TTC技術レポート
Technical Report

TR-M2M-0061v0.5.0

スマートシティサービスでのオントロジーの検討

Study on ontologies for Smart City Services

2023 年 3 月 17 日制定

一般社団法人
情報通信技術委員会

THE TELECOMMUNICATION TECHNOLOGY COMMITTEE

TR-M2M-0061v0.5.0

スマートシティサービスでのオントロジーの検討 [Study on ontologies for Smart City Services]

＜参考＞ [Remarks]

１．国際勧告等の関連 [Relationship with international recommendations and standards]

本技術レポートは、oneM2M で作成された Technical Report TR-0061-V0.5.0 に準拠している。

[This Technical Report is transposed based on the Technical Report TR-0061-V0.5.0 developed by oneM2M.]

２．作成専門委員会 [Working Group]

oneM2M 専門委員会 [oneM2M Working Group]

oneM2M
Technical Report

Document Number	oneM2M-TR-0061 -V-0.5.0
Document Name:	Study on ontologies for Smart City Services
Date:	2022-12-01
Abstract:	Smart cities create a diverse landscape of functions and frameworks, which are implemented at large scales and support numerous ICT functions. In order to support various services via oneM2M Smart City platforms, the development of ontologies for service domains in Smart City is an essential work to be done in oneM2M. This TR intends to develop ontologies for services in Smart City.
Template Version: January 2017 (Do not modify)	

About oneM2M

The purpose and goal of oneM2M is to develop technical specifications which address the need for a common M2M Service Layer that can be readily embedded within various hardware and software, and relied upon to connect the myriad of devices in the field with M2M application servers worldwide.

More information about oneM2M may be found at: http//www.oneM2M.org

Contents

2022 Scope

The present document is to provide studies on ontologies for smart city services.

2 References

2.1 Normative references

Normative references are not applicable in the present document.

2.2 Informative references

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] oneM2M Drafting Rules (http://member.onem2m.org/Static_pages/Others/Rules_Pages/oneM2M-Drafting-Rules-V1_0.doc)

[i.2] ETSI Smart Appliances REFerence (SAREF) ontology

NOTE: Available at https://sites.google.com/site/smartappliancesproject/ontologies/reference-ontology

[i.3] ETSI STF 534

NOTE: Available at https://portal.etsi.org/STF/stfs/STFHomePages/STF534

[i.4] AIOTI WG08 Report on Smart Cities

NOTE: Available at https://aioti.eu/aioti-wg08-report-on-smart-cities/

[i.5] H2020 Lighthouse projects

NOTE: Available at https://smartcities-infosystem.eu/scc-lighthouse-projects

[i.6] SAREF extension for Smart Cities Version 0.8

NOTE: Available at https://w3id.org/def/saref4city

[i.7] Open Geospatial Consortium (OGC) GeoSPARQL – A Geographic Query Language for RDF Data Version 1.0

NOTE: Available at https://www.opengeospatial.org/standards/geosparql

[i.8] W3C Recommendation 19 October 2017: "Time ontology in OWL"

NOTE: Available at https://www.w3.org/TR/owl-time/

[i.9] FOAF Vocabulary Specification 0.99 14 January 2014 – Paddington Edition

NOTE: Available at http://xmlns.com/foaf/spec/

[i.10] Core Public Service Vocabulary Application Profile

NOTE: Available at https://ec.europa.eu/isa2/solutions/core-public-service-vocabulary-application-profile-cpsv-ap_en

[i.11] W3C Semantic Web Interest Group: "Basic Geo (WGS84 lat/long) Vocabulary"

NOTE: Available at https://www.w3.org/2003/01/geo/

[i.12] FIWARE Key Performance Indicator data model

NOTE: Available at https://fiware-datamodels.readthedocs.io/en/latest/KeyPerformanceIndicator/doc/spec/index.html

[i.13] Dublin Core ontology

NOTE: Available at http://www.dublincore.org/specifications/dublin-core/dcmi-terms/terms/LinguisticSystem/

[i.14] W3C Recommendation 16 January 2014: “The Organization Ontology”

NOTE: Available at https://www.w3.org/TR/vocab-org/#class-organizationalunit

[i.15] SEAS ontology Version 1.1

NOTE: Available at https://w3id.org/seas/seas-1.1

[i.16] European ITEA project

NOTE: Available at https://itea3.org/about-itea.html

[i.17] W3C Recommendation 11 December 2012 : “OWL 2 Web Ontology Language Structural Specification and Functional-Style Syntax (Second Edition)”

NOTE: Available at http://www.w3.org/TR/2012/REC-owl2-syntax-20121211/

[i.18] Procedure Execution ontology Version.1.1

NOTE: Available at https://w3id.org/pep/pep-1.1

[i.19] ITEA2 12004 Smart Energy Aware Systems Deliverable 2.2: “The SEAS Knowledge Model”

NOTE: Available at http://www.maxime-lefrancois.info/docs/SEAS-D2_2-SEAS-Knowledge-Model.pdf

[i.20] W3C Recommendation 19 October 2017: “ Semantic Sensor Network Ontology”

NOTE: Available at https://www.w3.org/TR/vocab-ssn/

[i.21] R. Arp, B. Smith, A. D.Spear, “Building Ontologies with Basic Formal Ontology” (Book)

[i.22] P. Hitzler, M. Krtzsch, S. Rudolph, “Foundations of Semantic Web Technologies”. (Book)

[i.23] S. Staab, R. Studer, “Handbook on Ontologies” (Book)

[i.24] W3C Recommendation 10 February 2004: “OWL Web Ontology Language Reference”

NOTE: Available at https://www.w3.org/TR/owl-ref/

[i.25] C. Maria Keet, 8 October 2012, “Detecting and Revising Flaws in OWL Object Property”.

NOTE: Available at https://link.springer.com/chapter/10.1007/978-3-642-33876-2_23

[i.26] SEAS Feature of Interest Ontology Version 1.0.

NOTE: Available at https://ci.mines-stetienne.fr/seas/FeatureOfInterestOntology

[i.27] SEAS Evaluation Ontology Version 1.0.

NOTE: Available at https://ci.mines-stetienne.fr/seas/EvaluationOntology-1.0

[i.28] Procedure Execution Ontology Version 1.1.

NOTE: Available at https://ci.mines-stetienne.fr/pep/

[i.29] SEAS Upper Ontology Version 0.10.

NOTE: Available at https://ci.mines-stetienne.fr/seas/UpperOntology

[i.30]	The SEAS Device Ontology Version 1.1.
NOTE:	Available at https://ci.mines-stetienne.fr/seas/DeviceOntology
[i.31]	The SEAS Player Ontology Version 1.1.
NOTE:	Available at https://ci.mines-stetienne.fr/seas/PlayerOntology-1.1
[i.32]	The SEAS Zone Ontology Version 1.0.
NOTE:	Available at https://ci.mines-stetienne.fr/seas/ZoneOntology
[i.33]	SEAS-TechnicalSystemOntology Version 0.9.
NOTE:	Available at https://ci.mines-stetienne.fr/seas/TechnicalSystemOntology
[i.34]	The SEAS Forecasting Ontology Version 1.1.
NOTE:	Available at https://ci.mines-stetienne.fr/seas/ForecastingOntology
[i.35]	SEAS-WeatherOntology Version 0.9.
NOTE:	Available at https://ci.mines-stetienne.fr/seas/WeatherOntology
[i.36]	The SEAS Generic Property Version 1.0.
NOTE:	Available at https://ci.mines-stetienne.fr/seas/GenericPropertyOntology

3 Definitions, symbols and abbreviations

3.1 Definitions

None.

3.2 Symbols

None.

3.3 Abbreviations

SAREF	Smart Appliances REFerence ontology
SAREF4CITY	SAREF extension for the smart cities domain
SAREF4ENER	SAREF extension for energy domain
SAREF4ENVI	SAREF extension for the environment domain
SAREF4BLDG	SAREF extension for the building domain
SAREF4INMA	SAREF extension for the industry & manufacturing domain
SAREF4AGRI	SAREF extension for the smart agriculture and food chain domain
AIOTI	Alliance for Internet of Things Innovation
FOAF	Friend of A Friend ontology
CPSV	Core Public Service Vocabulary
STF	Specialist Task Forces
SEAS	Smart Energy Aware systems
ITEA	Information Technology for European Advancement
SDK	Software Development Kit
IRI	International Resource Identifier
OWL	Web Ontology Language

PEP Procedure Execution Ontology (a module of SEAS ontologies)

3.4 Namespace Prefixes

For the purposes of the present document, the [following] namespace prefixes [given in ... and the following] apply:

common	http://www.city-hub.kr/ontologies/2019/1/common# (namespace prefix for Common Ontology)
foaf	http://xmlns.com/foaf/0.1/ (namespace prefix for FOAF ontology)
pep	https://w3id.org/pep/ (namespace prefix for Procedure Execution Ontology)
saref	https://saref.etsi.org/core/ (namespace prefix for SAREF ontology)
s4city	https://saref.etsi.org/saref4city/ (namespaece prefix for SAREF4City ontology)
seas	https://w3id.org/seas/ (namespace prefix for SEAS ontology)

4 Conventions

The key words “Shall”, ”Shall not”, “May”, ”Need not”, “Should”, ”Should not” in this document are to be interpreted as described in the oneM2M Drafting Rules [i.1]

5 Existing SmartCity ontologies

5.1 Introduction

Smart Cities vision is vast as it aims to cover all the infrastructures involved which directly or indirectly affects its sustainability and development. Many different ontologies have developed, which cover different scope and use cases of smart city infrastructures including smart home, smart building, smart office, healthcare, transportation and mobility, governance, environment and energy, education, security, tourism, water management etc. Some of those work regarding defining ontologies, can be seen in Smart Applications REFerence (SAREF) and Smart Energy Aware Systems (SEAS) ontolotgies, focusing on the standardization of reference ontologies covering different domains which also coincide with the smart city domains. In addition, they also consider semantic interoperability, modularity and reusability as well as on providing common model of consensus, which are important aspects to achieve inter-domain interactions and synergy.The following study is focused on these ontologies from smart city perspective, even though they do not completely cover all the domains.

5.2 Smart City

5.2.1 SAREF4City

5.2.1.1 Introduction

The SAREF4City ontology is an extension of SAREF ontology [i.2] for the smart cities domain. This work has been performed in the context of STF(Special Task Force) 534 [i.3] under ETSI standardization which was established with the goal to create a set of extensions for different domains around a core ontology. The SAREF4City is the extension for smart city and it has been implemented considering the existing usecases and available data models relevant to this smart cities domain. In this regard, the development has been carried out with close collaboration with AIOTI [i.4], the H2020 Lighthouse projects on smart cities [i.5] and the ETSI activities in the smart cities. The ontology is available at SAREF4city Web [i.6].

The SAREF and SAREF4City are developed based on the standard ontologies such as GeoSPARQL [i.7], W3C Time ontology [i.8], FOAF [i.9] to enhance interoperability and consistency in the semantic modelling.

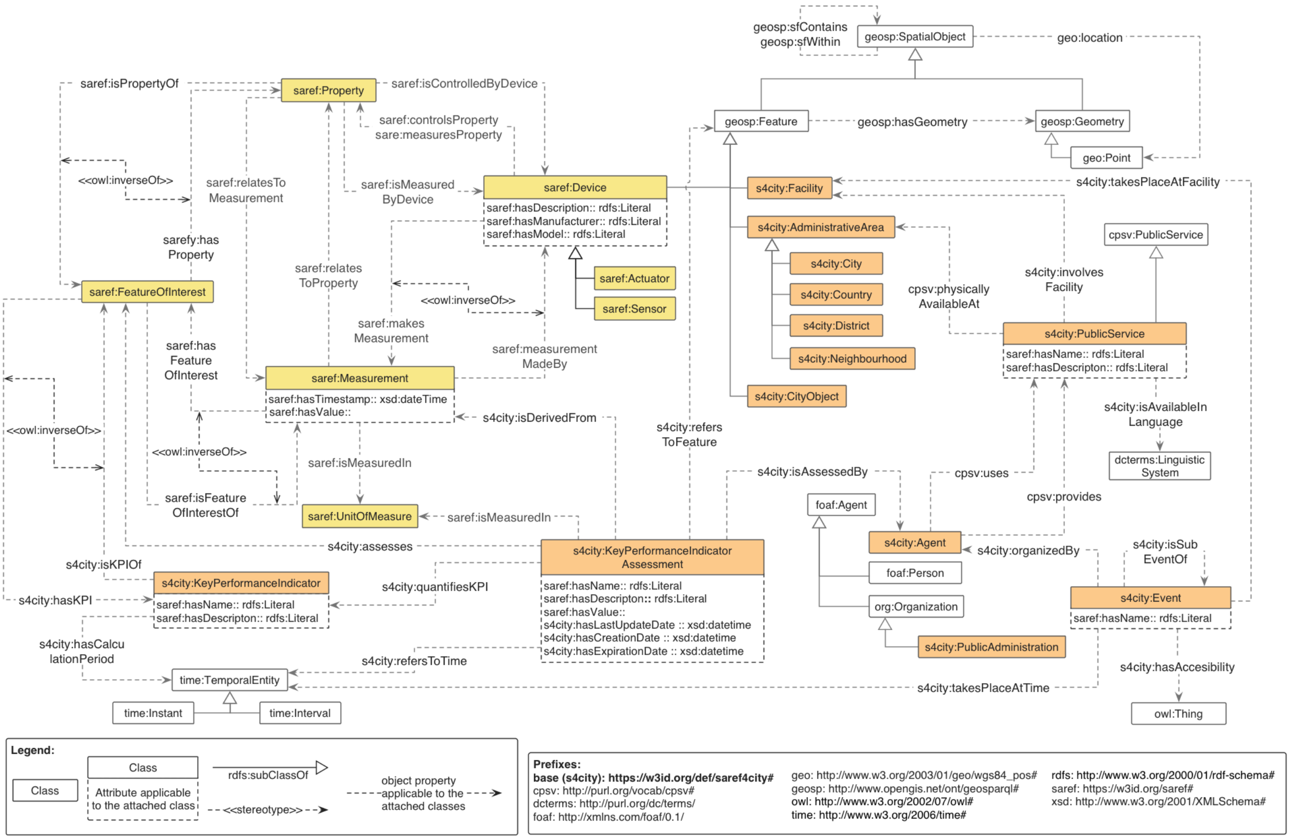


Figure 5.2.1.1-1 SAREF4City Extension [i.6]

5.2.1.2 Descirption

5.2.1.2.1 SAREF4City extension concepts

As shown in the Figure 5.2.1.1-1, the classes with prefix “saref:” represents the concept of SAREF ontology while prefix “s4city:”represents the SAREF4City extension concepts. It can be seen from the figure that the core of SAREF4City extensions involve regional (e.g. Administrative Area, Facility), economic (e.g. Public service, Event), and performance assessment (e.g. Key Performance Indicator, Key Performance Indicator Assessment) aspects.

