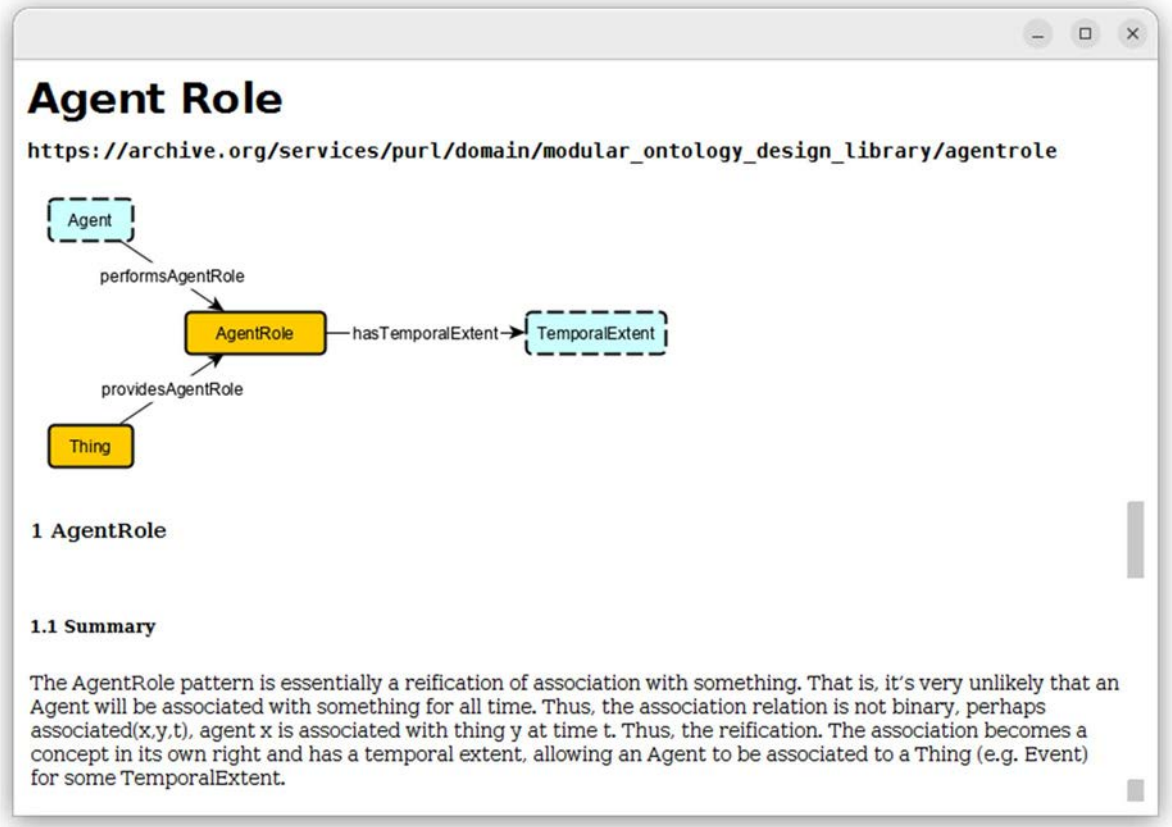

Figure 1: The Agent Role pattern taken from the CoMoIDE Protégé plugin

6 Analysis of the modularization and factorization potential of SAREF ontologies

6.1 SAREF Core

This clause provides an updated analysis of the potential of modularization and factorization of the SAREF core ontology V3.1.1 [i.4]. It updates the analysis in ETSI TR 103 549 [i.2], clause 7, and provides additional analysis for the other SAREF ontologies. It highlights inter-dependent parts of the ontology, parts that are more or less *central*, and parts that are repeated homogeneously for different concepts (patterns). The result of this analysis is illustrated on Figure 2.

In Figure 2, a box illustrates a module constituted by a subset of concept declarations and axioms of SAREF core V3.1.1 [i.4]. Each module has a label in bold font, and the lower part of the box lists the concept declarations that belong to this module. Axioms are not shown in Figure 2. For example, the box labelled **Service-core** contains the concept declarations and axioms related to the terms `saref:Service`, `saref:isOfferedBy`, `saref:offers`, `saref:represents`. The terms `saref:Function` and `saref:hasFunction` will also be grouped in one single module because they share a similar name.

A directed link between two boxes illustrates a dependency between the two modules. The label of a link explicates the rationale, and potentially the condition, for this dependency. For example, the module **Service-core** depends on the module **Functions and Commands** because there exists an axiom in SAREF stating that every `saref:Service` represents some `saref:Function`. Therefore, `saref:Service` cannot be defined without the `saref:Function`. In general, restrictions such as existential cardinality restrictions and minimal cardinality restrictions are used to decide on the direction of a dependency.

The grouping of concept declaration and axioms of SAREF in modules and the orientation of the dependencies between modules is partly made by choosing to view SAREF according to a certain perspective, and partly for intuitive reasons. For example, it makes sense to consider that `saref:Property` can be defined independently of `saref:Measurement`, but not the other way around. Therefore, the dependency link will be oriented from **Measurement-core** to **Property-core**.

It is necessary to group the concepts and axioms related to functions and commands into one single module **Functions and Commands**, because there exists axioms in SAREF stating that every `saref:Function` is associated to at least one `saref:Command`, and vice versa.

This analysis can be used to modularize SAREF core V3.1.1: modules or group of modules with no incoming dependencies are not required for other modules. They could be safely filtered out in the documentation on the portal or in some embedded implementation of SAREF, without impacting the rest of the documentation or application. For example, the two modules **Service-core** and **specific Service** have no incoming dependency, therefore they are not essential to the specification of the rest of the ontology.

In Figure 2, a box with an underlined label represents a pattern which can be applied to different concepts. For example, the pattern **specific Sensor** can be instantiated for `saref:SmokeSensor` and `saref:TemperatureSensor`. The pattern **specific Function** can be instantiated for `saref:ActuatingFunction`, `saref:OpenCloseFunction`, etc.

These instantiated patterns have dependency links with other modules, and potentially other patterns. The latter are to be understood as dependencies between two specific pattern instances. For example, the instance of pattern **specific Function** for `saref:OpenCloseFunction` has a dependency to instance of pattern **specific Command** for `saref:OpenCommand` and `saref:CloseCommand`, because they are the types of commands this function can have.

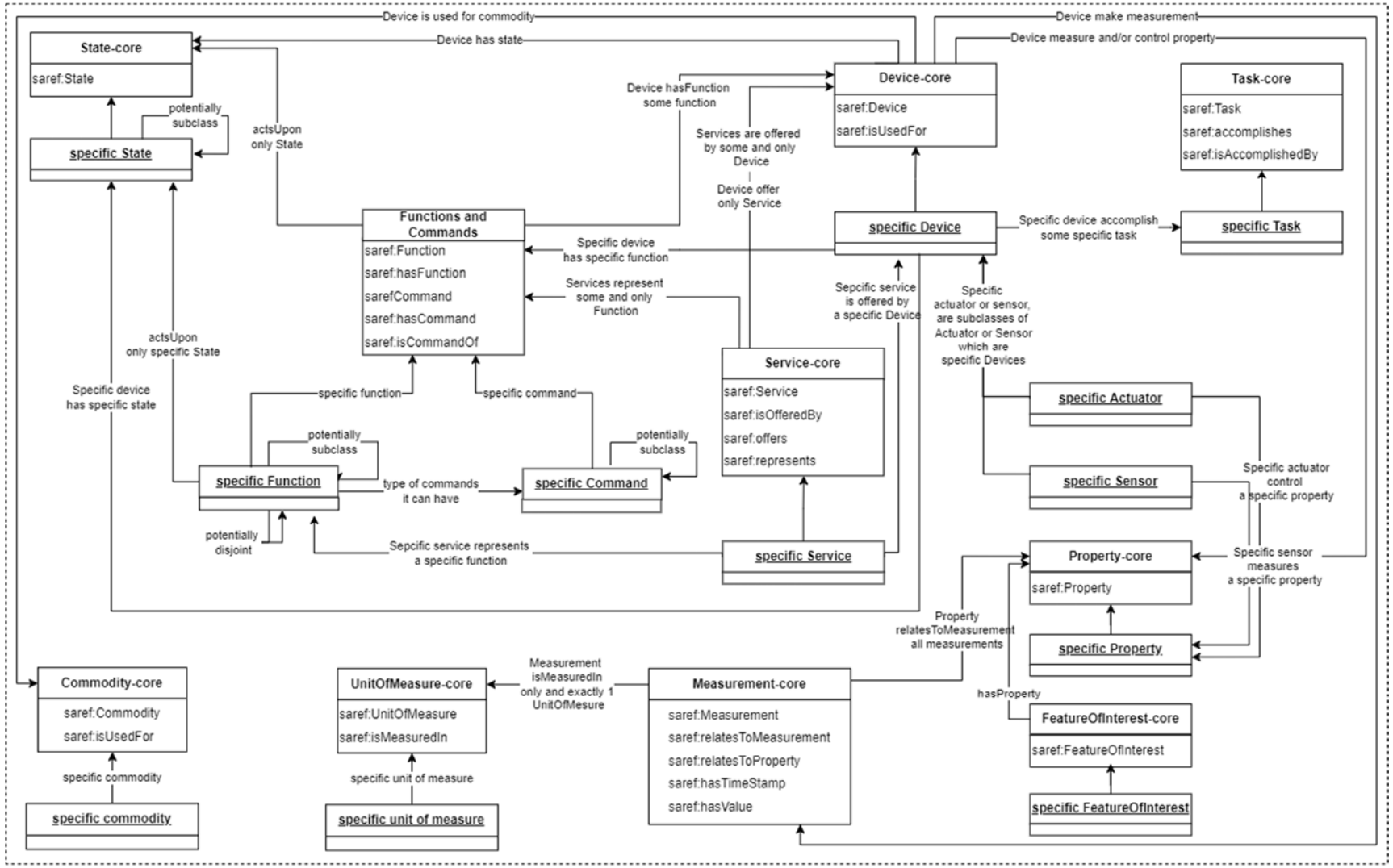

Figure 2: Analysis of the modularization and factorization potential of SAREF-Core (V3.1.1)

6.2 Predication-based Semantic Indexing

The goal of predication-based semantic indexing (PSI) is to assist in the modularisation and factorisation of the SAREF extensions by discovering existing reference ontology patterns. PSI starts with a collection of known facts or observations and combines them into a single semantic vector model in which both concepts and relationships are represented. Instead of manually comparing each of the Class, ObjectProperty, and NamedIndividual entities in the SAREF extensions, a PSI offered by a commercial graph database was constructed for the entire set of extensions compiled into the single file *merged-ontology.ttl* and loaded into the graph database. Once the similarity index was built, an automated cURL script compared each and every entity to each other and output a similarity score that ranged from 0 to 1,0. All scores ≥ 0,8 were collected, which retains only the most similar nodes. The similarity tests were run for all SAREF Classes (shown below), *NamedIndividuals* (SAREF4WEAR: 1 large cluster and 2 tiny ones; SAREF4WATR: 1 huge cluster! SAREF4ENVI: 1 large cluster; SAREF4EHAW: 1 large cluster; SAREF4AUTO: 1 large cluster and 1 small one; SAREF4AGRI: 1 large cluster), and *ObjectProperties* (with only 5 cluster pairs). These similar graph patterns are candidates as SAREF ontology reference patterns. The connectivity of each similarity cluster is quantified by the *node degree* of the classes in the cluster – a count of how many other classes are similar to a particular class. The similarity clusters are also plotted as Cytoscape graphs where a line between two nodes indicates a similarity score greater than 0.8 from one to the other. Two lines between the same two nodes indicates a structural match.

NOTE: The similarity index does not compare to non-SAREF ontologies; hence, every one of the 20 SAREF4CITY classes comes up empty in the predicate similarity index because they derive from a non-SAREF ontology or from `s4city:AdministrativeArea`, which is a subClassOf `geosp:Feature`. NamedIndividual similarities > 0,9 exist only within the SAREF4AUTO extension. SAREF4WATR contains a huge cluster for all the substances that poison water.