Besides the concept classes, the relationships (object properties and data properties) have also been reused from existing ontologies. For instance, classes s4city:AdministrativeArea and s4city:PublicService are connected using the object property cpsv:physicallyAvailabeAt that is defined in CPSV ontology [i.10].

The definition of region concepts has been done using GeoSPARQL vocabulary concepts and WGS84 Geo Positioning vocabulary standard [i.11], which are geosp:SpatialObject, geosp:Feature, geosp:Geometry and geo:Point. The SAREF4City extension classes have been defined with rdfs:subClassOf property with the geosp:Feature class. These classes are s4city:AdministrativeArea, s4city:Facility and s4city:CityObject. Furthermore, the more detailed region concepts, such as s4city:City, s4city:Country, have been defined with rdfs:subClassOf property with the s4city:AdministrativeArea class. These concepts have been reused in order to define location properties in different aspects, such as defining location using coordinate system or as a regional area, having other sub-regions defined using geosp:sfWithin or geosp:sfContains.

The next main entity defined in the SAREF4City extension is the s4city:Event class, which links with the relevant classes such as s4city:Facility, s4city:AdministrativeArea, s4city:Agent. The concept definition of s4city:Event has been considered in a social perspective, rather than that of computing. For example a ceremony or an occasion which can take place at some neighbourhood on a given time, by any person or an organization. In such case, assertions can be defined using s4city:Neighbourhood, time:TemporalEntity and s4city:Agent class. However, s4city:Agent class has been generalized for both person and software.

In order to cover economics aspects of service provisioning, s4city:PublicService class has been defined, which has object property relationships with s4city:Agent and other classes related to region concepts. This class can highlight the characteristics of services as a commodity, where it can be offered by some s4city:Agent (e.g. government, person) can be defined using s4city:PublicService class.

Another main concepts defined in this extention includes s4city:KeyPerformanceIndicator and s4city: KeyPerformanceIndicatorAssessment classes. These enable describing performance evaluation of an organization or a relevant activity. This concept definition has been adopted from FIWARE data model specifications [i.12]. These concept classes also cover more of social aspect than that of ICT related systems and computing. However, s4city: KeyPerformanceIndicatorAssessment has relation s4city:isDerivedFrom with saref:Measurement class, which covers more general aspects of measurements relevant to the observations of real world phenomena. These classes are also linked with core SAREF concepts such as saref:FeatureOfInterest and saref:UnitOfMeasure, in order to enhance the expressivity of the key performance evaluation, as well as maintaining the consistency with the SAREF core ontology model. Some examples of these relationships are s4city:isKPIOf, s4city:hasKPI, s4city:assesses, s4city:IsMeasuredIn.

5.2.1.2.2 External Ontology Concept

Other external ontology concepts have been brought into the SAREF4City extension. These ontologies are GeoSPARQL, Dublin Core [i.13], FOAF, W3C Organization [i.14] and Time ontologies, as well as WGS84 Geo Positioning vocabulary and CPSV. The utilized concepts and their relationships with SAREF and SAREF4City can be seen in figure 5.2.1.1-1.

5.2.2 SEAS

5.2.2.1 Introduction

The Smart Energy Aware System (SEAS) [i.15] is the European ITEA project [i.16], which aimed at standardization of energy related information exchange methodologies as well as enhancing interoperability among IT systems. The deliverables of SEAS project were the smart energy API, standard and the SDK. The smart energy API, a semantic information model, connects user intelligently and transparently. The SEAS framework used open architecture to increase interoperability. This work has been examined by adopting in the more than 120 use cases in 6 different categories with 30 ontologies in energy domain. The knowledge model can be expanded for different domains: smart homes, micro grids, electric vehicles, electricity market, distribution and retail operators and clients, weather forecast.

5.2.2.2 Description

5.2.2.2.1 SEAS knowledge model design goal and considerations

The SEAS is based on the OWL 2 DL ontology[i.17]. The practicses, adopted for the development of the knowledge model in SEAS followed semantic web standards, which involved IRI look up, reusing existing ontologies by importing vocabularies and set of logical axioms, versioning of ontology modules, alignment with the existing external ontologies etc.

5.2.2.2.2 SEAS ontology modules relevant to smart city domain

As the knowledge model design principles in SEAS follows modularization, each module focuses on some specific use case. Therefore, a selective set of modules would be utilized based on considered use case for implementation. Since this document focuses on Smart City domain, selective set of modules will be discussed, which are relevant to the scope of this document. The modules and the import sequence can be visualised in figure 5.2.2.2.2-1.

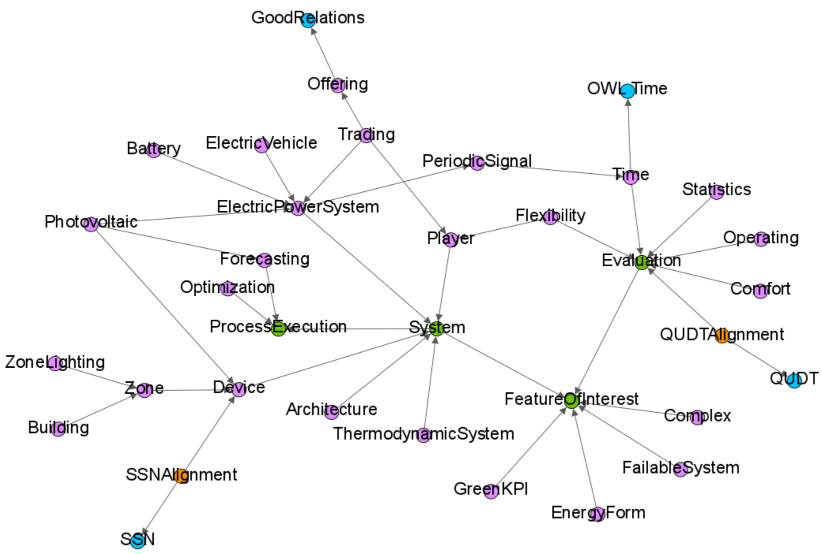


Figure 5.2.2.2.2-1: Core and extended modules[i.19]

In the figure 5.2.2.2.2-1, green nodes are core modules, pink nodes are extended modules and orange nodes are alignment modules respectively. The core of the ontology is composed of four modules:

- Feature Of Interest Ontology
- Evaluation Ontology
- System Ontology
- Process execution platform ontology

These modules define the generic concepts, which are required for defining the baseline of topology. On top of these core modules, the SEAS ontology defines several extended modules (referred to as "vertical" modules in SEAS documentation), that define specific knowledge model based on the core modules of a domain. These modules represent knowledge model for systems and their connections, their properties, devices involving sensors and actuators, region concepts (zone ontology), concept models for energy markets, demands/responses etc.

The module feature of interest ontology describes the considered single or set of feature of interest in the real world environment (e.g. a room, a server, a device, an organization) as well as their properties. These properties can be qualified, quantified, observed and/or based on the related feature of interest in the environment. One unique aspect in this module is that any property can also be defined as feature of interest, to support the idea that properties may also have some properties as the system evolves and new parameters are added in the environment. This module is not dependent on any other modules in the knowledge base.

The module evaluation ontology defines the concepts required for evaluating the properties of feature of interest, which are defined in feature of interest ontology. This module can be adopted to represent the qualitative or quantitative values, unique or constant values, statistical measurements, estimations or forecasts, temporal values and provenance information etc. One property defined in feature of interest ontology can have multiple evaluations defined using evaluation ontology. This module extends only feature of interest ontology as shown in figure 5.2.2.2.2-1.

The module system ontology describes the system knowledge model, which is focused on specific concept representation of feature of interest ontology, as it only imports feature of interest ontology. Here, a system is considered as a part of real world environment that can be virtually isolated. This modules can also be used to define the interactions of the multiple systems through concept representation of connections and connection points defined in it, such as connection, connectedTo, connectionPoint, etc.

The module Procedure Execution Platform Ontology [i.18] is defined an externalized core module, according to project deliverable D2.2 SEAS Knowledge Model document [i.19], as it generalizes the concepts defined in W3C Semantic Sensor Network ontology [i.20] and Semantic Actuator Network ontology. This module defines the concept representation of procedures and its executions. For example, "Sensing" can be defined as pep:Procedure whose execution can be defined as "Observation" using pep:ProcedureExecution, which will be executed by a "Sensor", defined using pep:ProcedureExecutor.

Addition to the core modules above, the SEAS ontology also defines extended ontology modules (reffered as SEAS vertical modules). The modules are divided into two categories based on the portability to different smart city use cases. The table 5.2.2.2.2-1 shows the categorization of these modules.

Table 5.2.2.2.2-1: Categorization of SEAS modules based on smart city Domain

use case independent (cross-cutting)	use case dependent (vertical)
building ontology	communication ontology
city ontology	electric vehicle ontology
device ontology	forecasting ontology
player ontology	street light system ontology
zone ontology	smart meter ontology
generic property ontology	trading ontology
offering ontology	zone lighting ontology
pricing property	
statistics ontology	
time ontology	

One of the essential information regarding smart city domain is the regional concept representation. For the semantic representation of such information, zone ontology, building ontology and city ontology are relevant modules. The module zone ontology defines the knowledge model for regions in more generalized terms, such that it can be used to define from small zones (e.g. room) to large ones (e.g. city). It also defines a sub-zone as well as connected zone relationships among different zones. This module imports system ontology and device ontology. The modules building ontology and city ontology can be considered as extended modules of zone ontology, hence they focuses on respective knowledge representations.

The module device ontology is essential in the aspect of knowledge representations for devices deployed in smart city. The module player ontology focuses on a person or an organization performing a business role. This role can be in terms of offering or execution of a service etc. This is essential module for knowledge representation of an entity, providing a service to users in smart city. Both device ontology and player ontology import system ontology and Procedure Execution Platform ontology, hence the concepts and relationships are extended accordingly.

The other use case independent modules considered for smart city, mostly define the properties and evaluation parameters of the feature of interest. Among these modules, time ontology can be most utilizable, as it will enable knowledge representation of temporal context of evalutations. The module statistics ontology is used to define the statistical evaluations of the properties for the considered feature of interest. The pricing module can be used in relation with player ontology and offering ontology, in order to represent the service provisioning and the respective pricing information. For other types of evaluations, generic property can be utilized, such as direction, length, speed etc.

For use case dependent modules, if a knowledge model specification involves a use case specific feature of interest the module can be utilized. For example, for electric vehicles electric vehicle ontology would be used.

6 Gap Analysis of existing ontology models for smart city domain

6.1 Introduction

This clause provides the detailed analysis of the limitations in existing ontologies for the smart city domain. The limitations are analysed using principles based on current best practices of ontology design.These principles involve the considerations for topology and structure, terminologies, granularity, logical criteria, etc. The issues highlighted in this study are mainly focused on the selected standard ontology models, considering only smart city domain. Regarding this analysis, scope limitations are identified and discussed to provide more reliable conclusions, considering a controlled environement, as an ontology model always remains the subject to be updated based on the open-world assumption [i.21].

6.2 Data Consideration for Smart City

The main goal of Smart City Ontologies is to enable semantic annotation, interoperability and reusability, for the data generated and exchanged in the smart city information systems. In this regard, the semantic support does not only considers devices, but different others as well. Smart City incorporates different IoT service provisioning, covering multiple domains, such as air-quality, weather, parking, public transportation, electricity, etc. In order to confine the scope of analysis, following aspects have been considered for Smart City domain:

2022) Device Aspect

The deployment of IoT devices in Smart City environment emphasizes the need to consider the device aspects in Ontology design and development. In this case, SSN [i.20] and other such standards have been referred, which specifically focused on Knowledge model representing IoT as well as non-IoT device concepts, for example, Input, Output, State, Function, Command, Variable, etc. This information are essential for the interworking with IoT applications. Using this knowledge model, Devices such as, sensors, actuators or any other computation capable IoT device can be represented and the semantically annotated information such as Function and Variable can be reused among devices and systems.

2) Service Aspect

Although oneM2M, SAREF and other IoT standards covers service aspects at architecture level including interfaces and APIs, there is still a need for the semantic representation of the information at higher level for the systems and platforms supporting such IoT network, especially in Smart City domain. For example, Weather and Air-Quality data acquisition through Weather station, Parking Service Provider's Profile, Parking Congestion Estimation, etc. These information cannot be annotated using above defined concepts. To overcome this limitation, standards such as SAREF and SEAS provide extended knowledge representation for the Smart City domain.

3) Non-IoT Data Aspects

The existing approaches focused on data acquisition and analysis, considering the IoT devices and other embedded systems. However, smart city covers much broader scope, which includes not only the IoT sensor data, but also requires concentration towards static, historical and other data aspects. These includes: statistical evaluations, profile information regarding person, organizations, places, events, contact information, pricing related data, census data, information regarding device and system models, their specifications, compatibility, etc. Hence the considered systems and environments should be able to support all this data representation schemes as well structuring them semantically, to support interoperability.

4) Modularization

Modularization is the key feature that is considered amongst almost all the IoT standards. The main advantage is the optimization in terms of data management, scalability, load balancing, query time, etc. For the efficient ontology management from both design and implementation aspects, we have considered modularization of ontology based on Information domains. The knowledge model has been designed considering two different aspects of representation. One is the Smart City common ontology model, which is the representation of generalized concepts and relationships, and second are the extensions of this model which represents specific domain information such as smart parking ontology, weather ontologym, air-quality ontology, etc.

6.3 Ontology Design Evaluation Criteria

6.3.1 Introduction

This clause describes the required criteria for the principle of ontology design, based on which the existing ontology models have been studied. These requirements are derived based on the best practices of ontology design[i.21, i.22, i.23]. Furthermore, these requirements have been categorized into two types of criteria: general evaluation and evaluation for granularity.

6.3.2 General evaluation criteria

General evaluation criteria involves principles, which are independent of application domains and must be validated in ontology design process.

- **Completeness:** This involves the essential requirement to provide sufficient conceptual representations that can cover the required domain. Although, this criteria will always be limited to existing reseach in the considered domain, it should be validated against existing knowledge repositories in each stage of ontology design process.
- **Adaptability:** The concepts of adaptability offers to achieve monotonic revision of ontology. In addition, it also supports to achieve the application goals of integration and scalability. In order to satisfy this criteria, the ontology should anticipate the considered tasks and should adhere to changes as well as extensions. The defined axioms should be stable enough to allow changes and addition of new ones.
- **Clarity:** This criteria incorporates different considerations in ontology design process. This involves clear and well defined concept classes and relationships representing the considered domain. Specifically, the definitions should be independent of subjectivity and context, should be documented and should be able to convey accurately, the meaning to its considered users.
- **Conciseness:** The ontology is considered concise if it does not include redundant and irrelevant axioms based on the considered domain. In addition, the relevant and essential axioms used for the concept representation, should be minimal.

6.3.3 Granularity based Evaluation criteria

In addition to the general evaluation criteria defined above, some requirement of particularity must be ascertain for ontology design evaluation. These are specified as follows.

- Logical criteria: that the ontology model should satisfy the basic logical constraints that are consistency and satisfiability. An ontology is inconsistent if it's axioms does not hold true in any possible world. However, a class may be consistent, but it will be unsatisfiable, if it is evaluated as an empty set in any model. In other words, if any instance is defined for that class, it will be evaluated as unsatisfiable.
- Structural criteria: This criteria validates the ontology structure as a topological arrangement of classes and relationships, evaluating the appropriate knowledge representation.
  - The ontology should contain a valid taxanomy. A taxanomy serves as the backbone of an ontology, where the concept classes form a hierarchy representing a directed acyclic graph with single root class. This hierarchy is developed using at least one of the two relations: Inheritance (is-a subsumption) and

composition (part-of subsumption). Generally, the hierarchy follows a direction from generic to specific concepts in inheritance and from aggregated to segregated concepts.

- The ontology relations should not demonstrate any ambiguity with respect to terminology and direction. In particular, the terminology should be able to distinguish between general and specific concept. In addition, the relationship terminology should also indicate the direction. For example; instead of “subSystem”, the relation should be termed as “subSystemOf”, because it indicates the direction from one class to another.

- The ontology concepts should not include multiple inheritance. In other words, the taxonomy should not involve any class with multiple parent classes. There are multiple factors considered in order to define this criteria. One is the computation performance, as the time and computation complexity will be reduced in inference as well as reasoning tasks by avoiding multiple inheritance. Second is the design maintenance and update, as the addition and removal of axioms will easier for the designer considering and ontology with single inheritance. Third factor is the consideration of integration and reuse. For example, complexity of ontology mapping, merging and alignment will reduce drastically using the ontology with single inheritance.

- Unique Identification: Each concept and relation in an ontology should be uniquely identified. This is one basic criteria which is essential for the application program, where the processes such as searching and crawling techniques are dependent on the predefined identifiers. In addition, the ontology itself should be associated with unique identifiers, for each version of the defined ontology.

- Role characteristics: The ontology should ascertain role characteristics such as disjointness, transitivity, functional and other such relations where necessary.

  - Disjointness is necessary in concepts where one instance can be represented by only one of two classes. For example: the instance “smart_phone” can only be defined under class “Device” and can never be defined in class “Human_User”. In this case, the property of disjointness should be defined between these two classes.

  - Transitive properties are used for defining sequence of relationship between one or more classes. For example, consider the owl:ObjectProperty “subZoneOf” defined as transitive, having both rdfs:domain and rdfs:range as “Zone” class. Then by defining the relationships “parking_spot subZoneOf parking_lot” and “parking_lot subZoneOf city_district”, the ontology user can infer the relation “parking_spot subZoneOf city_district”.

  - The functional property restricts the owl:ObjectProperty or owl:DatatypeProperty to have single unique relationship for any rdfs:domain associated with it. For example, each instance of class “Device” will have only one relationship “hasID” as owl:DatatypeProperty.

  - There are other role characteristics besides mentioned above, which ensures the completeness at granular level of an ontology. For further details, refer to [i.24].

- Univocity: The terminologies used in an ontology should ascertain univocity. That is, the term used to defined the concepts should communicate one unique meaning. In particular, the terms which are homographs (which have same spelling but different meaning), must include the exact definition to avoid ambiguity. For example, the term “lead” can be interperated as “to guide” or as “the metal”. Therefore appropriate meaning is required for the interpretation. For the case of different terms with same meanings, synonyms should be defined in annotations for the proper usage and to avoid any ambiguities in updating the ontology in future.

- Rigidity: The concept classes in an ontology should ascertain rigidity. For this criteria, the concept definitions must involve essential features without which its’ members can not exist. For example, an IoT device will have all its essential features such, being able to communication through Internet, can perform computation, etc. without which it can not be identified as an IoT device. In addition, it supports the property that a rigit class will always be a subclass of a rigid one. This property can be used to evaluate taxonomy appropriately.

- Singular terminology: The ontology concepts should be defined using singular terminology. Mostly the taxanomy defined in an ontology consists of either inheritance (is-a hierarchy) or composition (part-of hierarchy) of concepts. In such case, the representations are best representated when the defined terms are singular. For example, the relation “parking_lot is_a Zones” clearly shows ambiguous representation.

- Definitions for non-root terms: The ontology should include definitions for atleast all the non-root terms, where the definitions should involve the essential features and avoid circularity. The criteria for defining essential features has been discussed previously. A cicular definition will involve same term used in it’s representation.

For example, consider the following definition: "The area which represents Zone.". This will be a considered as a circular definition of class "Zone", because it does not stipulate additional information to describe it's nature or any of it's essential features. One recommendation to avoid circularity is to include the terminology from the parent or super class, followed by specifying the essential distinguishing features which defines it's existence. For example, the class "ParkingLot", which is defined as "The Zone, where cars or other vehicles are left temporarily", validates the above defined criteria.

The other aspects such as context, representation, modularization, reusability, realism, adherence to reality, etc. vary based on domain and ontology design approach, hence will be discussed according to the relevance.

6.4 Gap analysis of SAREF ontologies

6.4.1 Introduction

This clause provides the detailed analysis of the gaps highlighted in SAREF ontologies considering the smart city domain. Some concepts which are relevant in smart city domain, are not included as part of SAREF4City ontology, however, are defined in SAREF ontology. Therefore, those concept definitions have been considered for the evaluation. The discussion is organised into point-based highlighted gaps and categorized according to the general evaluation criteria defined in clause 6.3.2.

6.4.2 Gap analysis based on completeness

- The class saref:Service is defined as "a representation of a function to a network that makes the function discoverable, registerable, remotely controllable by other devices in the network". Here, the class covers only some part of computing aspect. There can be other services such as; the ones which are provided as a commodity. Also, in computing concept, there can be other services, which may not consider device aspects, for example, data management and analysis, data security, task monitoring and management, etc. Hence, the other aspects of services are not available, which may cover possible smart city concept representation.

6.4.3 Gap analysis based on adaptability

- SAREF provides relations with external ontology concept representations, which promotes reusability of existing ontologies. However, it lack the structure supporting taxonomy as well as open-ended ontology design process. See example A.1.1 for further details.

- The classes saref:FunctionRelated and saref:BuildingRelated though demonstrate a valid taxonomy, they may not be optimized in terms of usage and concept representations. For example, there will be many cases where a single device will be performing many functions, therefore it will be difficult to create concise assertions for them, using the current taxanomy.

- The axiom 'saref:DoorSwitch subClassOf(saref:consists of someValuesFrom saref:Switch)' may lead to inconsistent taxanomy or unsatifiable assertions due to the existence of axiom: 'saref:DoorSwitch subClassOf saref:Switch'.

6.4.4 Gap analysis based on clarity

- There are different concepts and relationships in SAREF ontologies, in which the terminologies used in their assertions, posses the nature of covering either wider or narrower scope, unlike their respective definitions. Therefore, the constraints applied for the usage of those assertions lack clarity, considering the terminology compared to their respective descriptions. In addition, these terminologies are preoccupied with existing scope limitations. This will increase the uncertainty as well as the complexity to search the assertions relevant to the required usage, as there will be multiple assertions defined in the future, having similar terminologies with different scope limitations. Examples A.1.2-A.1.9 highlight this specific issue.

- The class saref:State is defined as, "The state in which a device can be found …". First, this definition assert circularity. Secondly, this definition has limited scope as it is only focused on device aspect whereas other events can also have states to define their life cycle in the considered environment. For example, some process or system having state as running, paused, finished, queued, etc.

- The relation saref:isMeasuredIn highlights ambiguity as it includes saref:Commodity as its domain. However, entities such as service as a commodity can not be measured using saref:UnitOfMeasure class.
- The class saref:State should be defined as subClassOf saref:Property, as it should be considered as a property which deals with the state of any entity.
- In certain property definitions, some of the terminology used, highlights uncertainity in identifying their proper usage. See example A.1.10 for further details.

6.4.5 Gap analysis based on conciseness

Based on the current versions of the considered ontology modules, there are no assertions that are evaluated as redundant or irrelevant based on the criteria defined in clause 6.3.2.

6.5 Gap analysis of SEAS ontologies

This clause provides the detailed analysis of the gaps highlighted in SEAS ontologies considering the smart city domain. The following discussion considered the selected SEAS ontology modules that are highly relevant for the smart city domain. This includes all the core SEAS ontology modules as well as some of the vertical domain ontologies. The discussion is organised into point-based highlighted gaps and catagorized in accordance with the general evaluation criteria defined in clause 6.3.2.

6.5.1 Gap analysis based on completeness

- In seas:CityOntology module, there are some missing concept representations identified, with repsect to completing the scope of domain. For example, idustrial area, public area, neighbourhood, etc. Example A.2.1 provides details regarding these missing concepts.
- In seas:BuildingOntology, there are certain definitions, such as, those of seas:BuildingSpatialStructure, seas:BuildingSpace and seas:Ceiling, which do not provide enough details through which its usage can be ascertained. Refer to example A.2.2 for further details.
- Regarding the aspect of completeness in seas:DeviceOntology ontology, there are certain concepts whose representation is not available. Such concepts are discussed in example A.2.3.

6.5.2 Gap analysis based on adaptability

- seas:Property can some times also be declared as seas:FeatureOfInterest. This may create different limitations as well as complexities in terms of ontology extension. For example, if an entity requires representation using properties (i.e owl:ObjectProperty and owl:DataProperty) involving both seas:Property and seas:FeatureOfInterest, then two entities has to be created. In addition, seperate propery has to be defined, which represents the relationship between those two entities. Another complexity can be witnessed in deciding that which class should be used for representation, due to their dual nature of representation. In the class description (defined as a rdfs:comment) of seas:Property, it is well explained that how a seas:Property can be utilized as a seas:FeatureOfInterest. However, the example used in the description leads to another enigma regarding property subsumption, which will be discussed later in this clause. Besides, external applications can not identify this usage using any ontology definition other than the rdfs:comment defined for seas:Property. Hence an agent based application has to rely on Natural Language Processing in order to identify this comprehensive usage of seas:Property.

- seas:PercentageProperty that is subsumed by seas:Property, although is correctly defined based on univocity and rigidity, is too specific to representing the data aspect rather than representing the high level aspects of the properties. This may result in increased complexity in creating relationships between classes of domain seas:FeatureOfInterest, seas:Property and seas:Evaluation. Since it covers only the data aspect of represention, additional assertions will be needed to be defined and related to seas:PercentageProperty. Example A.2.4 discusses this clause in details.

- In the Zone ontology, the properties seas:absoluteHumidity, seas:populationFlow, seas:saturatedVapourPressure and seas:specificHumidity are defined as a sub-property of seas:hasProperty, in seas:FeatureOfInterestOntology module. The property seas:hasProperty has its domain and range defined as seas:FeatureOfInterest and seas:Property respectively. However, the above mentioned properties include both of their domains and ranges as sub-classes of seas:Property. This may result in ambiguity in defining assertions as all the individuals in the property assertion, involving the above mentioned properties, can not be related to each other using the property seas:hasProperty [i.25]. See example A.2.5 for details.

- Certain classes in the modules, seas:PlayerOntology, seas:CityOntology and seas:BuildingOntology, support multiple inheritance, which may cause a drawback for ontology extensions. These classes are seas:Player, seas:Bridge, seas:CarPark, seas:Stadium and class seas:Building.

6.5.3 Gap analysis based on clarity

- Based on the definition, the property seas:value, defined in seas:EvaluationOntology, should be renamed as seas:constantValue, in order to avoid preoccupation of the terms.

- Certain properties in the Procedure Execution Ontology (pep☺ module, have unspecified or restricted domains and ranges defined, in contrast of the terminologies used. These properties are pep:hasCommand, pep:hasInput, pep:hasOutput, pep:hasResult, pep:hasSimpleCommand, pep:hasSimpleResult, pep:implements, pep:made and pep:madeby. Example A.2.6 and A.2.7 highlight the particulars of this issue.

- Different SEAS ontologies have used the term “agent” and the class seas:Player in different assertions and descriptions. However there is a potential obscurity between these two terminologies and their usage in SEAS ontologies. The class seas:Player is defined as “One of the important people, companies etc involved in a particular industry, market, situation etc”. There are different SEAS ontology modules, such as seas:OfferingOntology, seas:FlexibilityOntology, seas:BuildingOntology, etc. where seas:Player has been used in object property assertions, while involving the term “agent” in their descriptions. Additionally, the term “agent” used in different places in SEAS, lack proper referencing, as it is important for the user application to identify the exact usage of the considered class. See example A.2.8 for further details.

- The definition for seas:CivilEngineeringWork accentuates subjectivity as it states that it should not be classified under buildings. In this case, there are two situations to be considered. First, this concept implies its existence more towards higher level ontology than seas:BuildingOntology module. Although its existence in seas:BuildingOntology is valid based on logical constraints, users may not expect such class in this ontology. Secondly, alghough it has provided different examples that can be considered as seas:CivilEngineeringWork, based on the open-world assumption, there can be infinite many concepts that may be considered under its definition since the it uses negation for describing it’s essential feature.

- The definition class seas:Garage in seas:BuildingOntology, and the classes seas:Authority, seas:ElectricityMarket and seas:SmartChargingProvider, in seas:PlayerOntology module, accentuates circularity.

- In seas:BuildingOntology module, the classes seas:Laundry, seas:LowEnergyHouse, seas:PassiveHouse, seas:Room and seas:Sauna, involve certain terms in their definitions, which highlights ambiguity in identifying their scope and usage. Example A.2.9 and A.2.10 expand on this discussion.

- In the seas:ZoneOntology, the terminologies used to define the properties seas:absoluteHumidity, seas:area, seas:humidity, seas:population, seas:populationFlow, seas:saturatedVapourPressure, seas:specificHumidity and seas:volume do not specify direction from their respective domains to ranges, which is not in accordance with the best practices of ontology design.

- The seas:CityOntology module definition is stated as: “The SEAS City ontology contains subclasses of zones usefull to describe cities”. However, it does not include the details that which aspect of features can be involved, for example infrastructure, administration, etc. Since the city domain requires extensive conceptual representation, therefore it is important for the user to realize the aspects which the ontology covers.