As an example, this is a list of the node degrees for the three class clusters within the SAREF4LIFT extension and between SAREF4LIFT and all other SAREF Extensions where the graphs give the extension as a prefix, say 'auto/'.

The SAREF4LIFT Statistics are all subClassOf `s4lift:StatisticsMeasurement`:

Within SAREF4LIFT extension:	Node Degree
UpwardTravelsStatistics	6
NumberOfResetSequencesStatistics	5
NumberOfCallsStatistics	5
TotalNumberOfOpeningOfDoorStatistics	5
NumberOfFaults	4
TotalReversalDirectionStatistics	3
DownwardTravelsStatistics	3
TotalFloorsCoveredStatistics	3
Between SAREF4LIFT and the other extensions:	
ElectricalDevice	6
VerticalSmartLiftingPlatform	4
MachineRoomLessSmartLift	4
SmartLiftWithoutEmergencyCallSupport	4
AccessibleGoodsOnlySmartLift	3
PassengerAndGoodsPassengerSmartLift	3
FirefightersSmartLift	3
GoodsSmartLift	3
SmartLiftEdgeControlUnit	3

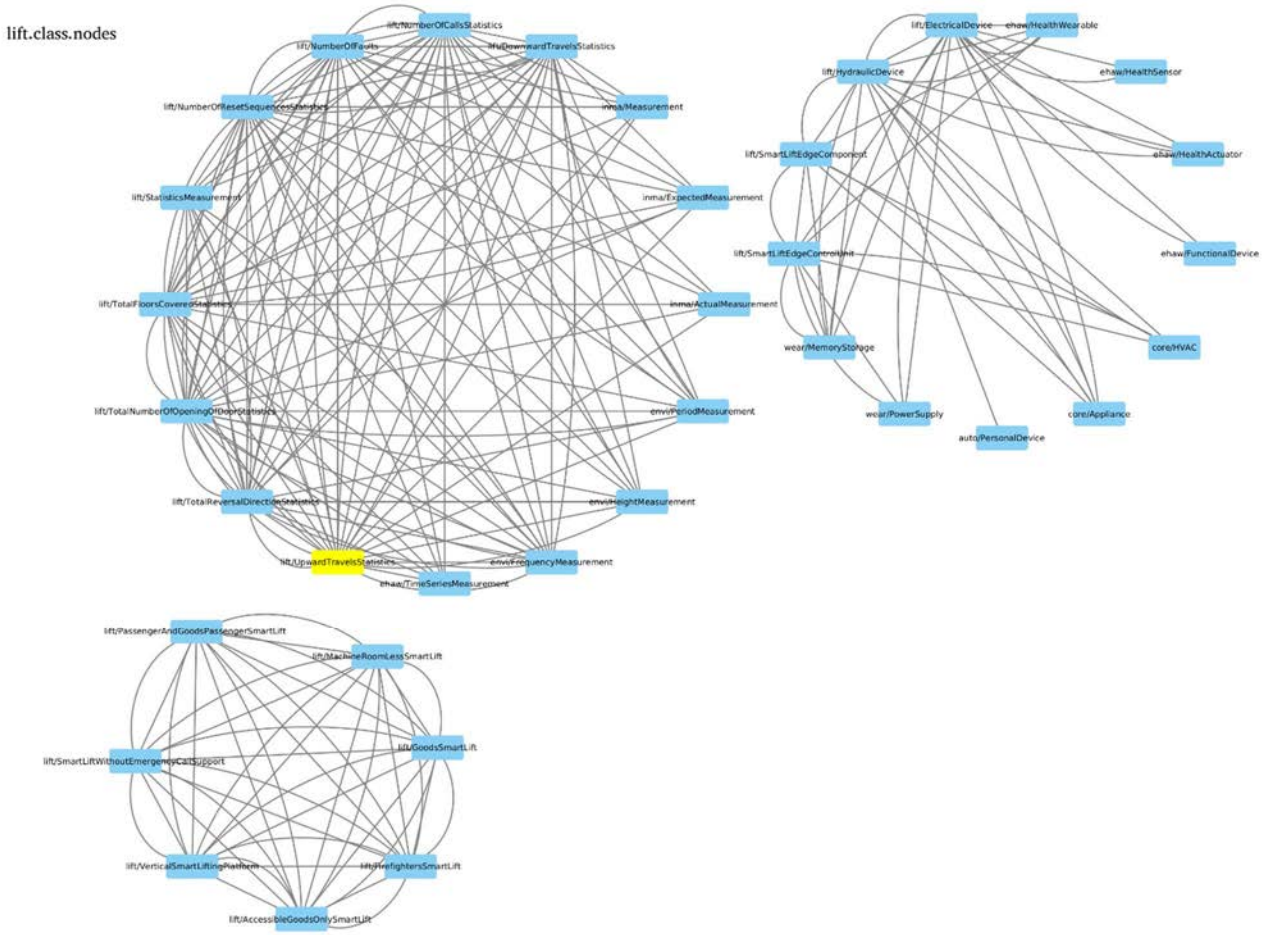

Figure 3: SAREF4LIFT similar class clusters

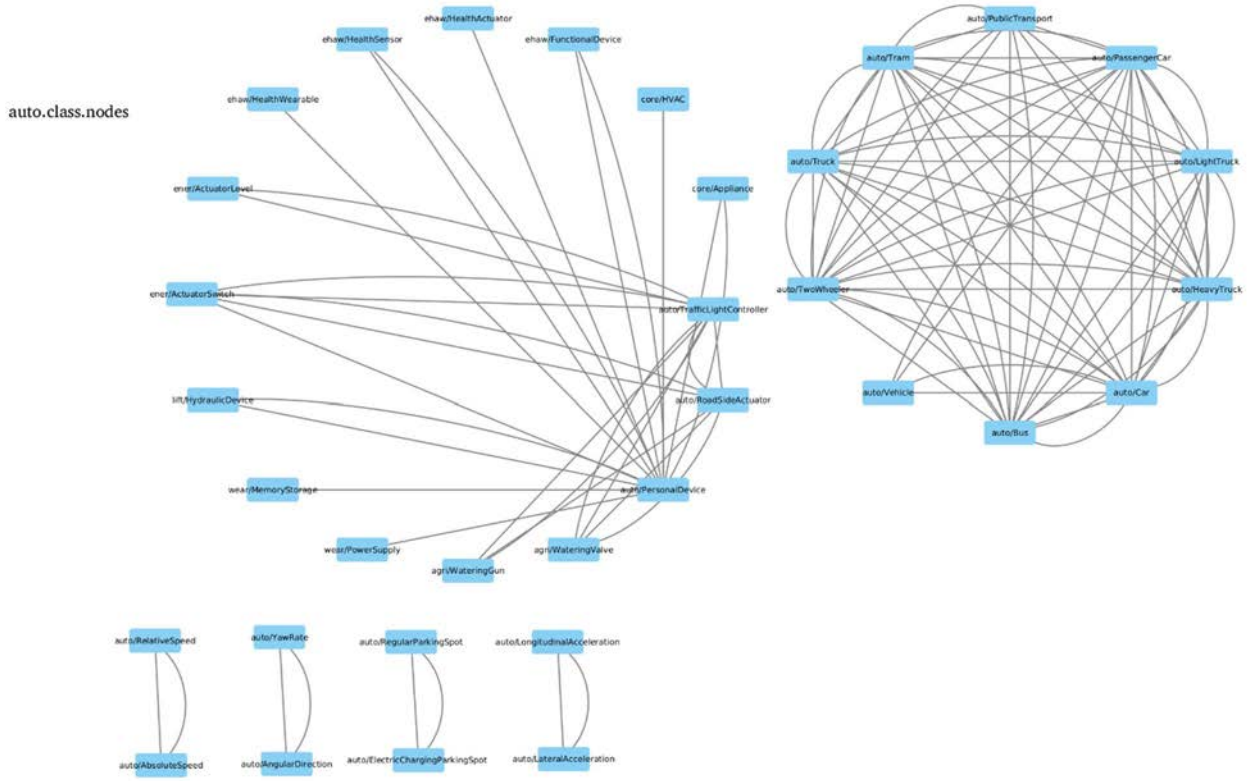

Figure 4: SAREF4AUTO similar class clusters

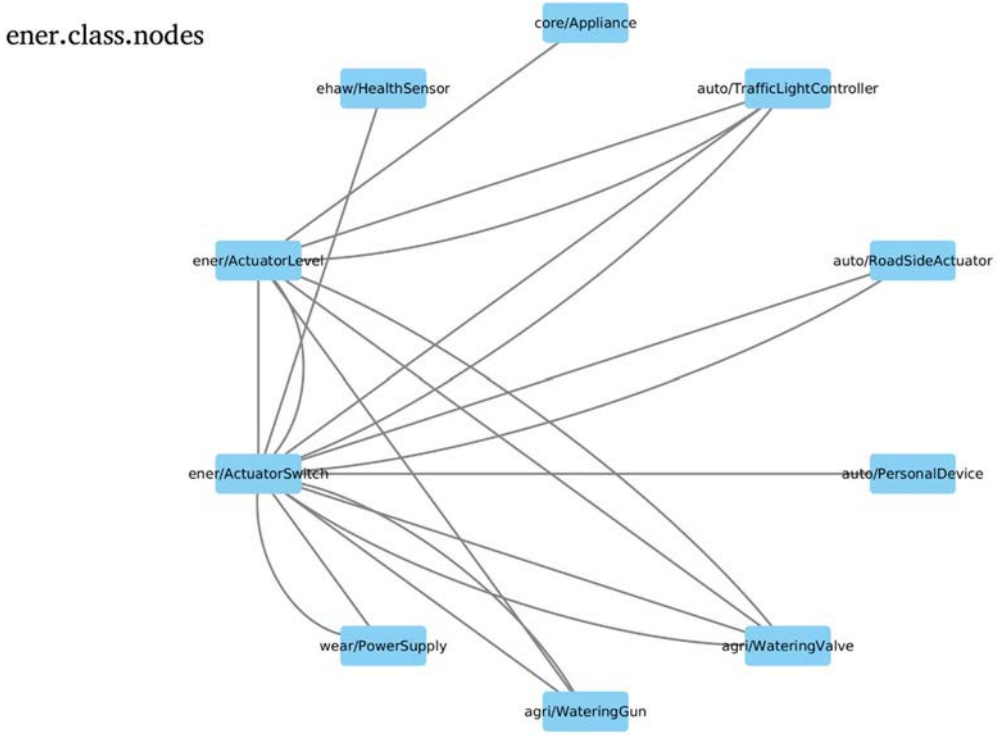

Figure 5: SAREF4ENER similar class clusters

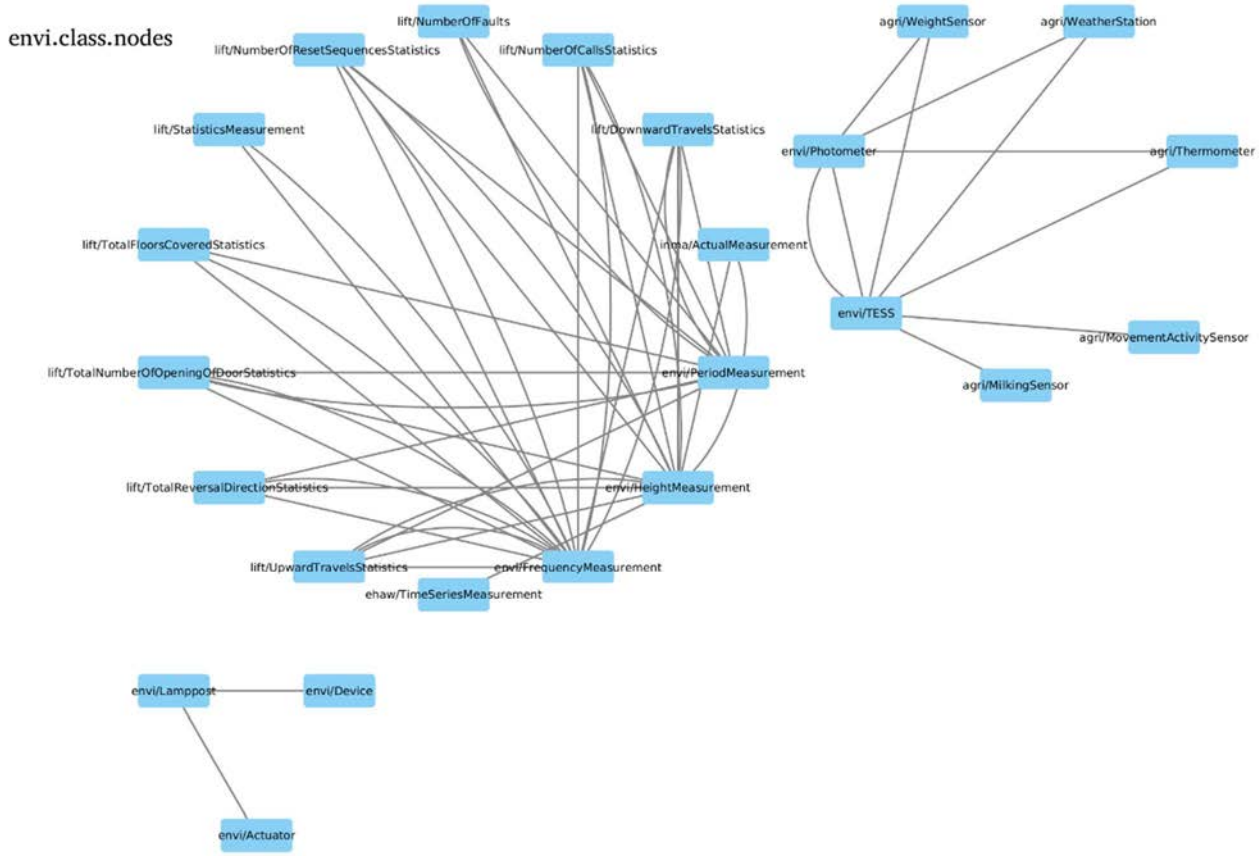

Figure 6: SAREF4ENVI similar class clusters

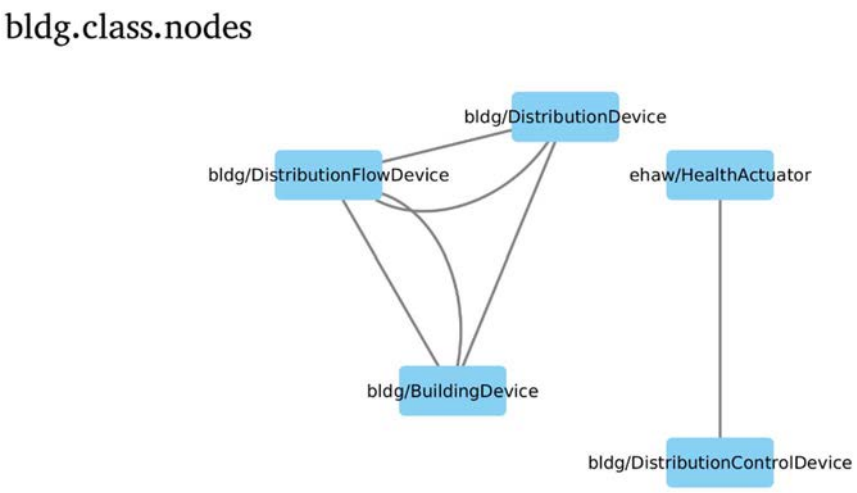

Figure 7: SAREF4BLDG similar class clusters

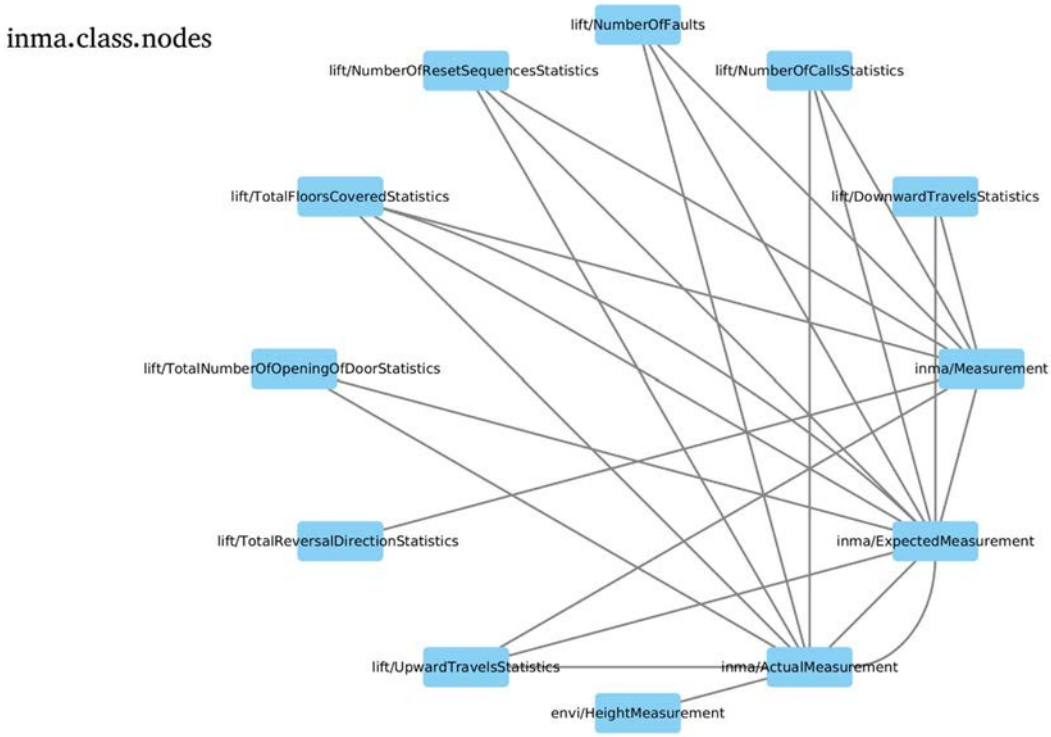

Figure 8: SAREF4INMA similar class clusters

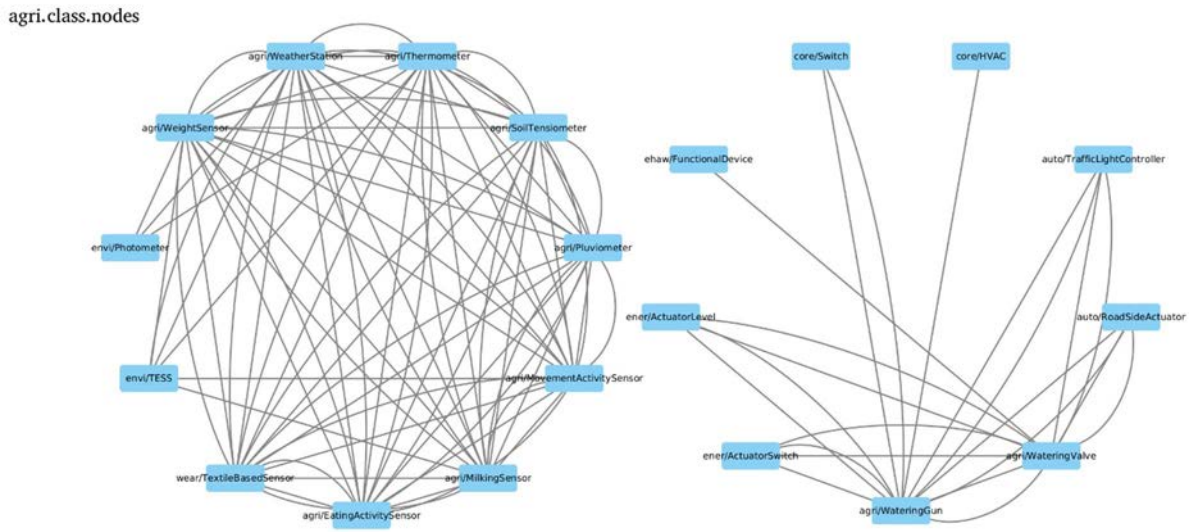

Figure 9: SAREF4AGRI similar class clusters

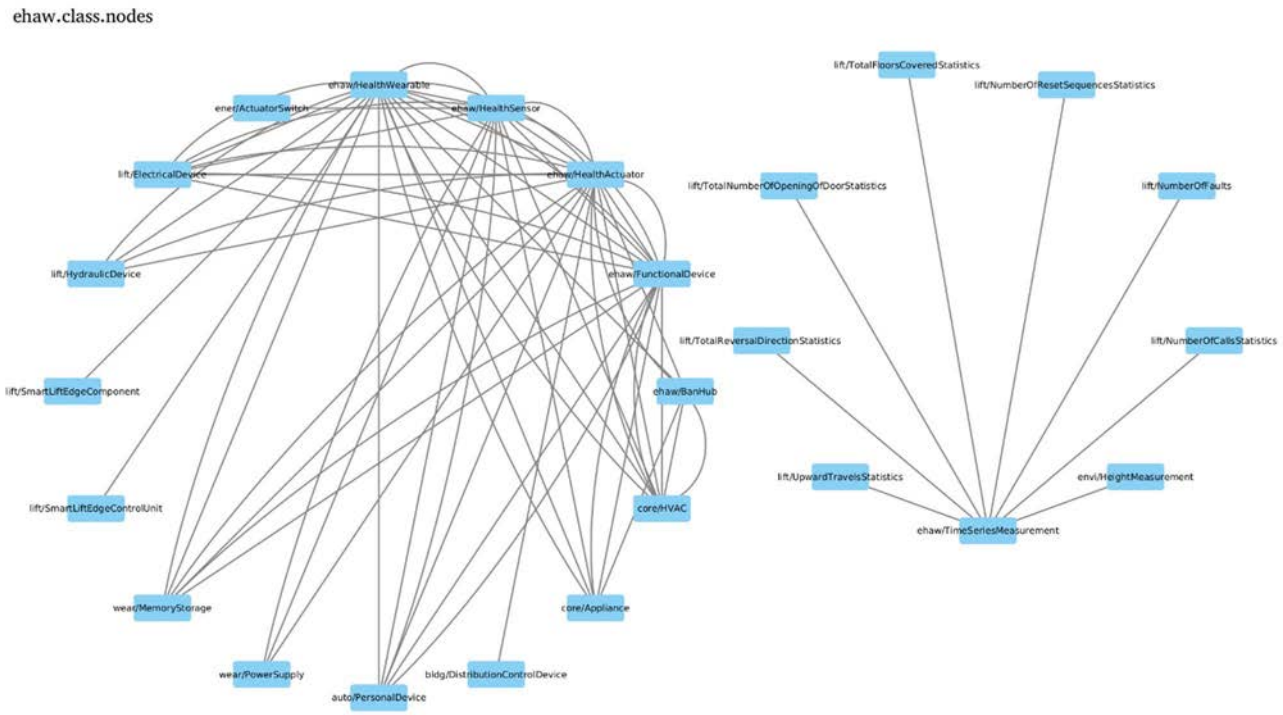

Figure 10: SAREF4EHAW similar class clusters

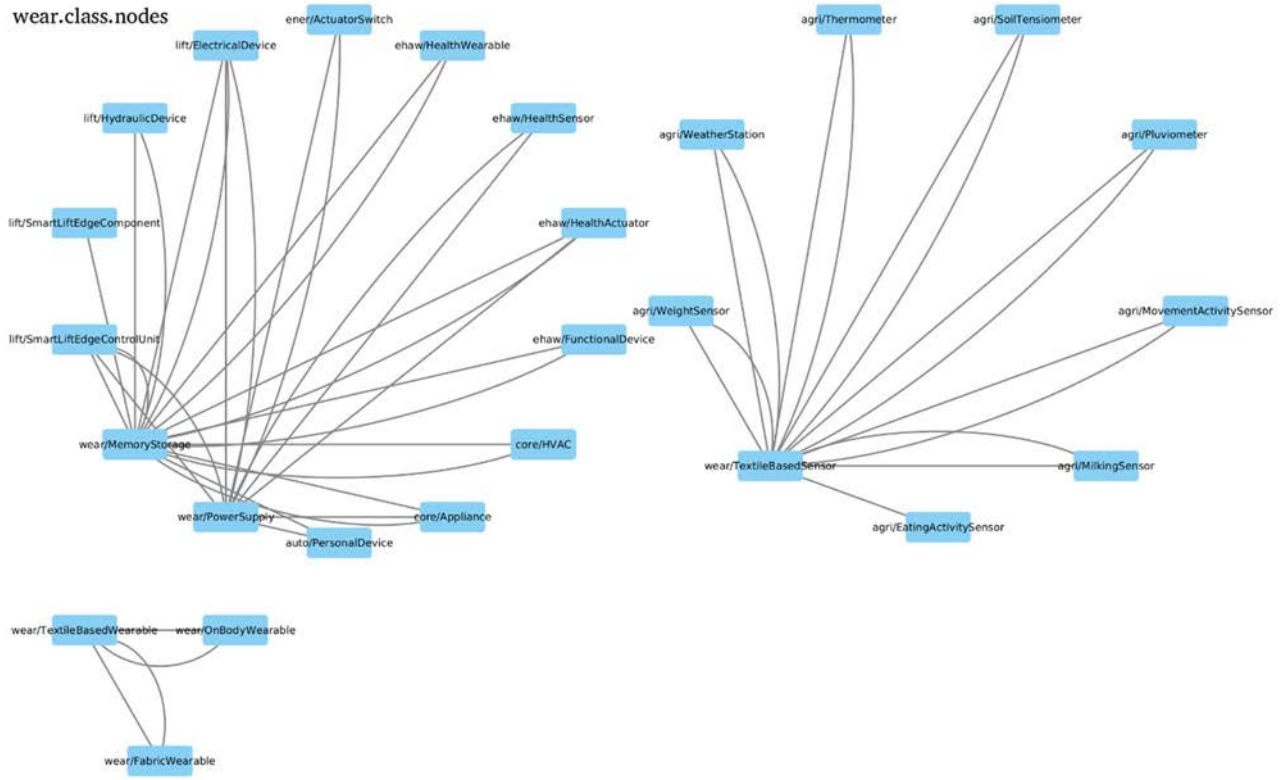

Figure 11: SAREF4WEAR similar class clusters

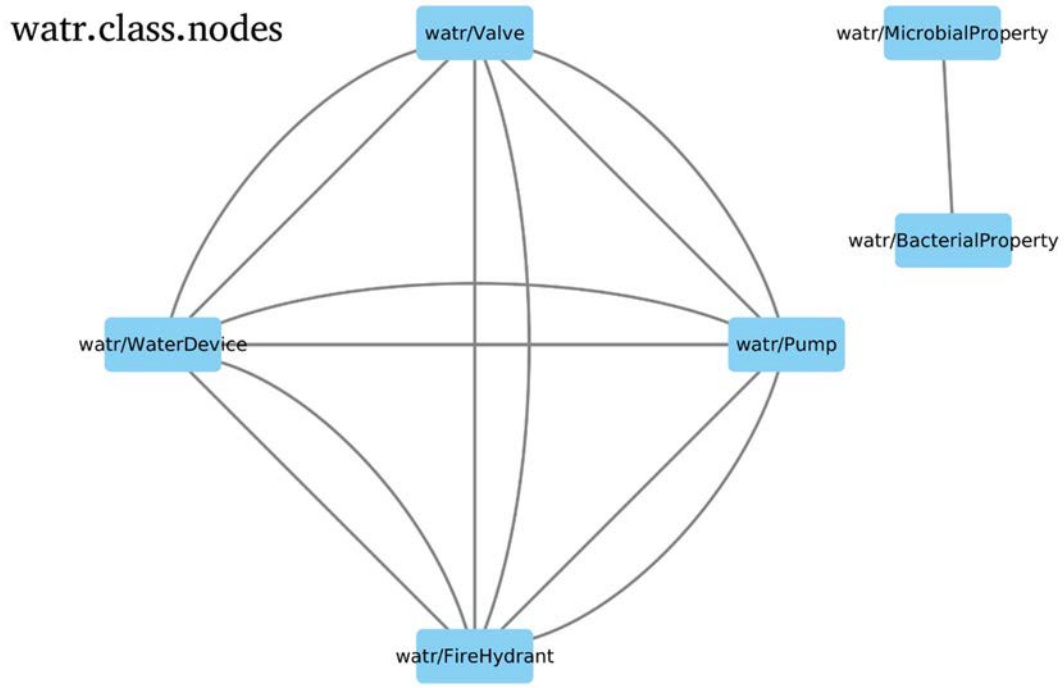

Figure 12: SAREF4WATR similar class clusters

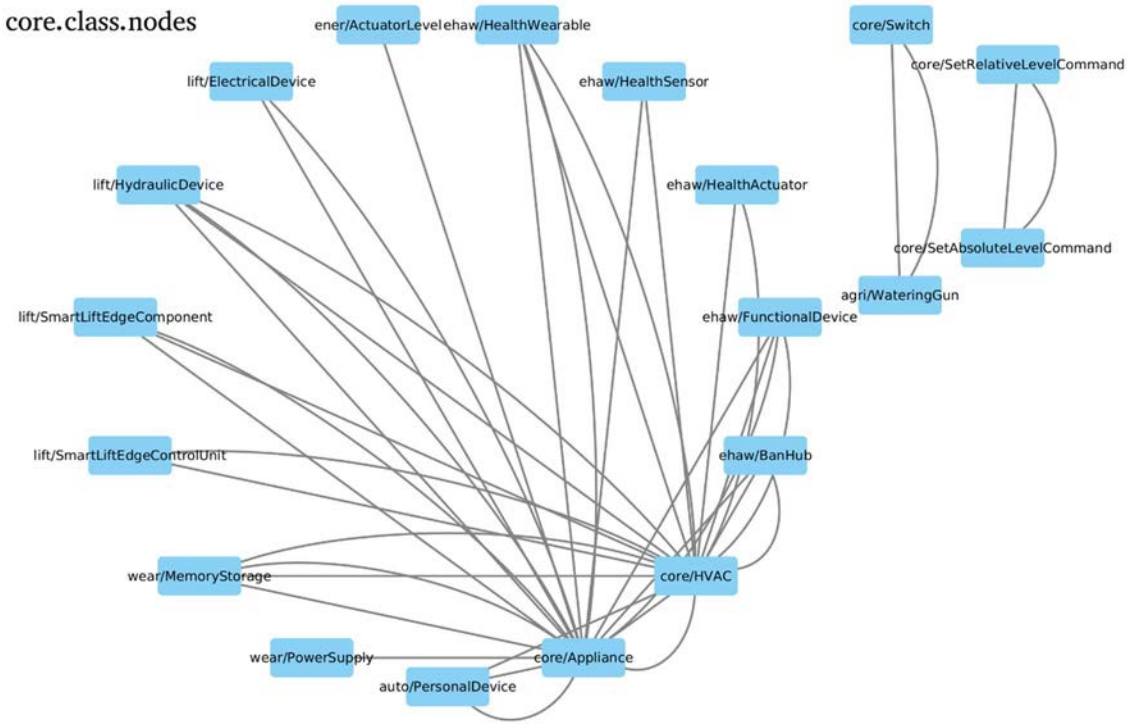

Figure 13: SAREF Core similar class clusters

6.3 Similarity Analysis Discussion