6.5.4 Gap analysis based on conciseness

Based on the current versions of the considered ontology modules, there are no assertions that are evaluated as redundant or irrelevant based on the criteria defined in clause 6.3.2.

6.6 Gap analysis summary

SAREF and SEAS ontologies and their extensions are aimed at supporting smart systems for different domains. Both standardization work mainly focused on semantic interoperability and reusability as well as on providing common model of consensus. The positive aspect of considering these ontologies is their high relevance towards smart city domain. SAREF has contributed by the development of SAREF4City ontology, specialized for smart city domain and SEAS ontologies involve different modules such as seas:CityOntology, seas:BuildingOntology, seas:PlayerOntology, etc. that support representation of smart city ecosystems. In this regard, these ontologies have been analysed based on the evaluation criteria defined in clause 6.3. Table 6.6-1 provides the overview of the highlighted gaps, identified in the considered ontologies discussed in clause 6.4 and clause 6.5, based on the general evaluation criteria defined in clause 6.3.2.

Table 6.6-1. General Citera Evaluation for SAREF/SAREF4City and SEAS

	SAREF / SAREF4City	SEAS
Completeness	Missing Concept Representations	Missing Concept Representations
Adaptability	Segregated taxonomy, inconsistency, unoptimized subsumption	Lack of univocity, unoptimized subsumption, improper domain/range definitions in the subsumption of object property, multiple inheritance.
Clarity	Terminology with limited scope, terminology with uncertain meaning and usage in ontology, circularity.	Terminology with contra unclear scope definitions, lack of univocity, circularity, uncertain concept definition and usage based on taxonomy, subjectivity
Conciseness	-	-

Table 6.6-2 provides the overview of the highlighted gaps, identified in the considered ontologies discussed in clause 6.4 and clause 6.5, based on the granular level evaluation criteria defined in clause 6.3.3.

Table 6.6-2. Granularity based Criteria Evaluation for SAREF/SAREF4City and SEAS

	SAREF / SAREF4City	SEAS
Logical Criteria	Axioms with potential unsatisfiable assertions	owl:objectProperties with potential unsatisfiable assertions
Structural Criteria	Segregated taxonomy, unclear class subsumption	Multiple inheritance, misleading class position in taxonomy with respect to its definition
Unique Identification	-	-
Role Characteristics	-	-
Univocity	Class definition with less optimized usage	Class definitions with similar scope and lack of uniquely identified features

Rigidity	-	Definitions with lack of essential features
Non-singular Terminology	-	-
Definitions for non-root terms	Circular definitions	Circular definitions

7 Ontologies for SmartCity in oneM2M

7.1 Introduction

This clause presents different ontologies, which all together can be considered potential set of ontologies for smart city domain. Due to limitations identified in the ontologies discussed in clause 6, they can not be used directly, to semantically represent the smart city data. In order to support modularization as well as to cover different sub-domains which can potentially be the part of smart city, a common ontology has been proposed, which covers the high level concept definitions. Using those concept definitions, this ontology has been extended by different domain ontologies which semantically represent those sub-domains in smart city. Refer to Annex B for the complete list of definitions of all the concepts.

7.2 Common Ontology

The core ontology, which is termed as common ontology, revolves around six main high level concept classes, which are shown in figure 7.2-1. The oval represents an owl:Class (and will be refered to as class / concept class) and the arrow indicates rdfs:subClassOf relationship, from child to parent class. These classes are considered from Feature of Interest Ontology [i.26], Evaluation Ontology [i.27] and Procedure Execution Ontology [i.28]. The base building block of this common ontology is class common:FeatureOfInterest and common:Property . Likewise Feature of Interest Ontology, these two classes in the common ontology have similar definitions, however, in any case, a common:Property can not be considered as common:FeatureOfInterest. This will ensure consistency and minimized ambiguity in future extensions. Refer to Annex B.1 for the specific definitions of each class.

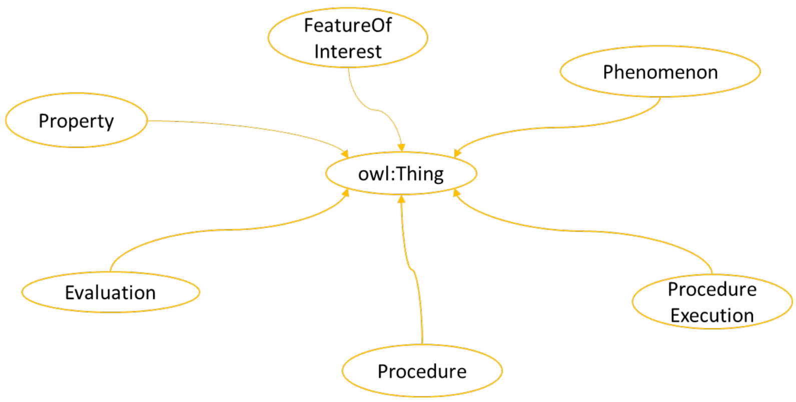


Figure 7.2-1. High level classes in Common Ontology

7.2.1 High level Classes

This clause discusses regarding the high level classes, the main aspects of smart city, which they cover and the major differences from the conceps in SAREF and SEAS ontologies.

7.2.1.1 Feature of Interest, Property and Evaluation

Following points highlight the main aspects covered by each class as well as the key differences to counterbalance the limitations, from smart cities' point of view.

- The class common:FeatureOfInterest is different than the one defined in SEAS as the characteristics of common:FetureOfInterest defining it's existence, remains same irrespective of time. But in SEAS it might change if the properties which define its characteristics change with time. However it doesn't mean that an instance of common:FeaturOfInterest will never cease to exist. It can cease to exist by being destroyed in some phenomena or an event, and then another common:FeatureOfInterest having same or different sub type may get recreated.

- In SEAS seas:Property can also be considered as seas:FeatureOfInterest but Common ontology emphasizes on clear distinction between the two. The existence and definition of a common:FeatureOfInterest remains consistent irrespective of time and the properties characterising it. Whereas the existence of a common:Property depends on the existence of some common:FeatureOfInterest. A common:Property can not exist without being linked to any common:FeatureOfInterest.
- The conceptualization can be further elaborated using common:Evaluation class, where the properties of a common:FeatureOfInterest can be enriched with in depth analytical or statistical assessments. Hence the common:Property class complements common:FeatureOfInterest in terms of characterization and the common:Evaluation class complements common:Property in terms of their analysis and assessments, however, each contains distinct representation of concepts.
- Based on the terminology, the common:Evaluation class here covers the broader domain than the one defined in SEAS. In common ontology, common:Evaluation provides the assessmen of common:Property, instead of the value of a common:Property. Although, there exists a owl:ObjectProperty in SEAS, relating common:Property and common:Evaluation. However, the main concern is making the scope of common:Evaluation broader, based on the terminology.

7.2.1.2 Procedure and Procedure Execution

The concepts of common:Procedure and common:ProcedureExecution are almost same as the ones defined in SEAS ontology. The slight difference of conceptualization is the way seas:Procedure defined in SEAS, compared to the one in Common ontology. In SEAS, the definition involves the following statement: “It explains the steps to be carried out to arrive at reproducible results”. However, in Common ontology, the results are not necessesarily reproducible. One such example can be the one involving stochastic process. In addition, the specific terminologies have been replaced by the phrase “series of steps or actions”, in order to provide a possibility to extend the scope.

7.2.1.3 Phenomenon and Observation

The reason of discussing these two concepts is the similarities in the definitions of common:Phenomenon and common:Observation in SEAS ontology. Following points elaborate the key differences.

- The seas:Phenomenon is defined as “A phenomenon is something that can be observed.” From the terminology’s perspective, seas:Phenomenon and seas:Observation both can be considered same as seas:Observation can also be considered as “something that can be observed”. Although, the definition of seas:Observation in SEAS Device Ontology, makes itself distinct from seas:Phenomenon as there, the seas:Observation is defined as “the execution of some sensing procedure by some sensor”, it can not be considered wrong if some concept, related to seas:Observation is described as a subclass of seas:Phenomenon. Therefore, to have a clear distinction from common:Observation, common:Phenomenon is defined as “fact or a situation” and “natural or artificial”. Whenever a concept related to common:Observation is defined as a subclass of common:Phenomenon, it can be regarded as natural or an uncontrolable occurance or event, instead of a procedure execution, which can be completely controlled by a procedure executor.
- The scope of seas:Observation is too constrained based on the terminology in SEAS, as it is stated as “An observation is the execution of some sensing procedure by some sensor”. In Common ontology, its scope is broadened by not specifying it as only to be sensed by a sensor, rather it is stated as “ProcedureExecution of monitoring, inspection, examination or recording of some information”. An example can be an observation from a servey data which is collected by user feedback. Such concept can be complex to defined as an seas:Observation in SEAS ontology as the device layer involved in the process is obscure, however, a concept of procedure executor can be defined easily, which is more generic than a concept of device.

7.2.2 Classes Hierarchies

Different sub classes are further defined to further project the scopes of the higher level classes. Based on the high level classes, six different hierarchies are discussed and differentiated in terms of their features, restrictions and relationships with other classes.

7.2.2.1 Feature of Interest Hierarchy

Figure 7.2-2 shows the expansion of common:FeatureOfInterest class. Based on the Smart city domain, common:FeatureOfInterest involves entities which represent systems, connections and such related concepts. The class common:System cover broader scope as it is based on System Theory, and it is different from the definitions of SEAS and SAREF from following aspects.

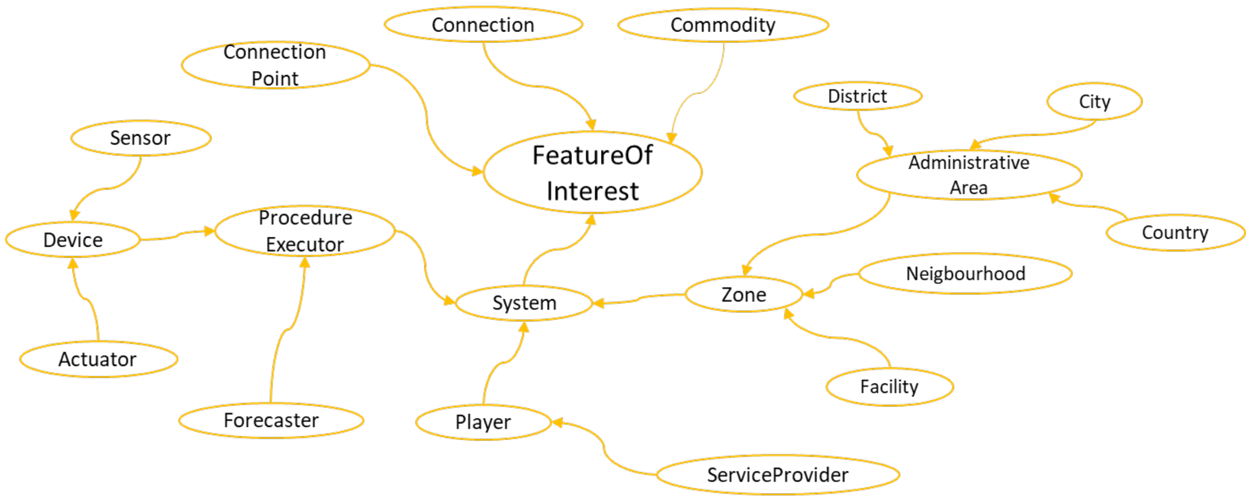


Figure 7.2.2.1-1: Taxonomy expansion under Feature of Interest

There are some important differences in the taxonomy which are described as follows:

- In SEAS and SAREF definitions, a system is defined as "class of systems virtually isolated from the environment". Here the scope of isolation becomes unclear. In common ontology, a common:System is not considered isolated in any way, rather it is "described by its boundaries, structure and purpose", so that it can even be considered as part of the environment, virtually or physically. The Boundaries, structure or purpose can be defined in terms of entities, working as its part, physical boundaries such as borders on a land, types of tasks performed (roles), etc. Furthermore, a common:System can be influenced by the environment in terms of interations or events of any sort.

- In SAREF, for environment ontology extension, there is another concept using same terminology "System", but defined from computer science's perspective. The common:System serves a much more generic perspective and systems focusing on some domains like computer science, can be extended from this class, hence ensuring sound and consistent basis of the concept hierarchy.

- The class common:System is defined as "a group of iteracting or interrelated elements", (also refered to as sub-systems) which as united, represent the existence of a system. This leads to the concept of sub-systems lying underneath. To represent such concept, the object property common:hasSubSystem has been defined, linking the two systems, where one (sub-system) is involved in the composition (among other sub-systems) of the other common:System. Similarly, the property common:subSystemOf links a sub-system to its super-system. Therefore, these relationships can be used to link systems within systems, in order to define the system hierarchy. Similar properties are also defined in SEAS and SAREF ontologies.

- Connections play vital role in composing the systems and sub-systems. While common:hasSubsystem and common:subSystemOf link the systems vertically in the hierarchy, common:Connection and the related properties can be used to link any common:System, in order to define their interaction with the systems outside or even within the environment (among sub-systems). Figure 7.2.2.1-2 and 7.2.2.1-3 describes the ways in which systems can be connected with each other by using common:Connection and common:ConnectionPoint respectively. The dashed line represents the linke between two classes and the label followed by an arrow, represents the owl:ObjectProperty and its direction respectively, from rdfs:domain to rdfs:range. The distinguishing feature of common:Connection in common ontology here is that it can exist without being connected to any common:System. Some example can be pending connection between two nodes in the network and a road leading to a dead end.