Simple classes by definition are more likely to be similar to other classes that are just as simple, and classes that are used more frequently are more likely to have higher similarity scores. Just because a similarity index reveals a class cluster does not necessarily imply the cluster is semantically useful, but it does warrant further investigation of the candidate.

The similarity index uses a vector of owl verbs to calculate a class's similarity score. Here is the frequency count of the RDF/OWL verbs in vector form collected from the *merged-ontology.ttl* file, where a certain percent are repeats. The vector gives some idea of what goes into calculating the score.

```
rdf:type                        2104
rdfs:label                      1826
rdfs:comment                    1741
rdfs:subClassOf                 1014
owl:onProperty                  878
owl:Restriction                 875
owl:Class                       779
owl:ObjectProperty              583
owl:allValuesFrom               552
owl:DatatypeProperty            296
owl:NamedIndividual             213
owl:someValuesFrom              171
owl:AnnotationProperty          156
owl:inverseOf                   48
owl:maxCardinality              33
owl:disjointWith                28
owl:cardinality                 16
owl:Ontology                    13
owl:versionInfo                 13
owl:oneOf                       12
owl:FunctionalProperty          10
owl:TransitiveProperty          8
owl:minCardinality              6
owl:Thing                       6
owl:unionOf                     6
owl:imports                     4
owl:intersectionOf              4
owl:hasValue                    3
owl:InverseFunctionalProperty   2
owl:SymmetricProperty           2
owl:equivalentClass             1
```

A **subClassOf** analysis of the similarity clusters displayed in Figures 3 to 13 is given in table 2.

Table 2

SAREF4LIFT	Upper-left cluster: all lift are subClassOf `s4lift:StatisticsMeasurement` Lower-left cluster: all lift are subClassOf `s4lift:SmartLiftInstallation` Upper-right cluster: all are subClassOf `saref:Device`
SAREF4AUTO	Upper-left cluster: subClassOf saref:Device Upper-right cluster: subClassOf s4auto:Vehicle
SAREF4ENER	Single cluster: subClassOf saref:Device
SAREF4ENVI	Upper-left cluster: subClassOf saref:Measurement Upper-right cluster: subClassOf saref:Sensor Lower-left cluster: subClassOf s4envi:Device with additional Restrictions.
SAREF4BLDG	Left cluster: subClassOf s4bldg:BuildingDevice
SAREF4INMA	Single cluster: subClassOf core:Measurement disjoint with s4inma:ExpectedMeasurement.
SAREF4AGRI	Left cluster: subClassOf saref:Sensor Right cluster: subClassOf saref:Device
SAREF4EHAW	Left cluster: subClassOf saref:Device Right cluster: subClassOf saref:Measurement with additional Restrictions.
SAREF4WEAR	Upper-left cluster: subClassOf saref:Device Upper-right cluster: subClassOf saref:Sensor Lower-left cluster: subClassOf s4wear:TextileBasedWearable
SAREF4WATR	Left cluster: subClassOf s4watr:WaterDevice
SAREF4CORE	Left cluster: subClassOf saref:Device

Some general conclusions from the similarity analysis are:

1) The large, circular clusters with many cross-over lines (i.e. complete or near-complete bipartite graphs) such as SAREF4LIFT upper- and lower-left, SAREF4AUTO upper-right, SAREF4AGRI left, and SAREF4WATR left are candidates for a new parent class or restriction that distinguishes them from other classes.

2) The multi-parent but mostly hierarchical clusters such as SAREFCORE left, SAREF4WEAR upper-left and right, both SAREF4EAHW clusters, SAREF4AGRI right, SAREF4INMA, SAREF4ENVI, SAREF4ENER upper-right, and SAREF4AUTO upper-left are candidates for a new parent class or restriction with some risk.

3) Consider the sets of two or three nodes connected by double-lines as transitive or inverse class pairs.

4) The specific modelling choices for Classes are straight-forward and do not appear to contravene each other.

7 In-depth analysis of SAREF extensions

7.1 Analysis of SAREF4ENER V1.1.2

At the time of writing the present document SAREF4ENER V1.2.1 is under development, so no report on SAREF4ENER V1.1.2 [i.5] is provided.