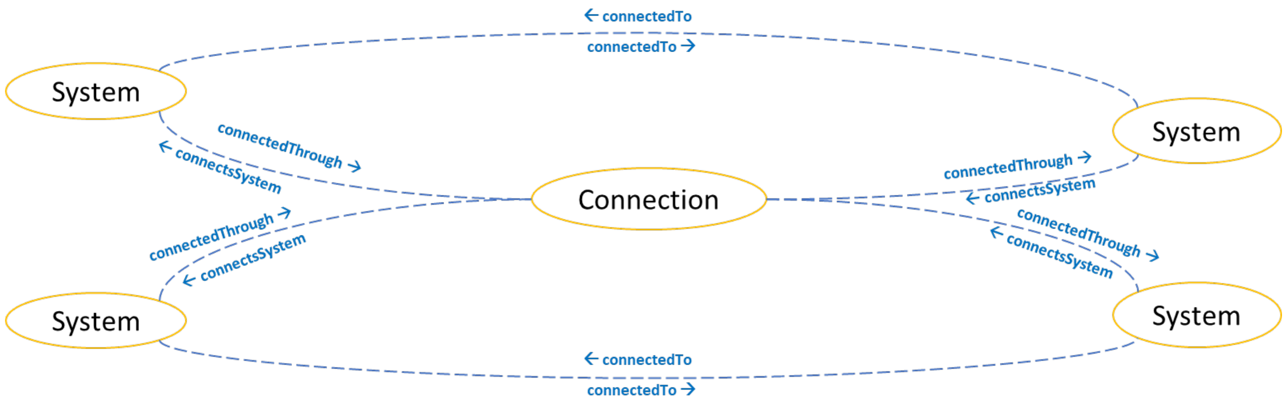


Figure 7.2.2.1-2: Connectivity among Systems using Connection

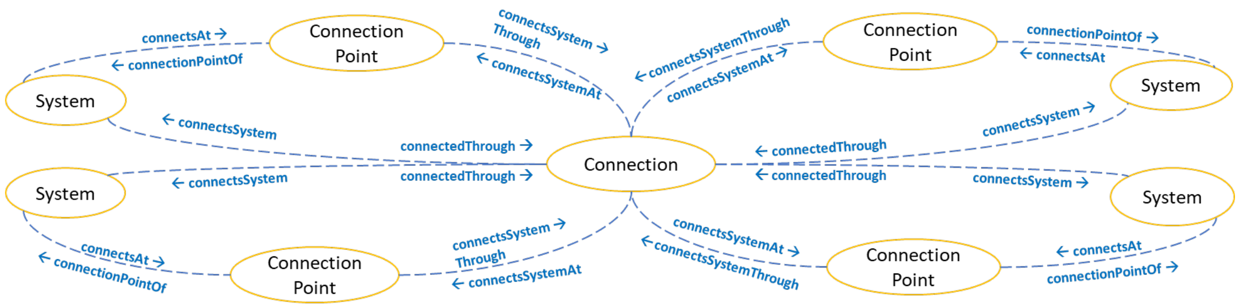


Figure 7.2.2.1-3: Connectivity among Systems using Connection and Connection Point

- The class common:ConnectionPoint can be used to define complex network of systems. This class has slightly different definition from the seas:ConnecitonPoint. In SEAS, a seas:ConnectionPoint belongs to or is dedicated to exactly one seas:System. Whereas in common ontology, a common:ConnectionPoint can be allowed to be an independent entity, that is, it can exist without a common:System. In this regard the properties common:connectsAt and common:connectionPointOf play a slightly different role in Common ontology. A common:System connected to a common:Connection point using common:connectsAt doesnot mean that the connection point belongs or bound to that specific system. Whereas, a common:ConnectionPoint connected to a common:System using common:connectionPointOf indicates its ownership by the common:System. This means, a common:ConnectionPoint can exist without being linked to a common:System. In addition, the property common:connectsAt indicates that common:ConnectionPoint is linked to some common:Connection.

- Based on the previous discussion, there can be many different types of common:System, as it covers broad scope. The subclasses of common:System, defined for the smart city domain, can be categorized into three sub domains: Computing domain, geographical and infrastructural domain, and Economical domain as highlighted in Figure 7.2.2.1-4. These three sub-domains can be discussed by describing their respective top level class which are common:ProcedureExecutor, common:Zone and common:Player.

- The class common:ProcedureExecutor has similar definition to pep:ProcedureExecutor in SEAS Procedure Execution Ontology. The main difference is it's position in the class hierarchy. In SEAS, pep:ProcedureExecutor does not have any parent class, whereas common:ProcedureExecutor is the subclass of common:System. This is possible because of vast scope that the common:System is covering. According to conceptualization in common ontology, a common:Device, common:Sensor, common:Actuator, common:Forcaster etc. is considered as such a common:System which executes a common:Procedure. For example, a device can involve multiple sensors and actuators, each executing their respective sensing or actuating procedure. In this case they should be considered of type common:ProcedureExecutor. However, in terms of System Theory, these sensors and actuators should be considered as sub-systems of the device. This kind of conceptualization can be well established using Common ontology. In addition, the definition is modified to the following: "The System involved in or implementing a Procedure".

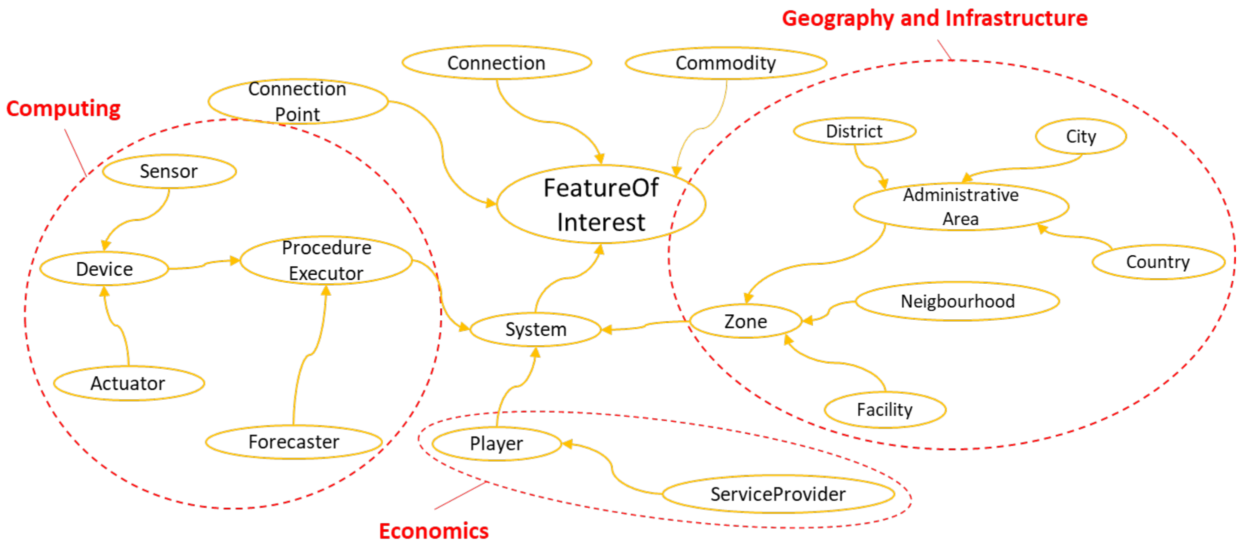


Figure 7.2.2.1-4: Categorization of System hierarchy based on domain scope

- The class common:Zone defines a common:System in terms of geographical and infrastructural characteristics. In SEAS Zone ontology, the class seas:Zone is defined as "A part or a section of a building, campus, town, etc.". However, it does not contain a comprehensive list of examples to clearify it's scope. For examples the concepts like no-fly zone or quarantine zone may or may not be defined as the sub-clasees. In order to realize this, other concepts and properties need to be analyzed. In Common ontology, a common:Zone is defined as an area or a stretch of land having some characteristics, purpose or restrictions. This clearifies the scope definition of common:Zone as different concepts can be represented not only from infrastructure's perspective, but also covering some aspects which involve geometrical, spatial or logical feature, but cannot be the part of an infrastructure like cities neibourhood etc. The concepts represented by the sub-classes of common:Zone, like common:AdministrativeArea, common:City, etc. are also defined in SAREF4City ontology, however, they are directly extendent from geosp:Feature.

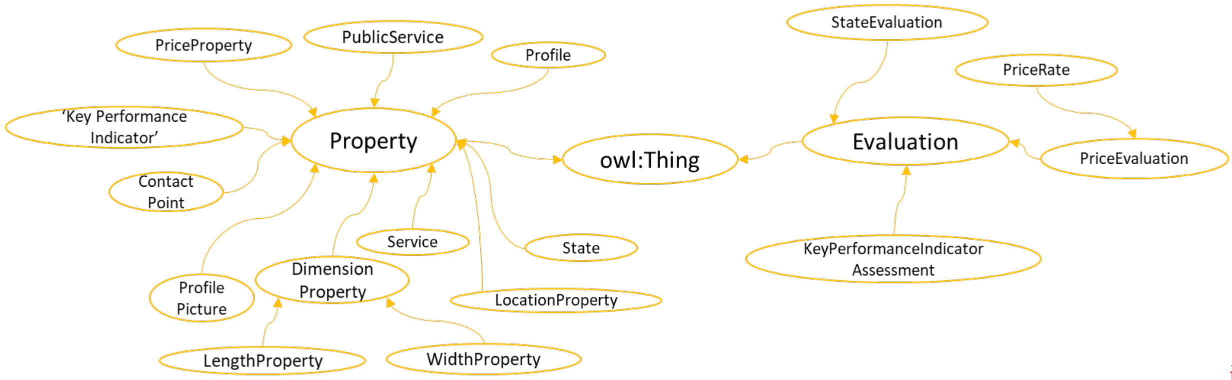
- One of the top classes in the economical domain the common:Player. The major variation from seas:Player class that the parent class hierarchy. In SEAS ontology, seas:Player has two parent classes, seas:System and pep:ProcedureExecutor. Having seas:System as the parent class is taxonomically explainable, as it can involve devices, people, companies, etc. However, considering this from a economical point, there are very rare cases where pep:ProcedureExecutor can be considered both as an entity which takes part in trade or stock market as well as executes a procedure. Nonetheless, in Common ontology, both common:ProcedureExecutor and common:Player are defined as a sub-class of common:System, whereas, common:ProcedureExecutor expands on a separate branch, covering Computing scope.

- The common:Commodity class has similar concept definition to seas:Commodity in SEAS Trading ontology and saref:Commodity in SAREF core ontology. However, in SEAS it is defined as a subclass of seas:Property. Since a commodity can exist as an independent entity and can have it's own definitive characteristics, therefore, in common ontology, it is defined as a sub-class of common:FeatureOfInterest.

- The class common:AdministrativeArea is a class considered from SAREF4City ontology. This also affects all the subclasses respectively.

7.2.2.2 Property and Evaluation Hierarchy

In the existing hierarchy, class common:Property and common:Evaluation cover concept representation from different domains including computing, geometry and geography, economics, and other generic concepts which can be used in multiple domains. These concepts are not defined as being focused to describing a specific common:FeatureOfInterest, rather they can be used to cover different aspects of characterization. For example, the common:ContactPoint can de email address or a phone number as a contact information, whereas, in a different scenario, it can include information like ip address as the information to communicate with some device.



inclu

Figure 7.2.2.2-1: Taxonomy expansion under Property and Evaluation

The expansion of common:Property and common:Evaluation class can be seen in figure 7.2-3. The aspects covered and modified are described here as follows:

- The classes common:DimensionProperty and common:LocationProperty are newly added classes. They serve the purpose of maintaining the taxonomy in case if additional sub-classes are added.
- The class common:State is considered from SAREF ontology. Its taxonomy is modified from owl:Thing to common:Property.
- The class common:KeyPerformannceIndicator and common:KeyPerformannceIndicatorAssessment are considered from SAREF4City ontology. Here, they are defined as subclass of common:Property and common:Evaluation respectively.
- Service can be one of the most critical concept specifically in this taxonomy. In SEAS Technical System Ontology [i.33], it is defined as a subclass of seas:AbstractEntity. In order to clearify it's scope to the level of Software Architecture domain, common:Service is defined as a rdfs:subClassOf common:Property. The other potential candidate classes, that can be defined as it's parent class, are common:Procedure and common:ProcedureExecution. The reason common:Service can not be defined as a subclass of either of them is thaI it cIlude both aspects of common:Procedure and common:ProcedureExecution in its composition as well as some of the types of common:System. Whereas here, it can be considered viable, if it is seen as a characteristic of a common:System. Now it's composition can involve relationships with different entities including common:System, common:Procedure and common:ProcedureExecution. Table 7.2.2.2-1 shows the list of candidate parent classes in order to determine whether the parent class can completely represent common:Service as it's subclass by their defining features.

Table 7.2.2.2-1 Candidate parent classes for common:Service

Class/Concept	Defining Features	Matched	Description
FeatureOfInterest	Abstraction of real world phenomena	No	The processes and executions involved are not real world phenomena, rather simulated.
FeatureOfInterest	Who's definition remains same irrespective of time.	No	There is no specific restriction on a service to remain same irrespective of time.
Property	Observable or operable characteristics of FeatureOfInterest	Yes	A FeatureOfInterest can execute or monitor a service, through which it can interact with other FeatureOfInterest.
Procedure	A reusable series of steps or actions.	Not Always	Service can also involve other aspects such as act of carrying out executions.
Procedure	Steps or actions, carried out to achieve results.	Not Always	Service can also involve other aspects such as act of carrying out executions.
ProcedureExecution	Act of carrying out a procedure.	Not Always	Service can also involve other information like procedures, policies, etc. which is not covered in this scope.
Functiona	Functionality to accomplish a task	Not Always	Service can also involve other information like procedures, policies, etc. which is not covered in this scope.

Task	Steps towards a specific goal.	Not Always	Service can also involve other aspects such as act of carrying out executions.

7.2.2.3 Procedure and Procedure Execution Hierarchy

Figure 7.2-4 ilustrates the taxonomy expansion under the classes common:Procedure and common:ProcedureExecution. Following are the covered aspects along with the modifications from existing ontologies:

- The classes: common:Actuation, common:Observation, common:Forecast and common:Forecasting have been considered from SEAS Device Ontology and Forecasting Ontology with same concept definitions.
- The classes: common:Estimation, common:Estimating and common:Observing are newly added, which cover some concepts related to ones mentioned above.
- The classes common:Task and common:Function are considered from SAREF with slight modification to broaden their scope. In SAREF ontology, both classes defined their relationship (in class definition) with a common:Device. However, in case of Common ontology, the relationship is defined with common:ProcedureExecutor, which means that any physical or logical entity capable of executing a common:Procedure can be related with these classes.

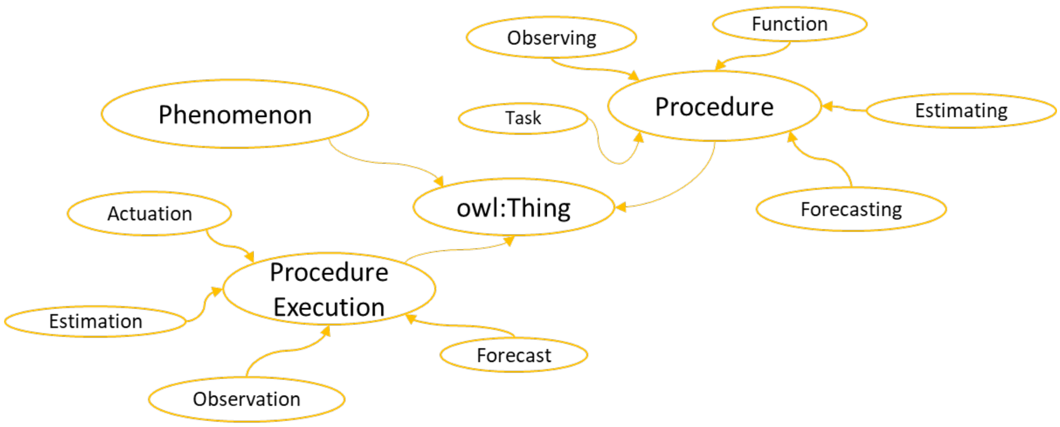


Figure 7.2.2.3-1 Taxonomy expansion under Procedure and ProcedureExecution

7.3 Smart Parking Extension

This clause describes the extension of Common Ontology for Smart Parking domain, based on following usecase. A parking lot is situated at some location in an urban area, which contains different parking spots. The Parking lot is described with general profile information such as id, contact point, geo location, address, parking service price rate etc. In addition, it involves some status information regarding the parking spot availability. Besides, each parking spot may also contains its individual profile information and have its status update, which is then accumulated to update the status of its respective parking lot. In this case, the services, in terms of both software architecture and commodity are involved. The service as a commodity includes parking service provided to the customer, which also charged with the parking service fee. The services in terms of software architecture includes one, which evaluates the available parking spots, and the other provides the estimated parking congestion.