7.2 Analysis of SAREF4ENVI V1.1.2

SAREF4ENVI V1.1.2 [i.6] copies the definition of several entities from SAREF Core. The evolution of SAREF Core to V3.1.1 made these definitions obsolete. This demonstrates this practice is difficult to maintain and should be prohibited:

- OP saref:isMeasuredIn, saref:makesMeasurement, saref:measuresProperty, saref:relatesToProperty.
- DP saref:hasDescription, saref:hasManufacturer, saref:hasTimestamp, saref:hasValue.
- Classes saref:Measurement, saref:Property, saref:Sensor, saref:Service, saref:UnitOfMeasure.

SAREF4ENVI introduces entities that aim to link SAREF Core entities, or are so generic they could be promoted to SAREF Core or replaced by entities from SAREF Core or other vocabularies:

- OP `s4envi:affectsProperty` links an actuator and the properties it can affect, but the domain and range is not defined. This property may be deleted and replaced by `saref:controlsProperty`, introduced in V3.1.1.
- OP `s4envi:contains` link a physical object and the physical objects that *can* be contained in it. The "can" in this definition makes it ambiguous; The domain and range is not defined; Physical objects are relevant for other extensions and could be promoted to SAREF Core, together with this property.
- OP `s4envi:encapsulates` with inverse `s4envi:hasDigitalRepresentation` may be useful in the context of the work on SAREF and Digital Twins. These may be useful for other extensions, and could be promoted to SAREF Core.
- DP `s4envi:hasCreationDate` is functional, it could be promoted to SAREF Core, or replaced with dct:created or some other existing property.
- DP `s4envi:hasRevisionNumber` defines the revision number of a certain entity (e.g. a device). It could be promoted to SAREF Core.
- DP `s4envi:hasVersion` defines the version of a certain entity (e.g. a device). It could be promoted to SAREF Core.
- Class `s4envi:PhysicalObject`, aligned with the DUL top-level ontology.

SAREF4ENVI defines classes with the same local name as SAREF Core classes, but with alternative definitions. This may be confusing for the end users and should be avoided:

- `s4envi:Actuator` is unrelated to `saref:Actuator`. It is defined as a sub-class of `s4envi:Device` with a universal restriction on s4envi:affectProperty to link to `saref:Property`.
- `s4envi:Device` is a subclass of `saref:Device`, with additional restrictions for the specific use cases addressed in SAREF4ENVI:
  - has a frequency measurement which is of type `s4envi:FrequencyMeasurement`.
  - has a transmission period which is of type `s4envi:TransmissionPeriod`.

SAREF4ENVI could use SAREF4SYST [i.1] patterns to describe the following concepts:

- `s4envi:hasComponent` links a system and the system in which it *might* be decomposed. The "might" in this definition makes it ambiguous ; The domain and range is not defined. This property could be replaced by `s4syst:hasSubSystem`.
- `s4envi:isConnectedTo` link a system and the system to which it is connected to. This property could be replaced by `s4syst:connectedTo`.
- `s4envi:usesCommunicationInterface` and `s4envi:usesCommmunicationProtocol`.
- `s4envi:System`, from its definition, read more like a CommunicatingSystem.

SAREF4ENVI [i.6] defines architectural ODPs that, given a property (Height, Frequency, Period), defines a sub-class of `saref:Property`, a sub-class of `saref:Measurement`, and a sub-class of `saref:UnitOfMeasure` of it. The definitions always look parallel:

- `s4envi:FrequencyMeasurement`: "Represents the measured value made over a frequency property. It is also linked to the frequency unit of measure in which the value is expressed and the timestamp of the measurement".
- `s4envi:HeightMeasurement`: "Represents the measured value made over a height property. It is also linked to the height unit of measure in which the value is expressed and the timestamp of the measurement".
- `s4envi:PeriodMeasurement`: "Represents the measured value over a period property. It is also linked to the period unit of measure in which the value is expressed and the timestamp of the measurement".

In addition, SAREF4ENVI uses a Naming ODP: the local name for these three measurement sub-classes end with string "Measurement" (`s4envi:HeightMeasurement`, `s4envi:FrequencyMeasurement`, `s4envi:PeriodMeasurement`).

SAREF4ENVI defines several instances of `saref:Property`, almost always using the same pattern, including the how the comment and label are formatted: `"An individual representing the {typeOfProperty}property {label}."@en`.

Some homogenization with other ontologies is necessary:

- SAREF4ENVI, SAREF4WEAR, and SAREF4LIFT, define communication protocols and interfaces. Only SAREF4LIFT uses the patterns from SAREF4SYST.
- SAREF4ENVI, SAREF4INMA, SAREF4AGRI, SAREF4AUTO, all define identifiers and how to link to identifiers of entities.

SAREF4ENVI uses the W3C Basic Geo Vocabulary. It copies the definition of geo Point, `geo:SpatialThing`, and add disjoint axioms to many classes in SAREF. Other ontologies use the OGC GeoSPARQL standard instead. One unique choice should be made for all SAREF.

SAREF4ENVI uses the W3C Time Ontology and copies the definition for `time:TemporalUnit`, but additionally states it is a subclass of `saref:UnitOfMeasure`. This is considered a bad practice.

SAREF4ENVI redefines many units from the OM ontology in its version 1.8. These units of measure are only useful for examples, and a global choice should be made in SAREF about which ontology of units of measure to reuse.

SAREF4ENVI imposes several disjointWith axioms, including across SAREF Core entities. These axioms should either be promoted to SAREF Core, or deleted from SAREF4ENVI.

7.3 Analysis of SAREF4BLDG V1.1.2

SAREF4BLDG V1.1.2 [i.7] copies, sometimes partially, the definition of several entities from SAREF Core:

- OP saref:isMeasuredIn, saref:relatesToProperty.
- DP saref:hasTimestamp, saref:hasValue.
- Classes saref:Actuator, saref:Device, saref:Measurement, saref:Property, saref:Sensor, saref:UnitOfMeasure.

SAREF4BLDG introduces entities that aim to link SAREF Core entities, or are so generic they could be promoted to SAREF Core or replaced by entities from SAREF Core or other vocabularies:

- Class `s4bldg:PhysicalObject`, aligned with the DUL top-level ontology. Exact same definition as in SAREF4ENVI, but different axiomatization:
  - `s4envi:PhysicalObject ⊑ ∀ s4envi:isContainedIn . s4envi:PhysicalObject`, while
  - `s4bldg:PhysicalObject⊑∀s4bldg:isContainedIn. (s4bldg:PhysicalObject⊔s4bldg:BuildingSpace)`
- OP `s4bldg:contains` has a different definition than `s4envi:contains`
- OP `s4bldg:isContainedIn` has a different definition than `s4envi:isContainedIn`

SAREF4BLDG introduces many OPs that are used in universal restrictions on devices or physical objects to link to measurements, for example `s4bldg:flowCoefficient`, `s4bldg:faceArea`, etc. Other ontologies would introduce instances of `saref:Property` in place of these OPs. Some harmonization is required.

Some of these OPs actually represent different kinds of measurements for the same property. For example, `s4bldg:airFlowRateMax`, `s4bldg:airFlowRateMin`, `s4bldg:nominalAirFlowRate`, `s4bldg:primaryAirFlowRateMin`, `s4bldg:primaryAirFlowRateMax`, `s4bldg:secondaryAirFlowRateMin`, `s4bldg:secondaryAirFlowRateMax`.

Some of these OPs actually represent the same kind of measurement for different properties. For example:

- s4bldg:operationTemperatureMax, s4bldg:outletTemperatureMax , s4bldg:pumpFlowRateMax link to a measurement of the maximal *allowable* value for the property.
- s4bldg:fluidFlowRateMax, s4bldg:powerOutputMax, s4bldg:supportedWeightMax link to a measurement of the maximal *possible* value for the property.
- s4bldg:nominalSupplyVoltage, s4bldg:nominalPressureDrop, s4bldg:nominalPowerRate link to a measurement of the nominal value for the property.

Some of these OPs actually contain the name of the type of `saref:FeatureOfInterest` it is applies to. For example, `s4bldg:basinReserveVolume`, `s4bldg:bladeThickness`, `s4bldg:coilLength`, `s4bldg:coilWidth`, (different properties for the same type of feature of interest), `s4bldg:electricMotorEfficiency`, `s4bldg:electricGeneratorEfficiency` (same property for different types of feature of interest).

`xsd:duration` is declared as a `rdfs:Datatype`, but not used.

SAREF4BLDG introduces many DPs, with no domain and a XSD Datatype as a range (`xsd:string` (53), `xsd:boolean` (18), or `xsd:integer` (9), `xsd:nonNegativeInteger` (1)).

SAREF4BLDG defines architectural ODPs that, given a DP p typically applied to a Class c with a certain datatype d, defines this DP with a range d, and a universal restriction on c that all values for p are d. In other words, declared sub-classes of `s4bldg:Device` have universal local restrictions on these DPs, and re-state the datatype is always what it should be. For example:

`⊤ ⊑ ∀ s4bldg:frameSize . xsd:string`

`s4bldg:ElectricMotor ⊑ ∀ s4bldg:frameSize . xsd:string`

This second axiom is useless and could be deleted to avoid redundancy.

Some of the DPs could actually be considered as quantifiable properties and be the object of measurement. For example `s4bldg:roughness` is defined as "A *measure* of the vertical deviations of the surface…". The typical unit for roughness could be micrometer.

Some of the DPs seem to accept enumerated values as objects, but these are only defined in the comment (ex. `s4bldg:refrigerantClass`) or not defined at all (ex. `s4bldg:pipeConnectionEnum`). They could therefore be modelled following a common pattern to be decided for the whole SAREF. For example, as an OP with as a range a sub-class of `skos:Concept`.

Some of the DPs (19) have a local name that ends with "type". This may be considered as a bad naming choice as types are usually modelled as classes in OWL. Some DPs explicit the expected values (ex. `s4bldg:circuitType`), other do not (ex., `s4bldg:integratedLightingType`).

Some homogenization with other ontologies is necessary:

- SAREF4BLDG, SAREF4ENVI define the class of Physical Objects, and properties contains and isContainedIn, with different axiomatization.

SAREF4BLDG copies the definition of the geo:location from the W3C Basic Geo Vocabulary, but it does not use it. Other ontologies use the OGC GeoSPARQL standard instead. One unique choice should be made for all SAREF.

SAREF4BLDG imposes disjointness between saref:Measurement saref:Property, saref:UnitOfMeasure. Although relevant, these axioms should either be promoted to SAREF Core, or deleted from SAREF4BLDG.

7.4 Analysis of SAREF4CITY V1.1.2

SAREF4CITY V1.1.2 [i.8] copies, sometimes partially, the definition of several entities from SAREF Core:

- OP saref:controlsProperty (with wrong comment and no rdfs:isDefinedBy metadata), saref:hasProperty (with wrong comment and no rdfs:isDefinedBy metadata), saref:isControlledByDevice, saref:isMeasuredByDevice (with no comment), saref:isMeasuredIn, saref:isPropertyOf, saref:makesMeasurement, saref:measurementMadeBy, saref:measuresProperty, saref:relatesToMeasurement, saref:relatesToProperty

- Classes saref:Actuator, saref:Device, saref:FeatureOfInterest, saref:Measurement, saref:Property, saref:Sensor, saref:UnitOfMeasure

By doing so, SAREF4CITY keeps some axioms from SAREF V2.1.1 that were deleted in SAREF V3.1.1. For example it still contains a universal restriction on `saref:measurementMadeBy` to only instances of the `saref:Device` class. In fact, `saref:Device` is now the range of `saref:measurementMadeBy` so this local restriction was considered useless, and deleted in SAREF V3.1.1.

Several OPs do not have a meaningful comment. A dummy one was added to pass the tests from the SAREF Pipeline.

OPs do not have domains and ranges, even when it is obvious. For example the range of `s4city:hasKPI` is `s4city:KeyPerformanceIndicator`.

SAREF4CITY does not define subclasses for core SAREF Core classes. It extends SAREF with:

- a hierarchy of classes below `geosp:SpatialObjects` from the OGC GeoSPARQL standard;

- a content ODP for describing events;

- a content ODP for describing Key Performance Indicators.