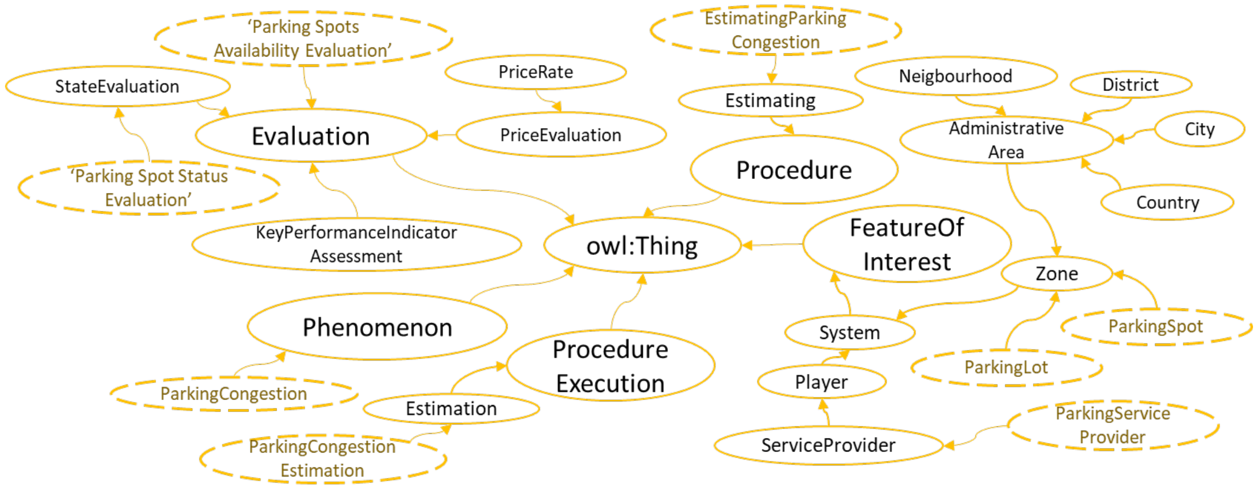


Figure 7.3-1 Extension of Common Ontology for Parking domain based on Feature of Interest, Procedure, Procedure Execution, Phenomenon and Evaluation

Figure 7.3-1 and 7.3-2 shows the extension of Common ontology for Smart Parking domain. Here the dotted oval represents the owl:Class for parking domain. Here classes parking:ParkingLot and common:ParkingSpot are described as subclasses of common:Zone, which also enables them to be represented as both local space and as a common:System. Profile related information can be stored in parking:ParkingLotProfile and parking:ParkingSpotProfile class. The class parking:ParkingSpotStatus in figure 7.3-2 and the class parking:ParkingSpotStatusEvaluation in figure 7.3-1 relates to the information about a particular parking spot, such that altogether, they provide the information of its availability at particular time stamp. Similarly, the class parking:AvailableParkingSpots in figure 7.3-2 and the class parking:ParkingSpotsAvailabilityEvaluation in figure 7.3-1relates to the information about parking lot, such that, the overall parking spots available in that particular parking lot will be represented here. All this information can be calculated by the instance of class parking:ParkingAvailabilityEvaluationService. In case of parking:ParkingCongestion, the instance of class parking:ParkingCongestionEstimationService will be responsible for execution of procedure of class parking:EstimatingParkingCongestion, and it's execution related information is represted by the instance of class parking:ParkingCongestionEstimation.

There are some classes in both figures, which are not included in common ontology. These classes are ServiceProvider and PriceRate. This is due to the reason that they are more suitable to be the part of some extention than Common ontology. They are considered to be the part of future extensions of ontology.

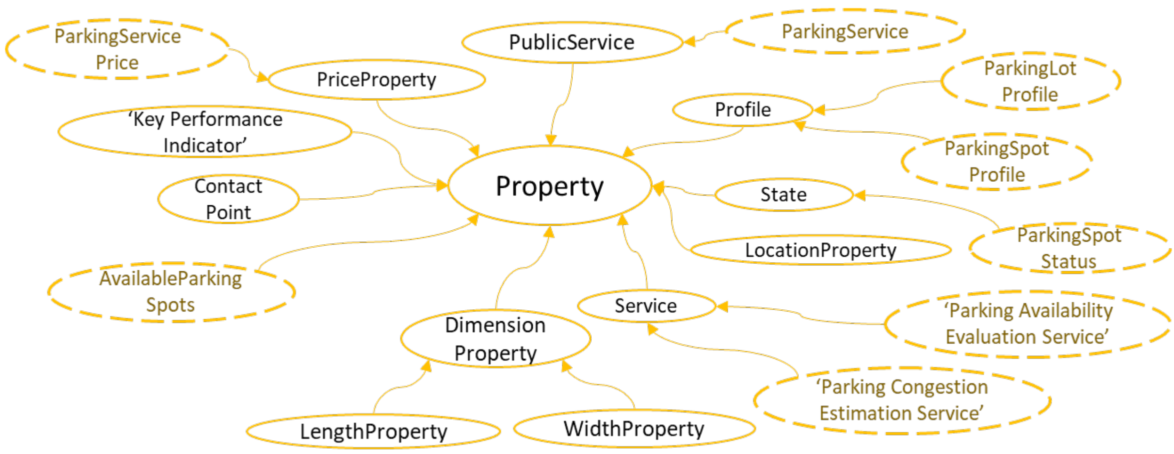


Figure 7.3-2: Extension of Common Ontology for Parking domain based on Property

7.4 Weather Extension

Figure 7.4-1 shows the extension of Common ontology for Weather domain. In this case the ontology covers mostly data aspect, when entiy responsible for data acquisition is a service rather than a device. Here weather is further defined using six different properties: air temperature, humidity, rainfall, wind, snowfall and visibility. Each of these has its own class of type common:Evaluation and common:Phenomenon. The classes representing the service responsible for data observation and estimation are weather:WeatherObservationService and weather:WeatherEstimationService respectively. Similarly, there are sublasses defined to represent procedures and their executions for each of two classes of type common:Service. This ontology can be compared with SEAS Weather Ontology [i.35] where some similarity can be witnessed due to the weather properties considered.

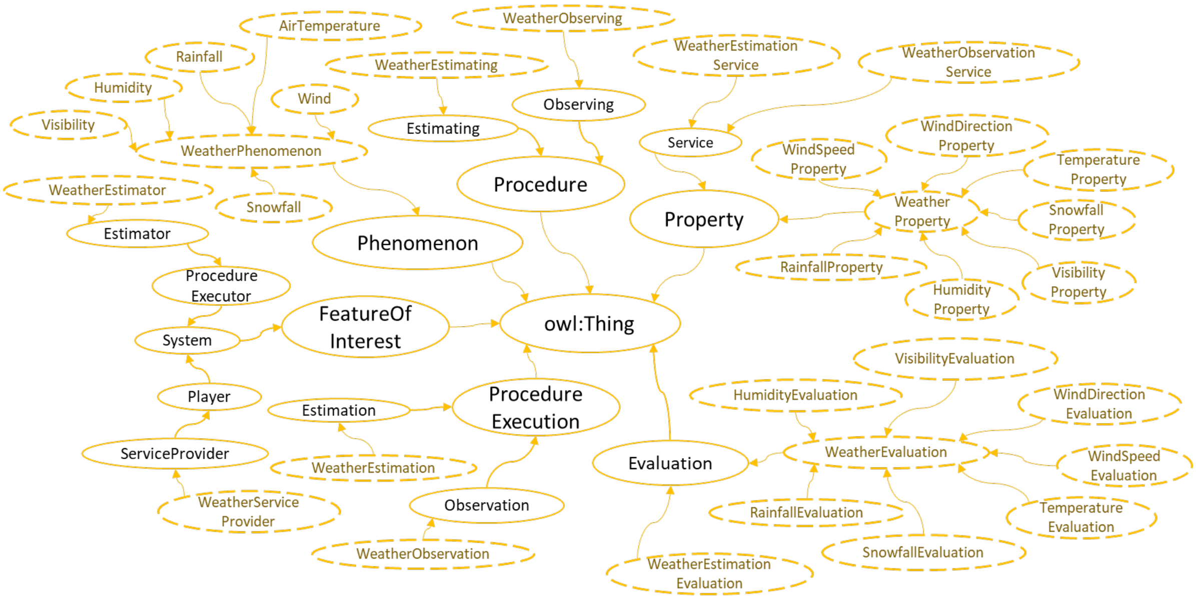


Figure 7.4-1: Weather Extension

7.5 Air-Quality Extension

Figure 7.5-1 shows the extension of Common ontology for Air-Quality domain. The structure is almost similar to that of Weather extension. The main difference is the properties considered for the air-quality data which are: air NO2 level, air O3 level, air CO Level, air SO2 level, air PM25 Level, air PM10 Level. This ontology can be compared with SEAS Generic Property Ontology [i.36] where some similar classes will be witnessed due the considered air-quality properties.

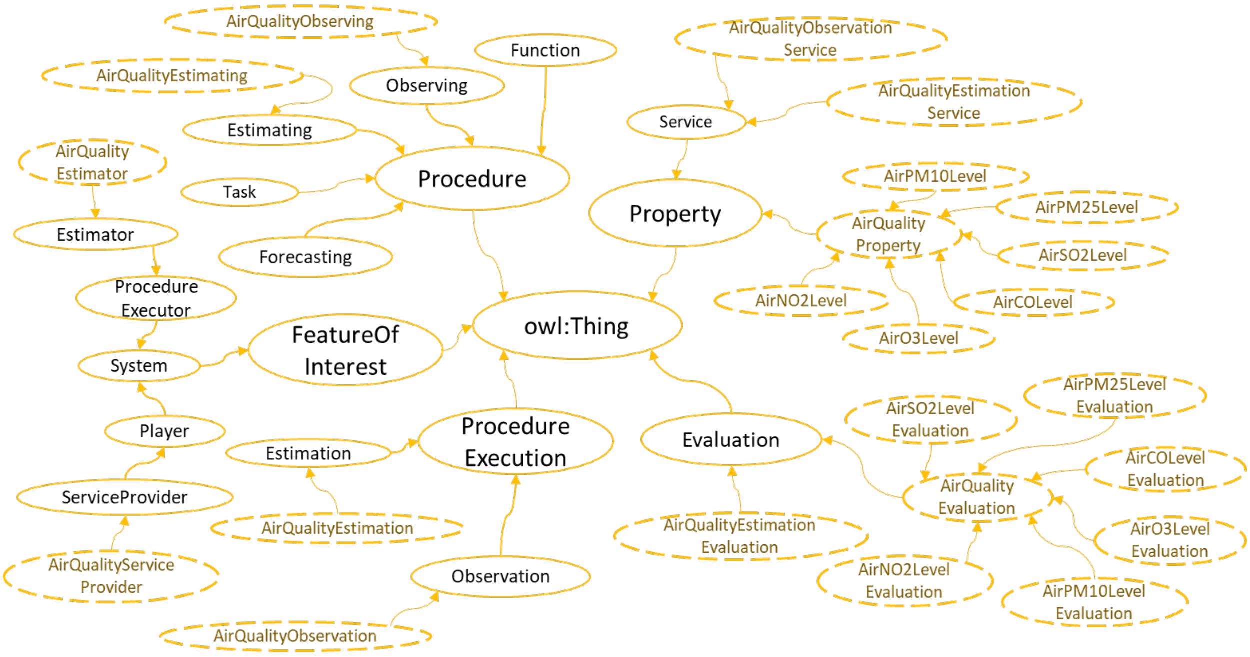


Figure 7.5-1 Air-Quality Extention

8 Conclusion

This technical report is the outcome of in-depth study and analysis of the existing ontologies, which support or enhance the smart city domain. In order to identify the gaps among these existing ontologies, different evaluation criteria have been defined, for the smart-city scope. The evaluation criteria have been realized a major resource in discussing the identified gaps in the existing ontologies. To minimize the identified gaps, a smart city core ontology has been defined, by adapting, integrating and consolidating the concepts of existing ontologies, and further, it has been extended by three domains namely: smart parking, air-quality, and weather. These extended domain ontologies demonstrate that by refining the core ontology, its extension becomes much easier, covering different aspects such device, service, non-IoT data and modularization. Although this smart city core ontology is the outcome of further enhancing existing well-defined ontologies, it is prone to modifications to interoperate with domains other than smart city, supporting the open-world assumption. Such enhancements will be supported in the future, considering this report as the basis.

Annex A: Examples of highlighted gaps

Examples of highlighted gaps, for the considered ontologies (SAREF and SEAS ontologies), are as follows.

A.1 Examples of highlighted gaps in SAREF ontologies

- **Example A.1.1:** In SAREF ontology, the class saref4city:KeyPerformanceIndicator and saref4city:KeyPerformanceIndicatorAssessment can be categorized as Evaluations. But the existing structure defines these classes as separate unique concepts without any more generalized class definition that may be used for the entailment. Reusing or integraging them with other concepts, such as statistical measures or any other quantitative analysis, may position them at same level of subsumption in the taxonomy, hence resulting in highly segregated and less organised ontology structure.

- **Example A.1.2:** The class saref:Property is defined as "anything that can be sensed, measured or controlled in households, common public buildings or offices". Here, this does not cover other aspests such as streets, open parking places, parks etc. The nature of the term "Property" can cover larger scopes. Therefore this terminology is pre-occupied with limited scope definition, which limits this ontology for future exention.

- **Example A.1.3:** The class saref:EventFunction is defined as "a function that allows to notify another device that a certain threshold value has been exceeded". In this case, this class covers very limited scope as it only includes information related to exceeding thresholds. However, there can be other events such as a user input, an exception or even a value is decreased than the defined threshold.