The class `s4city:City` is arguably best described in SAREF4CITY, but its sibling `s4city:Country` and its superclass `s4city:AdministrativeArea` could be relevant for other SAREF extensions.

The class s4city:Event describes temporal and scheduled events organized by some agent, that take place at a certain time and at a certain facility. This class discards events that take place in a moving facility such as buses or boats. The position of moving facilities could be considered a `saref:Property`, and be measured by devices.

The accessibility of the event (`s4city:hasAccesibility` contains a typo, it should be `s4city:hasAccessibility`. Same in the label. A Comment should be provided) is only provided as skos:Concept. A concept scheme should be provided or linked to.

The class `s4city:KeyPerformanceIndicator` is central in SAREF4CITY. It could be considered analogous to `saref:Property` in SAREF Core. Key performance indicators are defined as types of performance *measurement*, but are usually code lists similar to `saref:Property`. The class `s4city:KeyPerformanceIndicatorAssessment` is analogous to `saref:Measurement` in SAREF Core as it associates a `saref:FeatureOfInterest`, a `s4city:KeyPerformanceIndicator`, and a value. Table 3 continues the analogy.

Table 3: Analogy between saref:Measurement and s4city:KeyPerformanceIndicatorAssessment

s4city:KeyPerformanceIndicator	saref:Property
s4city:isKPIOf	saref:isPropertyOf
s4city:hasKPI	saref:hasProperty
s4city:hasCalculationPeriod	-
s4city:KeyPerformanceIndicatorAssessment	saref:Measurement
s4city:assesses	saref:isMeasurementOf
s4city:isAssessedBy	saref:measurementMadeBy
s4city:quantifiesKPI	saref:relatesToProperty
s4city:hasCreationDate	saref:hasTimestamp
s4city:hasLastUpdateDate	saref:hasTimestamp

This analogy can be used to develop patterns that generalize `s4city:KeyPerformanceIndicatorAssessment` and `saref:Measurement` with a common class, and `s4city:KeyPerformanceIndicator` and `saref:Property` with a common class.

SAREF4CITY uses the W3C Basic Geo Vocabulary. It declares class `geo:SpatialThing`, copies the definition of `geo:location`, `geo:lat`, `geo:long`, `geo:alt`, `geo:Point`. These concepts are not further used in the ontology and could be deleted. `geo:Point` is hijacked and defined as a subclass of `geosp:Geometry`.

SAREF4CITY uses the OGC GeoSPARQL standard. It copies the definition of `geosp:hasGeometry`, `geosp:sfContains`, `geosp:sfWithin`, `geosp:Feature`, `geosp:Geometry`, `geosp:SpatialObject`. Some classes in SAREF4CITY are defined as subclasses of `geosp:Feature`. Class `saref:Device` is hijacked and defined as a subclass of `geosp:Feature`.

SAREF4CITY uses the W3C Time Ontology and declares class `time:TemporalEntity`, `time:Instant`, and `time:Interval`. It uses `time:TemporalEntity` in local restrictions on classes `s4city:Event` and `s4city:KeyPerformanceIndicator`.

SAREF4CITY uses the Friend Of A Friend vocabulary (FOAF). It declares `foaf:Agent` and `foaf:Person`, and defines `s4city:Agent` that provide and use only public services.

SAREF4CITY uses the Public Service Vocabulary published by the European Commission. It copies the definition of class `cpsv:PublicService` and OPs `cpsv:physicallyAvailableAt`, `cpsv:provides`, `cpsv:uses`. However, it leaves out the domains and ranges.

SAREF4CITY imposes disjointness between saref:Measurement saref:Property, saref:UnitOfMeasure. Although relevant, these axioms should either be promoted to SAREF Core, or deleted from SAREF4BLDG.

7.5 Analysis of SAREF4INMA V1.1.2

SAREF4INMA V1.1.2 [i.9] copies, sometimes partially, the definition of several entities from SAREF Core:

- OP saref:hasFunction, saref:hasState, saref:makesMeasurement, saref:measuresProperty, saref:controlsProperty (with no rdfs:isDefinedBy metadata), they are used in local restrictions of s4inma:ProductionEquipment.
- Classes saref:Device, saref:FeatureOfInterest, saref:Function, saref:Measurement, saref:Property, saref:State (with no rdfs:isDefinedBy metadata).

OP `s4inma:belongsToCategory` is very generic and could be promoted to SAREF Core.

OP `s4inma:consistsOfBatch` and `s4inma:consistsOfItem` read close to `saref:consistsOf`, but are unrelated. Either they could be deleted and `saref:consistsOf` could be used instead, or they could be made subproperties of `saref:consistsOf`.

OP `s4inma:hasFeatureOfInterest` and `s4inma:isFeatureOfInterestOf` are named very generically, are defined to link a feature of interest and an equipment, but are in fact also used to link a measurement and a feature of interest. These properties could be confused with `sosa:hasFeatureOfInterest` for example.

OPs do not have domains and ranges, even when it is obvious. For example the range of s4inma:hasFeatureOfInterest could be saref:FeatureOfInterest. The range of s4inma:hasSize could be s4inma:Size.

SAREF4INMA defines patterns for specifying different types of identifiers and how to link to identifiers of entities. Some harmonization is needed with SAREF4ENVI, SAREF4AGRI, and SAREF4AUTO.

SAREF4INMA reuses some classes from SAREF4BLDG such as s4bldg:Building, s4bldg:BuildingSpace, s4bldg:PhysicalObject, and specialize them as s4inma:Factory, s4inma:Area, s4inma:Site, s4inma:ProductionEquipment.This is a case for promoting some of the most generic SAREF4BLDG concepts to SAREF Core.

SAREF4INMA specialize saref:Measurement into s4inma:Measurement, with the additional restriction that it is about a Batch or Item feature of interest. Having the same local name for these two concepts could be confusing to the users, s4inma:Measurement cannot apply to s4inma:Factory or s4inma:ProductionEquipment for example.

SAREF4INMA defines an architectural ODP that consists in sub-classing s4inma:Measurement to specify the kind of measurement that is performed: s4inma:ActualMeasurement and s4inma:ExpectedMeasurement. Both subclasses end with string "Measurement" (naming ODP pattern). This could be generalized for other types of measurements such as temporal or logical aggregates, like "MinimumMeasurementOverTemporalInterval", etc.

SAREF4INMA defines an architectural ODP that derives, for certain entities, a class for batch and categories of these entities. For example s4inma:Item - s4inma:ItemBatch – s4inma:ItemCategory, s4inma:MaterialBatch – s4inma:MaterialCategory, s4inma:ProductionEquipment – s4inma:ProductionEquipmentCategory. OP s4inma:isCategoryOf links the latter element to the underlined element. The notion of Category allows to group and describe similar properties for a uniform collection of objects, thus factorizing their description. This may be generalized to other concepts in SAREF and extensions, such as saref:DeviceCategory, s4bldg:BuildingCategory, etc.

SAREF4INMA specializes saref:Property into the class s4inma:Size ("the amount of certain objects in a collection"). Depending on the choice for modeling properties, this may be turned into a named individual instead. SAREF4INMA also defines OP s4inma:hasSize, but this OP does not specialize saref:hasProperty. It is used to link a s4inma:Batch to its size, as a specific property (the size *of a certain batch*), but nothing is said about how to measure it.

SAREF4INMA specializes saref:Function into the class s4inma:ProductionEquipmentFunction. The naming is thus "{name of the feature of interest}Function", while in SAREF Core it is often "{type of action}Function".

"Batch" is said to be a production process in the definition of s4inma:create, but defined as a collection of tangible objects in its own definition.

The definition of s4inma:WorkCenter states that it is a subclass of s4inma:ProductionEquipment and therefore also of s4bldg:PhysicalObject, however this is neither asserted nor can be inferred. It may be appropriate to explicit that saref:Device is a subclass of s4bldg:PhysicalObject, or saref:PhysicalObject if this entity gets promoted to SAREF Core.

SAREF4INMA redefines and specializes the class skos:ConceptScheme from the W3C Simple Knowledge Organization System (SKOS) vocabulary. It considers that s4inma:ID is a sort of concept scheme, like taxonomies, thesaurus, etc.

7.6 Analysis of SAREF4AGRI V1.1.2

SAREF4AGRI V1.1.2 [i.10] copies, sometimes partially, the definition of most entities from SAREF Core. These could be deleted to limit the number of changes necessary in SAREF4AGRI when SAREF Core evolves. For example saref:FeatureOfInterest contains rdfs:comment <https://saref.etsi.org/core/>, which is wrong.

SAREF4AGRI defines s4agri:Intake and s4agri:Yield, as "amounts". These could be modeled as properties of an animal or agricultural product.

SAREF4AGRI defines several instances of saref:Property or its subclasses. This indicates that properties are generic to many features of interest.

The following entities have two comments. Sometimes a comment is just a copy paste of the label. geo:Point, saref:Property, saref:UnitOfMEasure, s4agri:Deployment, s4agri:hasDeploymentPeriod, s4agri:isDeployedAtSpace.

SAREF4AGRI defines some OP that are very generic, such as s4agri:hasName, and saref:hasDescription. saref:hasDescription has been deprecated in SAREF. Other properties should be used, for example from the RDFS or Dublin Core Terms vocabularies.

SAREF4AGRI defines four DPs with range xsd:dateTime: s4agri:hasPlantDate and s4agri:hasHarvestDate, s4agri:hasBirthDate and s4agri:hasDeathDate. It may be useful to generalize how events occurring over the lifespan of features of interests are modeled, maybe using a pattern. For example generalizing the class s4city:Event. The kind of event could then be defined by a link to a concept from a code list that depend on the type of feature of interest.

OP s4agri:isLocatedIn applies for example to s4agri:Animal or s4agri:AnimalGroup to express the physical location of the entity. As such, the location of the animal cannot change. It may be useful to model the location of an entity using a saref:Property, that can be measured.

SAREF4AGRI defines classes s4agri:Building, s4agri:BuildingSpace, and OP s4agri:contains. These local names are already used to define entities in SAREF4BLDG, which may create confusion to the end users.

SAREF4AGRI defines seven types of sensors. None of them have restrictions associated such as the type of property they measure. Yet, the definitions of some sensors is explicit about which property they measure. For example the s4agri:SoilTensiometer is said to measure the soil moisture, but it is not linked to s4agri:SoilMoisture.

SAREF4AGRI uses the Friend Of A Friend (FOAF) vocabulary, which is not an OWL ontology. It declares OP foaf:based_near, foaf:member, and classes foaf:Agent and foaf:Person, and hijacks the class foaf:Agent to additionally assert that all values for foaf:member are foaf:Agent.

SAREF4AGRI uses schema.org vocabulary, which is not an OWL ontology. It declares class schema:Organization as a subclass of foaf:Agent.

SAREF4AGRI uses the W3C Simple Knowledge Organization System (SKOS) ontology. It declares skos:definition and skos:prefLabel, but do not use them elsewhere in the ontology. These declarations could be deleted.

SAREF4AGRI uses the W3C Provenance ontology (PROV-O). It declares prov:hadPrimarySource as an AP, while in PROV-O this entity is declared as an OP. This may induce inconsistencies. Furthermore, this entity is not used anywhere else in SAREF4AGRI. It may be deleted.

SAREF4AGRI uses the W3C Basic Geo Vocabulary, like SAREF4CITY but with a different prefix wgs84:. It declares class geo:SpatialThing, copies the definition of geo:location, geo:lat, geo:long, geo:alt, geo:Point. These concepts are not further used in the ontology and could be deleted. In particular, geo:long is defined as an OP in SAREF4AGRI while it is defined as a DP in SAREF4CITY. Therefore, using these two ontologies together would lead to an inconsistency. geo:Point is hijacked and defined as a subclass of geosp:Geometry.

SAREF4AGRI uses the OGC GeoSPARQL standard, like SAREFCITY but with a different prefix geo:. It copies the definition of geosp:hasGeometry, geosp:sfContains, geosp:sfWithin, geosp:Feature, geosp:Geometry, geosp:SpatialObject. Some classes in SAREF4CITY are defined as subclasses of geosp:Feature. Class saref:Device is hijacked and defined as a subclass of geosp:Feature.

SAREF4AGRI uses the OGC and W3C SOSA/SSN ontologies. It redefines OP sosa:hosts, sosa:isHostedBy, and ssn:hasSubSystem, but these are not used in the rest of the ontology. It redefines the entities related to deployment from SOSA/SSN, and extend ssn:Deployment with s4agri:Deployment, linking to the system that is deployed, the platform where it is deployed, the temporal entity of the deployment, and the geo:SpatialObject where the deployment occurs. Deployment may be considered important for SAREF and be promoted to SAREF Core, under the saref: namespace.

SAREF4AGRI uses the W3C Time ontology. It copies the definition of time:Instant, time:Interval, and time:TemporalEntity. Other SAREF ontologies use these concepts, therefore it could be useful to define them at the level of SAREF Core.

SAREF4AGRI uses the Ontology of Measurement vocabulary, in its version 2. SAREF4ENVI use this vocabulary in a different version 1.8. IRIs of these concepts contain the version number and are thus tied to a specific version. Apart from being redefined in SAREF4AGRI, these units of measure are not used in the ontology. They could be safely deleted.

7.7 Analysis of SAREF4AUTO V1.1.1

SAREF4AUTO V1.1.1 [i.11] copies, sometimes partially, the definition of most entities from SAREF Core. These could be deleted to limit the number of changes necessary in SAREF4AUTO when SAREF Core evolves.

SAREF4AUTO uses an architectural pattern that consists in defining an OP (for example, `s4auto:hasAutomationLevel`) with no domain and range, a class that would be an obvious choice for the range of that OP (for example, `s4auto:AutomationLevel`), but no instance that populate this class.

SAREF4AUTO uses an architectural pattern that consists in defining sub-properties of `saref:hasProperty` (for example, `s4auto:hasHeight`) with no domain and range, then a class that would be an obvious choice for the range of that OP (for example, `s4auto:Height`), and define SomeValuesFrom restrictions on several classes that would have this property (ex., `s4auto:Vehicle`, `s4auto:RoadEntity`). This suggests SAREF4AUTO is using properties as specific to features of interest.

OP `s4auto:detectsPosition` links a device to the position it can detect. If positions were modelled as properties, this OP could be replaced by `saref:measuresProperty`. Similarly, OPs `s4auto:hasDestination`, `s4auto:hasPosition`, `s4auto:hasOrigin`, `s4auto:hasEstimatedRendezvousLocation`, could all be modelled as subproperties of `saref:hasProperty`, or as instances of `saref:Property` (with some renaming). By the way, the same position could be measured or evaluated in different ways: absolutely with a latitude and a longitude, relatively to some point of interest, or relatively to some other point of interest.

SAREF4AUTO defines many sub-classes of `saref:Property`, which suggests SAREF4AUTO is using properties as specific to features of interest. However the class `s4auto:Movement` contains five instances such as `s4auto:crossingLeft` and `s4auto:crossingRight`. These may be considered different possible values for the property `s4auto:Movement`. Their modelling could be updated in harmonization with the modelling of postures in SAREF4EHAW.

SAREF4AUTO defines many sub-classes of `saref:State`, and in parallel, many sub-properties of `saref:hasState` with similar names, but no domain or range. In the bottom of the hierarchy of `saref:State`, classes like `s4auto:Searching`, `s4auto:Platooning`, etc. represent specific states that need to be instantiated and timestamped.

The specific state `s4auto:Unknown` is defined twice as a class, as sub-classes of `s4auto:PlatooneState` and `s4auto:PlatoonVehicleState`, and once as an individual, instance of `s4auto:PlatoonRole`.

SAREF4AUTO defines membership of vehicles in platoons or of entities in vehicle environment. This requirement is also found in SAREF4AGRI for animals that belong to animal groups, and could be useful in other extensions as well. This may call for a promotion to SAREF Core.

SAREF4AUTO defines an interesting architectural ODP for roles that some entities play, which link to a class populated by a set of individuals. For example, `s4auto:hasPlatoonRole` - `s4auto:PlatoonRole` and `s4auto:hasVehicleRome` – `s4auto:VehicleRole`. A generalization of this ODP could be promoted to SAREF Core.

SAREF4AUTO uses the W3C Time ontology. It copies the definition of `time:Instant`, `time:Interval`, and `time:TemporalEntity`, and several properties. Other SAREF ontologies use these concepts, therefore it could be useful to define them at the level of SAREF Core.

SAREF4AUTO uses the W3C Basic Geo Vocabulary, with no prefix. It copies the definition of `geo:location`, `geo:lat`, `geo:long`, `geo:alt`, `geo:Point` and `geo:SpatialThing`. These concepts are used in the ontology in the definition of `s4auto:Point` and `s4auto:AbsolutePosition`.

SAREF4AUTO uses the OGC GeoSPARQL standard, with prefix `geosp:`. It copies the definition of `geosp:hasGeometry`, `geosp:Feature`, `geosp:Geometry`, `geosp:SpatialObject`. These concepts are used in the ontology in the definition of `s4auto:Point` and `s4auto:ParkingSpot`.

7.8 Analysis of SAREF4EHAW V1.1.1

SAREF4EHAW V1.1.1 [i.12] declares and sometimes copies the definition of some SAREF entities. These could be deleted to limit the number of changes necessary in SAREF4EHAW when SAREF Core evolves.

SAREF4EHAW seems to be the result of a manual transformation of a relational database schema to an ontology. It falls in several pitfalls. For example, SAREF4EHAW uses enumerated string datatypes as range of some properties such as `s4ehaw:banTopology`. This contravenes best practices as it makes it impossible to define other topologies.

SAREF4EHAW extensively uses and extends the SAREF classes `saref:Service`, `saref:Function`, and saref:Command. No other extension currently does. A deeper analysis of the way services are modelled in SAREF4EHAW could be necessary, and maybe some proposals could be promoted to SAREF Core. In particular, it is unclear how the classes `s4ehaw:ServiceGrounding` (how to access the service), `s4ehaw:ServiceProcess` (how the service works), and `s4ehaw:ServiceProfile` (what the service does) may be used. SAREF4WEAR defines OPs that can serve as the basis of service composition (input, precondition, effect, output, result), which has to be clarified.

SAREF4EHAW defines one sub-class of `saref:Property`, `s4ehaw:Posture`, which has five instances such as `s4ehaw:Walking`. These may be considered different possible values for the property `s4auto:Posture`, which should be modelled as an instance of `saref:Property`. Their modelling could be updated in harmonization with the modelling of postures in `s4auto:Movement`.

It is unclear the relation between `s4ehaw:Mode` and `saref:State`. It could be deleted and `saref:State` could be used instead.

It is unclear what the class `s4ehaw:Data` represents for functions, and how it is intended to be used. The definition states: "A function has one or many data". It may indicate for the need to link a `saref:Function` to some instances of `saref:Property` (currently with OP `s4ehaw:hasData`), however a class is not needed for that.

Some classes are unnecessary named after the property they are the range of. For example `s4ehaw:Contact` is the range of `s4ehaw:hasContact`, and asserted to be equivalent to s4ehaw:HealthActor. Only the latest could suffice.

DP `s4ehaw:serialNb` conflicts with how serial numbers are represented in SAREF4INMA.

SAREF4EHAW attempts to model time series as `s4ehaw:TimeSeriesMeasurement`, a subclass of `saref:Measurement`, which has exactly one `xsd:decimal` as a value. The modelling is partial. Time series may be of interest to other extensions. Some more discussion is needed to model time series in SAREF, and this part could be deleted from SAREF4EHAW in the meantime.

SAREF4EHAW uses the Friend Of A Friend vocabulary (FOAF). It declares `foaf:Agent` and declares different sub-classes of that class, which are more roles than classes. For example, a person could have the role of a caregiver in some situation, and of a patient in another situation. A ODP for roles should be used instead.

7.9 Analysis of SAREF4WEAR V1.1.1

SAREF4WEAR V1.1.1 [i.13] copies, sometimes partially, the definition of many entities from SAREF Core, SAREF4CITY, SAREF4SYST. These could be deleted to limit the number of changes necessary in SAREF4WEAR when these ontologies evolve.

SAREF4WEAR hijacks SAREF4CITY by defining `s4city:Event` as a subclass of `s4wear:Occurrence`.

SAREF4WEAR introduces different OPs that link devices to the features of interest they measure or control. This is missing from SAREF Core and could be useful to other extensions. It could be promoted to SAREF Core.

OP `s4wear:hasPowerSupply` is used to link a device to the type of power supply it is equipped with. SAREF4LIFT also models similar knowledge, differently. Some harmonization may be needed.

OP `s4wear:hasSensor` could be replaced by `saref:consistsOf`, which serves the same purpose.

OP `s4wear:installs` could be deleted if the notion of Deployment from OGC and W3C SOSA/SSN is promoted to SAREF Core.

SAREF4WEAR defines an architectural ODP, defining three OPs `s4wear:isLocatedIn`, `s4wear:isLocatedNear`, `s4wear:isLocatedOn`, which are all sub-properties of `s4wear:isLocated` and whose domain is respectively

s4wear:InBodyWearable, s4wear:NearBodyWearable, s4wear:OnBodyWearable. A similar ODP could be adopted for other types of devices and located differently on/in/near other types of features of interest.

Some static properties are provided as DPs, for example s4wear:hasLength, s4wear:hasHeight, s4wear:hasWeight. OPs with same local name are present in other SAREF extensions. Some naming ODP should be chosen to avoid ambiguity between DP and OP names. The range of these DPs is xsd:double, but no unit is mentioned.

SAREF4WEAR defines DP s4wear:hasCommand when there exists a OP saref:hasCommand for a similar purpose. The modelling pattern from SAREF could be used instead.

Devices types s4wear:MemoryStorage and s4wear:PowerSupply, and Feature of Interest types s4wear:Software and s4wear:User, could be useful to other extensions and could be promoted to SAREF Core.

OPs s4wear:triggers and s4wear:isTriggeredBy could be useful for other extensions and could be promoted to SAREF Core.

SAREF4WEAR adopts the generic properties modelling choice as it does define different subclasses of saref:Property, and some classes contain a set of instances such as s4wear:CrowdSize and s4wear:HeartRate.

SAREF4WEAR does use the SAREF4SYST pattern and specializes s4syst:connectedTo into s4wear:sendsInformationTo and s4wear:sendsNotificationsTo.

SAREF4WEAR uses the SSN System module from the OGC and W3C SOSA/SSN ontology, to assign system capabilities to s4wear:Wearable instances. A SAREF version of this module could be developed and promoted to SAREF Core.

Both schema:Person and foaf:Person are declared, but foaf:Person is not used. It could be deleted.

SAREF4WEAR uses the W3C Basic Geo Vocabulary, with prefix geo:. It copies the definition of geo:location, geo:Point. These concepts are not further used in the ontology and could be deleted. geo:Point is hijacked and defined as a subclass of geosp:Geometry.

SAREF4WEAR uses the OGC GeoSPARQL standard with prefix geosp:. It copies the definition of geosp:hasGeometry, geosp:sfContains, geosp:sfWithin, geosp:Feature, geosp:Geometry, geosp:SpatialObject. Some classes in SAREF4WEAR are defined as subclasses of geosp:Feature. Or geosp:SpatialObject.

7.10 Analysis of SAREF4WATR V1.1.1

SAREF4WATR V1.1.1 [i.14] copies, sometimes partially, the definition of many entities from SAREF Core, SAREF4CITY, SAREF4SYST. These could be deleted to limit the number of changes necessary in SAREF4WATR when these ontologies evolve.

SAREF4CITY concepts of Key Performance Indicator and Key Performance Indicator Assessment are redefined in SAREF4WATR, but not used elsewhere by the ontology. This makes a case for promotion of these concepts to SAREF Core, and they could be deleted from SAREF4WATR.