- **Example A.1.4:** The property saref:isUsedFor is generic term used to define highly specific relation i.e, "saref:Device saref:isUsedFor saref:Commodity". In many considered cased this terminology is suitable for wide variety of domains and ranges, such as, "command isUsedFor task", "measurement isUsedFor event", "unitOfMeasure isUsedFor Measurement" etc.

- **Example A.1.5:** The class saref:Profile has limited scope definition as it covers only device profile. There can be different considered environments, users, organizations, etc. that may also require a profile for their related data representation.

- **Example A.1.6:** The class saref:LevelControlFunction has limited scope definition as it is only focused on actuator. In contrast, many other entities may be required to utilize this class concept such as user or system.

- **Example A.1.7:** The property saref:hasValue has range xsd:float. However, there can be many possible types for measurement values.

- **Example A.1.8:** The class saref:Service is defined as "a representation of a function to a network that makes the function discoverable, registerable, remotely controllable by other devices in the network". Here, the term used is more generic than the concept it represents. There can be other services such as; the ones which are provided as a commodity. Also, in computing concept, there can be other services, which may not consider device aspects, for example, data management and analysis, data security, task monitoring and management etc.

- **Example A.1.9:** The relationship saref:actsUpon is defined as "a relationship between a command and a state". However, the term "acts upon" has potential to cover broader scope and may be used to defined other relationships as well. This may cause inconsistency in future ontology extensions or reuse.

- **Example A.1.10:** The property saref:hasCommand and saref:isCommandOf are inverse of each other. However, the definition of saref:hasCommand is stated as "A relationship between an entity (such as a function) and a command" and of saref:isCommandOf is stated as "A relationship between a command and a function.". In the formar definition, the term "entity" makes the usage of saref:hasCommand uncertain, as the user may define any other entity that may not be subsumed under saref:Function and in such case the property saref:isCommandOf can not be utilized. Another example is the definition of property saref:hasFunction, which is stated as, "A relationship identifying the type of function of a device". However the rdfs:range of this property is saref:Function which defines the complete function, rather than the type of the function.

A.2 Examples of highlighted gaps in SEAS ontologies

- **Example A.2.1:** many classes have been defined to represent different types of roads and paths, however other concepts such as zone based divisions (example: Idustrial area, public area, neighbourhood, etc.) are not available. Since the taxonomy is providing a hierarchy of deductive assertions by subsumption (i.e. generic concept classes to specific ones), the concepts regarding the zone based divisions should reside at the level in between seas:Zone and existing sub classes in seas:CityOntology.

- **Example A.2.2:** seas:BuildingSpatialStructure is defined as: "A man made structure with spatial properties". Contraty to this definition, there are many other man made structures having spatial properties, such as statues, bridges, containers, etc. Hence this definition indicates more generalized concept than that of seas:BuildingSpaceStructure. Although, it can be identified through its parent class (seas:BuildingSpace) that this "man made structure" will be specific to building, yet both definitions of seas:BuildingSpace and seas:BuildingSpaceStructure define spatial properties to be their essential feature. Therefore, it becomes complex to identify the exact usage of each of these classes based on their taxonomy and scope definitions.

- **Example A.2.3:** there can be a device which performs both sensing and actuating functionalities as well as performs some processing, such as statistical analysis, prediction etc. In such case, the properties, such as seas:actsOn, seas:actsOnProperty, seas:observes, seas:observesProperty can not represent the respective relationship as they do not have seas:Device included as rdfs:domain

- **Example A.2.4:** Consider figure A.2-1, where the orange oval describes the class defined in SEAS ontology and the others as their respective instances. The blue arrow connecting the classes are the object properties defined in SEAS ontology and the dotted arrows are their respective assertions connecting the instances. Based on the usage, it may not be possible to define "WaterLevel" as an instance of seas:PercentageProperty, as it may involve multiple data aspects including percentage. In that case, it becomes necessary that separate instance should be defined to represent the percentage property of "WaterLevel". In addition, there will be many cases, when instance of seas:Evaluation will be needed to further represent information related to seas:PercentProperty. Based on the existing structure and definitions, one proposed solution is to define seas:PercentageEvalutation instead of seas:PercentageProperty, subsumed under seas:Evaluation in Evaluation Ontology module. In that case, the instance "WaterLevel" will have the direct relationship seas:hasEvaluation with the instance "PercentLevelEvaluation". Which will reduce instance definitions and hence will reduce the complexity.

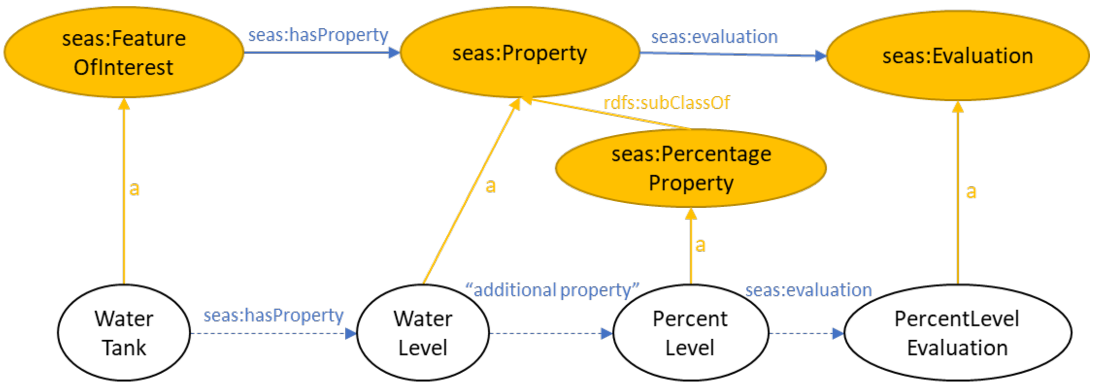
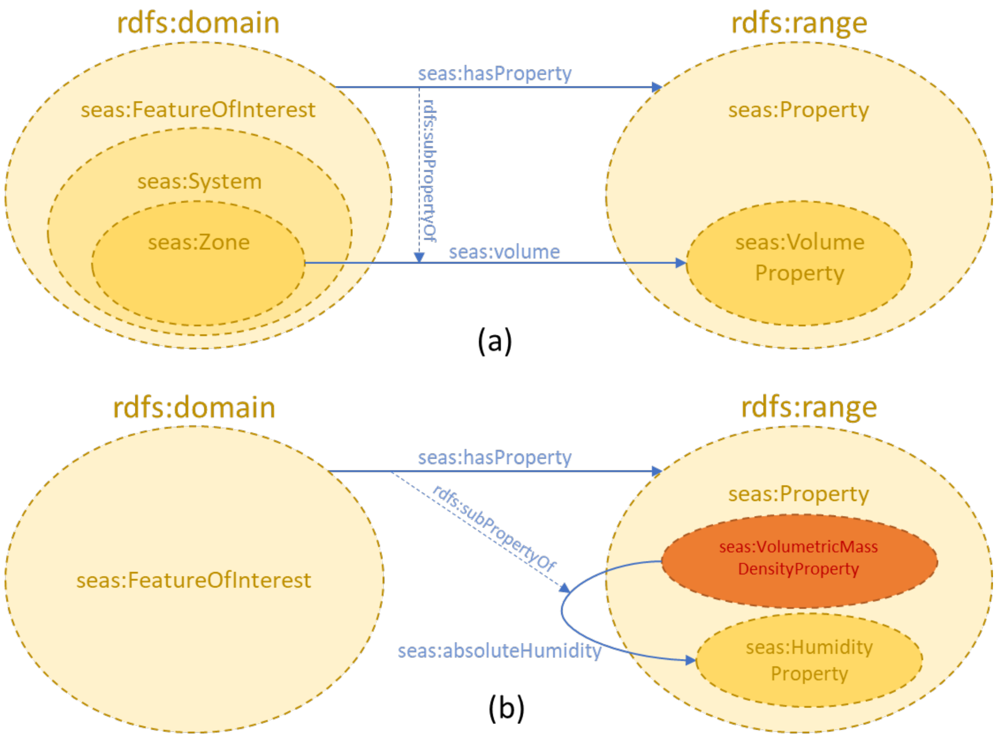


Figure A.2-1: Assertion example



•

Figure A.2-2: rdfs:domain and rdfs:range assertions of object properties defined in SEAS ontologies

- **Example A.2.5:** Consider figure A.2-2, where two object properties assertions are compared, which are seas:volume and seas:absoluteHumidity. Both have relation rdfs:subPropertyOf with seas:hasProperty. In figure A.2-2(a), it can be considered as a valid assertion for seas:volume, as both of it's rdfs:domain and rdfs:range are subsumed under the rdfs:domain and rdfs:range of seas:hasProperty respectively. Whereas for seas:absoluteHumidity in figure A.2-2(b), the assertions are uncertain and complex to realize by the system, because according to description in seas:FeatureOfInterestOntology, seas:Property can also be considered and utilized as seas:FeatureOfInterest, yet any assertion supporting this description is not available

- **Example A.2.6:** pep:hasInput is described as, "Links a Procedure to the (unique) description of its input". Whereas, it's domain and range are not specified.

- **Example A.2.7:** pep:implementsProcedure can have more clear indication of it's usage than pep:implements, and this also avoids preoccupation of term implements.

- **Example A.2.8:** The object property seas:seeks in seas:OfferingOntology is defined as "Links an agent to a procedure it seeks", while having seas:Player defined as it's rdfs:domain. The importance of this realization becomes critical in the seas:ComfortOntology module, where it has used the term "agent" referring to foaf:agent. This external class has been imported in this module from Friend Of A Friend (FOAF) ontology, which is defined as “An agent (eg. person, group, software or physical artifact)……. The Agent class is the class of agents; things that do stuff”. The definition of seas:Player does not specify the difference, similarity or any relationship between foaf:Agent and seas:Player.

- **Example A.2.9:** The class seas:Laundry is defined as, “A room or zone, as in a home or apartment building, reserved for doing the family wash”. Here the term “family wash” highlights ambiguity, as there was no reference available for it’s description. In general laundry is also used in industrial zones where they may process different cloth. Hence based on the terminology, the definition requires more clarity.

- **Example A.2.10:** The class seas:LowEnergyHouse is defined as, “A house typically consuming half the energy than a norm house”. However, in this ontology the term “building” is refered and defined instead of “house”, as this class is defined as subclass of seas:Building. Same is the case with class seas:PassiveHouse. Generally, house can be considered as more specific concept of building in the taxonomy. Nonetheless, unlike seas:Building, the concept definition of “house” is not available. Although the ontology has defined class seas:SmallHouse, it is not referred in case of seas:LowEnergyHouse.

Annex B: Ontology Concept Definitions

This Annex provides the definitions of the concepts proposed in Clause 7.

B.1 Definitions of Classes in Common Ontology

Following table provides the concept definitions of Classes defined in Common Ontology.

Table B.1-3: Class Definitions defined in Common Ontology

Class	Definition
Actuation	Actuation is a Procedure Execution of some Procedure, by an Actuator, and has an influence on the environment.
Actuator	An Actuator is a Device that implements one or many Actuation, based on some Procedure.
AdministrativeArea	An Administrative Area is a Zone, covering a region of land for the purposes of administration.
City	A City is a permanent and densely populated Administrative Area focused on urbanization. Unlike rural areas, cities generally have extensive and well organized systems for housing, transportation, sanitation, utilities, land use, production of goods and communication.
Commodity	Commodity is a Feature of Interest, which represents a marketable good or resource, having substantial fungibility.
Connection	A Connection is a Feature of Interest, which represents the connectivity between two or more instances of a System.
ConnectionPoint	A Connection Point is a Feature of Interest, which represents a common point of connectivity among Connections and Systems. It may or may not also be controlled by a System through which the System allows the other Systems to be connected through one or more Connections.
ContactPoint	Contact Point is a Property, which represents the information required to communicate with certain types of Feature of Interests, such as Player and Facility.
Country	A country (sometimes also called a state or a nation) is an Administrative Area which has its own government and acts as a political entity in the world.
Device	Device is a physical or tangible System which is built to execute one or more procedures to perform certain tasks. The execution of this device may or may not impact the phyiscal world.
DimensionProperty	The Property to define measurable extent of some kind, such as length, breadth, depth, or height.
District	A District is an Administrative Area in a Country, which has some distinguishing feature from surrounding areas or is used for some official purpose, managed by local governing authorities.
Estimating	Estimation is a Procedure, which includes the steps to calculate the average or rough evaluation of some Property such as: Width Property, Length Property, Location Property, Price Property, etc. It can be executed by Estimation
Estimation	Estimation is a Procedure Execution of an average or rough evaluation of some Property like Width Property, Length Property, Location Property, Price Property, etc. It's outcome can be come Evaluation.
Evaluation	An Evaluation contains the assessment or judgement about a Property.
Facility	Facility is a Zone, which represents a A place, amenity, or piece of equipment provided for a particular purpose.
FeatureOfInterest	A Feature of Interest is an abstraction for real world phenomena (thing, person, event, etc.), who's definition remains same irrespective of time. Consider an example, where "Square" is defined ais one of the subclasses of Feature of Interest. If through time it changes its shape to rectangle, then it cannot be considered as Feature of Interest. Rather class Quadrilateral will be better definition to cover this phenomenon.
Forecast	Forecast is a ProcedureExecution of Forecasting Procedure, which involves prediction or estimation of future events or trends. It can be executed by a Forcaster.

Forecaster	A Forecaster is a Procedure Executor, which implements some Forecasting procedure, and may generate Forecasts.
Forecasting	Forecasting is a Procedure, which includes the steps to predict or estimate some information in a future event or trend.
Function	The functionality necessary to accomplish the task for which a Procedure Executor is designed.
KPI	A Key Performance Indicator (KPI) is a type of Property which used to evaluate success of a Feature of Interest or of a particular activity in which it engages.
KPIAssessment	Represents the assessment of a KPI calculated by a given agent in a given time.
LengthProperty	Length Property is a Dimension Property, which specifies the length or the extent of some Feature Of Interest from one end to another.
LocationProperty	The Property defining a particular place or position.
Neighbourhood	Neighbourhood is a Zone representing a geographically localised community, with considerable face-to-face social interactions among the inhabitants. A Neighbourhood usually is situated in an Administrative Area or some other localization such as village.
Observation	Observation is a ProcedureExecution of monitoring, inspection, examination or recording of some information in a Phenomenon or event.
Observing	Observing is a Procedure, which includes the steps to monitor, inspect, examin or record some information in a Phenomenon or event.
Phenomenon	Phenomenon is a fact or a situation that is observed to exist or happen, especially the one whose cause or explanation is in question. Phenomenon can be natural or artificial.
Player	Player is a System, which represents a company, organization or an individual that has influence within an activity, industry or type of work.
PocedureExecutor	A System, involved in the execution or implementation of a Procedure.
PriceEvaluation	The Evaluation, which describes the assessment or judgement about Price Property.
PriceProperty	PriceProperty is a Property, which describes the cost or value of certain types of Feature of Interests like Commodity, that can be measured certain Currency.
PriceRate	Price Rate is a Price Evaluation, which specifies the per unit cost of specific Commodity. Here the unit can be specified in terms of a Currency or some other means of payment.
Procedure	A reusable series of steps or actions that can be carried out to achieve results.
ProcedureExecution	The act of carrying out a Procedure.
Profile	Profile is a Property which describes the important or relevant facts to characterize a particular System. NOTE: has broader scope than saref:Profile
ProfilePicture	Profile Picture is a Property, which represents an image for the characterization of a Feature of Interest.
Property	An observable or operable quality or characteristics of Feature of Interest or a Phenomenon, which can be observable, measurable or operable by a Feature of Interest.
PublicService	Public Service is a Property, which represents acts or performences by a Service Provider, to provide value to the customer, and which has a transaction cost.
Sensor	Sensor is a Device which, based on some procedure, performs one or many Observation of a physical Phenomenon from the environment.
Service	From the perspective of software or system architecture, service is a single or a set of characteristics of a System, with a purpose of enabling clients to perfrom a single or a set of tasks. Here a client can be but not limited to a Procedure, Player or foaf:Person.
State	A particular condition that someone or something is in at a specific time.
StateEvaluation	State Evaluation is an Evaluation, which describes the assessment or judgement about a State.
System	System is a group of interacting or interrelated elements (sub systems) that act according to the set of rules to form a unified Feature of Interest. A system in an environment, is described by its boundaries, structure and purpose and is influenced by the environment.
Task	A task represents the steps carried out towards the specific goal for which a Procedure Executor is designed (from a user perspective)
WidthProperty	Width Property is a Dimension Property, which specifies the width or the horizontal extent taken at right angles to the length of some Feature Of Interest.
Zone	An area or stretch of land or space having a particular characteristic, purpose, or use, or subject to particular restrictions

1058

B.2 Definitions of Properties in Common Ontology

Following table provides the concept definitions of Object Properties defined in Common Ontology.

Table B.2-1: Class Definitions defined in Common Ontology

Object Property	Definition
acceptsCurrency	This Object Property links an individual of type common:PriceRate to an individual of type saref:Currency, indicating currency accepted for business. Domain: common:PriceProperty Range: common:PriceRate
connectedThrough	This Object Property links an individual of type common:System to an individual of type common:Connection, describing it’s connectivity using a connection, to other systems of the environment. A single connection can be used to connect more than two systems to represent common or sharing connectivity among systems. Domain: common:System Range: common:Connection
connectedTo	Represents a relationship of connectivity between the individuals of type common:System. This property can be used to only state the connectivity between two systems, but can not represent the features such as mode or end of connectivity. This property is Symmetric. If systems are using common mode of connection, then this property can be used to specify connected pairs of systems through that connection. Domain: common:System Range: common:System
connectionPointOf	This Object Property links an individual of type common:ConnectionPoint to an individual of type common:System. This property indicates a point of connectivity for a particular system. However, multiple connection point can belong to a single System. Domain: ConnectionPoint Range: System
connectsAt	This Object Property links an individual of type common:System to an individual of type common:ConnectionPoint. It does not directly complements common:connectionPointOf relationship, as it indicates connectivity when the linked connection point is also linked with some connection. If not, then it is recommended to used common:connectionPointOf instead, to show the relationship between a system and a connection point. Domain: common:System Range: common:ConnectionPoint
connectsSystem	This Object Property links an individual of type common:Connection to an individual of type common:System. This relationship complements common:connectedThrough relationship between same individuals, linked using common:connectedThroughthis property. Domain: common:Connection Range: common:System
connectsSystemAt	This Object Property links an individual of type common:Connection to an individual of type common:ConnectionPoint. This relationship complements common:connectsSystemThrough relationship between same individuals, linked using common:connectsSystemThroughthis property. Domain: common:Connection Range: common:ConnectionPoint
connectsSystemThrough	This Object Property links an individual of type common:ConnectionPoint to an individual of type common:Connection. Through this property it can be described that which systems are connected together through the specific connections and connection points. Domain: common:ConnectionPoint Range: common:Connection
consumesProcedureExecutor	This Object Property links an individual of type common:Service to an individual of type common:ProcedureExecutor, representing a service utilizing the procedure executor, who will perform certain procedure executions. Domain: common:Service Range: common:ProcedureExecutor
consumesService	This Object Property link an individual of type common:System to an individual of type

	common:PublicService. This defined the consumer relationship for a particular service. Domain: common:System Range: common:PublicService.
definesContactpoint	Any individual can be linked to define its relationship with an individual of type common:ContactPoint, to associate contact information. Range: common:ContactPoint
definesService	This Object Property links an individual of type common:Profile to an individual of type common:PublicService or common:Service. This represents a defining the service description which may include information such as nature, scope, policies, etc. Domain: common:Profile Range: common:Service OR common:PublicService
derivesFrom Connection	A Connection is a common:FeatureOfInterest, which represents the connectivity between two or more instances of common:SystemRepresents a uni-directional relationship between individuals of type common:Property, where one individual is derived from the other. This relationship can be transitive between 3 or more individuals of type Property, having the direction towards the property specificed as a range. Domain: common:Property Range: common:Property
estimtatedOn	This Object Property links an individual of type common:Estimation to an individual of type time:Instant, representing time of estimation. Domain: common:Estimation Range: time:Instant
evaluatedForDuration	This Object Property links an individual of type common:Evaluation to an individual of type time:TemporalDuration, indicating the duration of evaluation. Domain: common:Evaluation Range: time:TemporalDuration
evaluatesProperty	This Object Property links an individual of type common:System to an individual of type common:Property. Domain: common:System Range: common:Property
evaluationOf	This Object Property links an individual of type common:Evaluaition to an individual of type common:Property. This relationship complements common:hasEvaluation relationship between same individuals. Domain: common:Evaluation Range: common:Property.
executionDuration	This Object Property links an individual of type common:ProcedureExecution to an individual of type time:TemporalDuration, indicating duration of execution. Domain: common:ProcedureExecution Range: time:TemporalDuration
executionTime	This Object Property links an individual of type common:ProcedureExecution to an individual of type time:Instant, indicating the time of execution. Domain: common:ProcedureExecution Range: time:Instant
hasAddress	This Object Property links an individual of type common:Zone to an individual of type common:ZoneBasedLocation, representing the location information in terms of Zones, such as Neighborbood, City, Country etc. Domain: common:Zone Range: common:ZoneBasedLocation
hasCommand	This Object Property links an individual of type common:Procedure to an individual of type saref:Command, representing the commands to be executed, specified inside the procedure. Domain: common:Procedure Range: saref:Command
hasDateCreated	Any individual can be linked to define its relationship with an individual of type time:Instant, representing date of creation. Range: time:Instant
hasDateModified	Any individual can be linked to define its relationship with an individual of type time:Instant, representing the date of modification. Range: time:Instant
hasEvaluation	Represents a uni-directional relationship between an individual of type common:Property to anan individual of type common:Evaluation, which evaluates that

	particular property. There can be multiple evaluations of a property, which can be represented using this relationship. Domain: common:Property Range: common:Range
hasPriceProperty	Any individual can be linked to define its relationship with an individual of type common:PriceProperty, to represent its cost. Range: common:PriceProperty
hasPriceRate	This Object Property links an individual of type common:PriceProperty to an individual of type common:PriceRate, indicating price rate. This property extends common:hasEvaluation. Domain: common:PriceProperty Range: common:PriceRate
hasProperty	This Object Property links an individual of type common:FeatureOfInterest to a single unique individual of type common:Property. This represents the relationship with the Properties that characterized some Feature of Interests. This property is Functional as well as Inverse Fuctional. Domain: common:FeatureOfInterest Range: common:Property
hasState	This Object Property links an individual of type common:FeatureOfInterest to an individual of type common:State Domain: common:FeatureOfInterest Range: common:State
hasStateEvaluation	This Object Property links an individual of type common:State to an individual of type common:hasStateEvaluation. Domain: common:State Range: common:StateEvaluation
hasSubSystem	Represents a uni-directional relationship between individuals of type common:System, where the individual in the domain is composed of the one specified in range (the sub-system). However, this does not restrict that the sub-system is essential for the existence of super-system. This relationship can be transitive between 3 or more individuals of type common:System, having direction towards the sub-systems. Domain: common:System Range: common:System
implements	This Object Property links an individual of type common:ProcedureExecutor, which implement a procedure, to an individual of type common:Procedure. A An individual of type common:pProcedure eExecutor can implement have 0 to multipleany p common:implements relationships with different individuals of type common:Procedures, which can be representated using this relationship. Similarly, a single procedure can involve multiple procedure executors. Domain: common:ProcedureExecutor Range: common:Procedure
includesProfilePicture	This Object Property links an individual of type common:Profile to an individual of type common:ProfilePicture. Domain: common:Profile Range: common:ProfilePicture
isAboutPhenomenon	Any individual can be linked to define its relationship with an individual of type common:Phenomenon. This relationship can represent a natural, artificial or conceptual involvement of an entity in a phenomenon. Range: common:Phenomenon
isExecutedBy	This Object Property links an individual of type common:ProcedureExecution to an individual of type common:ProcedureExecutor, specifying and authorizing the procedure executor to perform the execution. Domain: common:ProcedureExecution Range: common:ProcedureExecutor
isLocatedAt	This Object Property links an individual of type common:FeatureOfInterest to an individual of type common:LocationProperty, representing the location information of a feature of interest. Domain: common:FeatureOfInterest Range: common:LocationProperty
isOfferedAtLocation	This Object Property links an individual of type common:PublicService to an individual of type common:LocationProperty. This indicates the location/locations where a public

	service can be offered or is allowed to offer. Domain: common:PublicService Range: common:LocationProperty
isOfferedAtZone	This Object Property links an individual of type common:PublicService to an individual of type common:Zone, represting the availability of a public service in the specified zones. Domain: common:PublicService Range: common:Zone
isOfferedBy	This Object Property links an individual of type common:PublicService to an individual of type common:ServiceProvider. This relationship complements common:offersService relationship between same individuals, linked using this property. Domain: common:PublicService Range: common:ServiceProvider.
isOfferedForDuration	This Object Property links an individual of type common:PublicService to an individual of type time:TemporalDuration. This can be used to defined the duration of the availability of the public service to be offered. Domain: commonPublicService Range: time:TemporalDuration
isOfferedOnTime	This Object Property links an individual of type common:PublicService to an individual of type time:Instant. This can be used to defined multiple time instants of the availability of the public service or the log, when the service was offered. Domain: common:PublicService Range: time:Instant
isPropertyOf	This Object Property links an individual of type common:Property to single uniquean individual of type common:FeatureOfInterest. This relationship complements common:hasProperty relationship between same individuals, linked using this pcommon:hasProperty. This property is Functional as well as Inverse Functional. . Domain: common:Property Range: common:FeatureOfInterest
observedOn	This Object Property links an individual of type common:Observation to an individual of type time:Instant, representing the time of observation. Domain: commonObservation Range: time:Instant
offersService	This Object Property links an individual of type common:ServiceProvider to an individual of type common:Service. Through this property, a service provider can offer multiple services. However, a single service can be offered by only one service provider at a time. Though, here can be multiple entities representing a single service provider (hence can be represented by common:hasSubSystem relationship). Domain: common:ServiceProvider Range: common:PublicService
operatingAtLocation	This Object Property links an individual of type common:ProcedureExecutor to an individual of type common:LocationProperty, indicating the location, where the procedure executor is operating. Domain: common:ProcedureExecutor Range: common:LocationProperty
performsExecution	This Object Property links an individual of type common:ProcedureExecutor, which executes a pProcedure, to an individual of type common:ProcedureExecution. Multiple procedure executors can perform a single procedure execution (representing the concepts parallel processing). individuals of type common:ProcedureExecutor can have this relationship with a specific individual of type common:ProcedureExecution. Also, multiple individuals of type common:ProcedureExecution can be linked with a specific individual of type common:ProcedureExecutor. Domain: common:ProcedureExecutor Range: common:ProcedureExecution
providesService	This Object Property links an individual of type common:System to an individual of type common:Service or common:PublicService, indicating the type of service which a system can provide. Domain: common:System Range: common:Service OR common:PublicService
subSystemOf	Represents a uni-directional relationship between individuals of type common:System, where one individual (the sub-system) takes part in the composition of the other. This

	relationship can be transitive between 3 or more individuals of type common:System, having direction towards the systems specified in range. This relationship complements common:hasSubSystem relationship between same individuals, linked using common:hasSubSystemthis property. Domain: common:System Range: common:System
supportsProcedureExecution	This Object Property links an individual of type common:Service to an individual of type common:ProcedureExecution, indicating which procedures can be executed by this service. Domain: common:Service Range: coomon:ProcedureExecution
usedProcedure	This Object Property links an individual of type common:ProcedureExecution, which involves a Procedure, to an individual of type common:Procedure. A procedure execution can utilize multiple procedures and similarly a procedure can be a part of multiple procedure executions. Domain: common:ProcedureExecutionor Range: common:Procedure

History

This clause shall be the last one in the document and list the main phases (all additional information will be removed at the publication stage).

Publication history		
V1.1.1	<yyyy-mm-dd>	<Milestone>

Draft history (to be removed on publication)		
V1.1.1	<yyyy-mm-dd>	<CR ID> applied – <Summary of changes>
V0.0.1	2019-02-22	Skeleton
V0.1.0	2019-05-24	Incorporated agreed documents from RDM#40 RDM-2019-0049R02-TR-0061_Study_on_SAREF4City RDM-2019-0053R01-TR-0061_Study_on_SEAS
V0.2.0	2020-01-21	Incorporated agreed documents from RDM#43 RDM-2019-0128R03-TR-0061_Smart_Ontology_Gap_Analysis
V0.3.0	2021-01-08	Incorporated agreed documents from RDM#47.1 RDM-2020-0086R02-Smart_City_Ontologies_for_oneM2M RDM-2020-0087R02-Examples_for_Gap_Analysis
V0.4.0	2022-11-30	Incorporated agreed documents from RDM#54 RDM-2021-0050R02-TR-0061-Addition_of_Common_Ontology_concept_definitions_and_descriptio RDM-2022-0030R01-introduction_to_existing_smart_city_ontologies
V0.5.0	2022-12-01	Incorporated agreed documents from RDM#57 RDM-2022-0095R02-TR-0061_conclusion_and_clean-up