OP s4watr:hasPenomenonTime is introduced on saref:Measurement as a more precise property than saref:hasTimestamp. This property is named and defined after sosa:hasPhenomenonTime from the W3C SOSA/SSN ontology, and could be promoted to SAREF Core.

SAREF4WATR does define many subclasses of geosp:SpatialObject, saref:FeatureOfInterest, and s4syst:System. This could be recommended for other extensions as well.

SAREF4WATR does define many subclasses of saref:Device that are specific to the Water domain, which is something that could be recommended for other extensions as well.

Class s4watr:Water is a subclass of saref:FeatureOfInterest, and has different individuals. It may be problematic that this class contains generic individuals such as s4watr:DrinkingWater etc. which will be the object of measurements of all kinds. It may be better to have a pattern where these concepts are instances of s4watr:WaterKind, and an instance of s4watr:Water (a specific quantity of water) is linked to its kind.

Class saref:Meter is defined in SAREF4WATR as a subclass of saref:Sensor, which is not an axiom that is part of SAREF Core and could be considered hijacking the concept.

SAREF4WATR defines different subclasses of saref:Property, which are then populated by many instances. Thus, SAREF4WATR adopts the generic property modelling choice.

SAREF4WATR reuses the schema.org vocabulary. It declares schema:Person and schema:Organization. Other ontologies use FOAF for these entities. One unique choice should be made for SAREF.

The notion of s4watr:Tariff may be useful for other commodities and other extensions, and could be promoted to SAREF Core.

DPs s4watr:fabricationNumber, s4watr:hasFirmwareVersion, s4watr:hasHardwareVersion, s4watr:hasVersion, could apply to other extensions and could be promoted to SAREF Core.

DP s4watr:operatesAtRadioFrequency would best be applied to some ontology of communication protocols. Other extensions SAREF4LIFT, SAREF4EHAW, SAREF4WEAR, do define communication interfaces and protocols. Some harmonization would be required.

The class of s4watr:WaterUse is populated by a set of instances. Many occurrences of such a pattern are found in other extensions. This could be a candidate for defining a pattern where skos:Concept and skos:ConceptScheme may be used.

SAREF4WATR uses the W3C Time ontology. It copies the definition of time:DayOWeek, time:Instant, time:Interval, time:TemporalEntity, and time:TemporalDuration, and several properties. Other SAREF ontologies use these concepts, therefore it could be useful to define them at the level of SAREF Core. SAREF4WATR use these concepts in ranges of some properties that apply to the class s4watr:Tariff, which may be interesting to promote to SAREF Core.

SAREF4WATR uses the OGC GeoSPARQL standard, with prefix geosp:, and the companion OGC Spatial Features ontology, with the prefix sd: It copies the definition of geosp:hasGeometry, geosp:sfContains, geosp:sfWithin, geosp:Feature, geosp:Geometry, geosp:SpatialObject, sf:Point, sf:Polygon. Only geosp:Feature is used as a superclass of s4watr:WaterAsset and s4watr:WaterInfrastructure.

7.11 Analysis of SAREF4LIFT V1.1.1

SAREF4LIFT V1.1.1 [i.15] extensively uses the SAREF reference patterns from SAREF4SYST.

Several classes and OPs relate to the building and to the energy domain, and could be promoted in respective extensions.

SAREF4LIFT defines several subclasses of saref:Command. Instances of commands are timestamped, may have a value (saref:hasValue), and be about (saref:isAbout) some entity.

SAREF4LIFT defines many sub-classes of saref:Property such as s4lift:Load and s4lift:Voltage, and adopts the specific property modelling choice. Some harmonization is required with other extensions.

SAREF4LIFT defines many subclasses of saref:State, such as s4lift:FireOperationState, and s4lift:DetectedLoadState. Instances of saref:State are timestamped and are about some feature of interest. As such, they are used like measurements.

SAREF4LIFT defines many subclasses of saref:Measurement, such as s4lift:NumberOfCallsStatistics. These measurements can be timestamped with saref:hasTimeStamp, but the phenomenon time is not available in SAREF. The property s4watr:phenomenonTime could be useful in SAREF4LIFT.

SAREF4LIFT defines a content ODP for s4lift:Signal, that convey some information such as a fault, a state, or a measurement. Descendants of s4lift:Signal inherit the saref:hasTimestamp, saref:isAbout, and s4lift:conveys properties defining the timestamp when the signal has been generated, the features of interest that the signal is about, and the states or measurements conveyed, respectively.

SAREF4LIFT defines entities to model network connectivity and network connections. This may be useful to other extensions and could be promoted to SAREF Core.

8 Cross-vertical analysis of candidate patterns

Each table below describes a single candidate pattern. It lists in some SAREF version the candidate instances of a certain pattern.

1	**SP_MustChooseOneFromTwoClasses:** only two discrete, disjoint class choices that are subClassOf a *parentClass*. Often used as an *owl:someValuesFrom* restriction. Not extendable.
SAREF-CORE	*SetLevelCommand*: SetRelativeLevelCommand, SetAbsoluteLevelCommand *State:* OnOffState, OpenCloseState, StartStopState, MultiLevelState?
SAREF4AUTO	*Speed:* RelativeSpeed, AbsoluteSpeed *Position:* RelativePosition, AbsolutePosition *ParkingSpot:* RegularParkingSpot, ElectricChargingParkingSpot
SAREF4LIFT	*MovingDirectionState:* MovingDownwardDirectionState, MovingUpwardDirectionState

2	**SP_MustSpecifyBothClasses**: only two discrete class choices that are subClassOf a parent class, both which has to be specified. Not extendable.
SAREF4AUTO	*Heading:* YawRate, AngularDirection *Acceleration:* LongitudinalAcceleration, LateralAcceleration

3	**SP_VariableStateOfNamedIndividual**: a type of NamedIndividual that can exist in only one state out of related, multiple, disjoint class states but can change this state at another time. Not extendable.
SAREF4AUTO	*VehicleRole:* ReverseDirection, SameDirection, Static, CrossingLeft, CrossingRight, CrossingLeft. *PlatoonRole:* Follower, Leader, ReadyForLeading, Trailing, Trailing

4	**SP_FixedStateOfNamedIndividual**: a type of NamedIndividual that can exist in only one state out of related, multiple, disjoint class states and cannot change this state. Not extendable.
SAREF4AUTO	*VehicleRole:* Agriculture, Commercial, DangerousGood, Military, PublicTransport, Rescue, RoadOperator, Emergency, RoadWork, SafetyCar, SpecialTransport, Taxi.

5	**SP_VariableStateOfClass**: a type of class that can exist in only one state out of multiple, disjoint class states but can change this state at another time. Not extendable.
SAREF4AUTO	*ParkingVehicleState:* AtDropOffSpot, AtPickUpSpot, DrivingToParkingSpot, DrivingToPickUpSpot, Parked, Parking, Charging. *ParkingSpotState:* Charging, Closed, Free, Occupied, Reserved. *PlatoonState:* Assembling, Disengaging, Platooning, Standalone, Unknown. *PlatoonVehicleState:* Disengaging, Engaging, Forming, Platooning, Searching, Standalone, Unknown.

6	**SP_MultipleNamedIndividualAttributes**: a set of related (derived from one class) NamedIndividual attributes. Extendable.
SAREF4WATR	*ChemicalProperty:* Acrylamide, Aluminium, Ammonium, Antimony, Arsenic, Benzene, Benzoapyrene, Boron, Bromate, Cadmium, Chloride, Chromium, Copper, Cyanide, Epichlorohydrin, Fluoride, Iron, Lead, Manganese, Mercury, Nickel, Nitrate, Nitrite, Oxygen, PolycyclicAromaticHydrocarbons, Selenium, Sodium, Sulphate, Tetrachloroethene, TotalOrganicCarbon, Trichloroethene, Trihalomethanes, Tritium, VinylChloride, 12Dichloroethane. *BacterialProperty:* ClostridiumPerfringens, ColiformBacteria, ColonyCount22C, ColonyCount37C, Enterococci, EscherichiaColi, PseudomonasAeruginos. *AcceptabilityProperty:* Colour, Conductivity, Hardness, HydrogenIonConcentration, Odour, Oxidisability, Taste, Temperature, TotalIndicativeDose, TotalDissolvedSolids, TotalSuspendedSolids, Turbidity. *EnvironmentalProperty:* AtmosphericPressure, ExternalTemperature, Humidity, Precipitation. WaterMeterProperty*:* BatteryOperatingTime, BatteryRemainingTime, BatteryLastChange, MeterOnTime, MeterOperatingTime. *WaterUse:* Agriculture, Aquaculture. *Water:* DrinkingWater, RawWater, Stormwater, Wastewater. *WaterFlowProperty:* FlowPressure, FlowRate, FlowTemperature, FlowVolume. *WaterUse:* Domestic, Industry, Recreation.

7	**SP_MultipleClassAttributes**: a type of class that links to multiple, related, class states or attributes ultimately derived from a single *parentClass* {that may or may not exist within a hierarchy} and which can change at any time. Extendable.
SAREF4LIFT	*State:* AlarmState {AlarmInTheCarState, AlarmInTheWellState, AlarmInTheRoofState, AlarmInTheMachineryState}, AlarmVoiceCommunicationState, FireOperationState, SmartLiftInstallationFaultState, CarInUnlockingSpaceState, InspectionOperationState, OutOfServiceState, OverloadedState, TestRideInExecutionState, DetectedLoadState, PowerSupplyState {StandardPowerSupplyState, EmergencyPowerSupplyState}, ReservedServiceState, RealTimeModeState, BatteryPowerState {GoodBatteryPowerState, WarnBatteryPowerState, CriticalBatteryPowerState, InsufficientBatteryPowerState}. *StatisticsMeasurement:* NumberOfCallsStatistics, NumberOfFaults, TotalFloorsCoveredStatistics, NumberOfResetSequencesStatistics, TotalReversalDirectionStatistics, TotalNumberOfOpeningOfDoorStatistics, UpwardTravelsStatistics, DownwardTravelsStatistics.

8	**SP_DatatypeList**: a *range* of related, disjoint DatatypeProperty attributes with a specified class domain and qualified by EnumeratedDatatype. Order is not guaranteed. Not extendable. Collections differ from containers in allowing branching structure with an explicit terminator.
SAREF4EHAW	*hasGender:* rdfs:domain s4ehaw:HealthActor; *banTopology:* rdfs:domain s4ehaw:Ban.

9 Modelling discrepancies issues in the SAREF suite of ontologies and recommended resolution

Modelling discrepancies identified issues in the SAREF suite of ontologies and proposed resolution are recorded as issues in the saref-portal repository on the ETSI Forge https://saref.etsi.org/sources/saref-portal/issues/.

These proposals are input for the harmonisation of the SAREF suite of ontologies and for the development of the next versions of each ontology. They have been discussed during online SmartM2M meetings. When a resolution has been agreed on, this is stated explicitly.

Proposals use the following labels:

- AGREED: the proposed solution has been agreed to.
- CANDIDATE PATTERN: the identified feature is a candidate pattern for use in other ontologies.
- CLARITY ISSUE: the identified issue makes the ontology misleading, or ambiguous, which may lead to misinterpretations, and therefore interoperability issues.

- CONTRAVENES BEST PRACTICE: the identified issue contravenes best practices. Adopting best practices encourages the adoption of SAREF.
- LACK OF HOMOGENEITY: A modelling choice in this SAREF version is different than in some other SAREF version
- PROMOTION TO CORE: Describes promotion of terms defined by extensions to SAREF Core.
- REDUNDANCY ISSUE: Some term or part of the ontology is already described in another SAREF version, potentially differently.

As of 2023-07-25, there is a total of 91 topics, some with multiple reasons: the solution to 9 topics has been agreed to, 16 describe candidate patterns, 20 topics request clarifications, 16 topics contravene best practices, 17 issues involve a lack of homogeneity, 14 suggest promotions to core, 12 topics of redundancy.

History

Document history		
V1.1.1	September 2023	Publication