



Project Acronym: **VICINITY**

Project Full Title: **Open virtual neighbourhood network to connect intelligent buildings and smart objects**

Grant Agreement: **688467**

Project Duration: **48 months (01/01/2016 - 31/12/2019)**

Deliverable D2.3

Evaluation of the semantic model in real world scenarios

Work Package: **WP2– Standardization Analysis and VICINITY platform conformity**

Task(s): **T2.3 – VICINITY platform standardization and conformity assessment**

Lead Beneficiary: **UPM**

Due Date: **30 June 2019 (M42)**

Submission Date: **16 July 2019 (M43)**

Deliverable Status: **final**

Deliverable Type: **R**

Dissemination Level: **PU**

File Name: **VICINITY_D2.3_SemanticModelEval_v1.0.docx**



*This project has received funding from the European Union's Horizon 2020
Research and innovation programme under Grant Agreement n°688467*

VICINITY Consortium

No	Beneficiary		Country
1.	TU Kaiserslautern (Coordinator)	UNIKL	Germany
2.	ATOS SPAIN SA	ATOS	Spain
3.	Centre for Research and Technology Hellas	CERTH	Greece
4.	Aalborg University	AAU	Denmark
5.	GORENJE GOSPODINJSKI APARATI D.D.	GRN	Slovenia
6.	Hellenic Telecommunications Organization S.A.	OTE	Greece
7.	bAvenir s.r.o.	BVR	Slovakia
8.	Climate Associates Ltd	CAL	United Kingdom
9.	InterSoft A.S.	IS	Slovakia
10.	Universidad Politécnica de Madrid	UPM	Spain
11.	Gnomon Informatics S.A.	GNOMON	Greece
12.	Tiny Mesh AS	TINYM	Norway
13.	HAFENSTROM AS	ITS	Norway
14.	Enercutim – Associação Empresarial de Energia Solar de Alcoutim	ENERC	Portugal
15.	Municipality of Pylaia-Hortiatis	MPH	Greece

Disclaimer

This document reflects only the author's views and the European Union is not liable for any use that may be made of the information contained therein.

¹ **Deliverable Type:**

R: Document, report (excluding the periodic and final reports)
 DEM: Demonstrator, pilot, prototype, plan designs
 DEC: Websites, patents filing, press & media actions, videos, etc.
 OTHER: Software, technical diagram, etc.

² **Dissemination level:**

PU: Public, fully open, e.g. web
 CO: Confidential, restricted under conditions set out in Model Grant Agreement
 CI: Classified, information as referred to in Commission Decision 2001/844/EC.

Authors List

Leading Author (Editor)				
Surname	First Name	Beneficiary	Contact email	
Fernández Izquierdo	Alba	UPM	albafernandez@fi.upm.es	
Co-authors (in alphabetic order)				
No	Surname	First Name	Beneficiary	Contact email
1.	Cimmino	Andrea	UPM	cimmino@fi.upm.es
2.	García Castro	Raúl	UPM	rgarcia@fi.upm.es
3.	Poveda Villalón	María	UPM	mpoveda@fi.upm.es

Reviewers List

List of Reviewers (in alphabetic order)				
No	Surname	First Name	Beneficiary	Contact email
1.	Oravec	Viktor	BVR	viktor.oravec@bavenir.eu
2.	Mach	Marian	IS	Marian.Mach@tuke.sk
3.	Kölsch	Johannes	UNIKL	koelsch@cs.uni-kl.de

Revision Control

Version	Date	Status	Modifications made by
0.1	08.05.2019 (M41)	Initial Draft	Alba Fernández Izquierdo (UPM)
0.2	09.06.2019 (M42)	First Draft with contributions	Alba Fernández Izquierdo (UPM), Andrea Cimmino (UPM), Raúl García Castro (UPM), María Poveda Villalón (UPM)
0.5	09.06.2019 (M42)	<i>Deliverable version uploaded for Quality Check</i>	Alba Fernández Izquierdo (UPM), Andrea Cimmino (UPM), Raúl García Castro (UPM), María Poveda Villalón (UPM)
0.6	02.07.2019(M43)	General update according to QAR	Alba Fernández Izquierdo (UPM), Andrea Cimmino (UPM),
0.7	15.07.2019 (M43)	<i>Final revision</i>	Izquierdo (UPM)
1.0	16.07.2019 (M43)	Submission to the EC	Zivkovic (UNIKL)

Executive Summary

The present document is the deliverable “D2.3 - Evaluation of the semantic model in real world scenarios” of the VICINITY project, funded by the European Commission’s Directorate-General for Research and Innovation (DG RTD), under its Horizon 2020 Research and Innovation Programme (H2020). The VICINITY ontology network developed in the context of this project consists so far of five ontology modules, i.e., the VICINITY Core (Core), the Web of Things (WoT), the WoT mappings (Mappings), the VICINITY Adapters (Adapters), and the Datatypes (Datatypes) ontologies, belong to the VICINITY ontology network and aim to provide interoperability in the IoT domain. The Core ontology represents the information needed to exchange IoT descriptor data between peers through the VICINITY platform; this ontology is being created by following a cross-domain approach and implements requirements from different domain experts. The WoT ontology aims to model the Web of Things domain according to the W3C WoT Interest Group¹⁹ descriptions. The Mappings ontology represents the mechanism for accessing the values provided by web things in the VICINITY platform. The Adapters ontology aims to model all the different types of devices and properties that can be defined in the VICINITY platform. Finally, the Datatypes ontology aims to model the required and provided datatypes that are used in the interaction patterns of the platform. The methodology for developing this ontology network is iterative and based on the NeOn methodology [1].

This document covers the following main topics:

- Validation regarding the model, to assure that there are no inconsistencies in the ontologies (by using semantic reasoners) and that there are no modelling errors (by using the tool OOPS!).
- Verification regarding their ontological requirements, to guarantee that all the requirements asked by the domain experts are satisfied by the ontology.
- Validation regarding pilot data, to analyse how the data is used in VICINITY relying on the VICINITY ontology.
- Verification regarding IoT standards, which analyses the coverage of the VICINITY ontology regarding the ontological commitments of a set of well-known IoT standards.

From the evaluation presented in this document it could be concluded that the VICINITY network does not have inconsistencies or modelling errors, and that it covers all the requirements given by the partners. Additionally, it was also concluded that, even though the VICINITY ontology network does not cover all the requirements in those standards, there are no inconsistencies between the standards and the VICINITY ontologies and that VICINITY ontology network has partial conformance with the IoT standards. However, this coverage analysis also shows that the VICINITY ontology network is out of scope of the analysed IoT standards, which is expected because the IoT standards are more generic than the VICINITY ontology network.

We have analysed the Things and Thing Descriptions registered by the pilots to verify which parts of the ontology are used the most. It is well known that VICINITY relies on several core components that work with this data, and therefore, the correct specification of it is paramount. The results of our analysis advocates that pilots are correctly using the ontology, although some parts should be used more; like the parts that refer to the contextual data, or the mappings.

List of Definitions and Abbreviations

Abbreviation	Definition
EC	European Commission
IoT	Internet of Things
WoT	Web of Things
RDF	Resource Description Framework
LSP	Lexico-Syntactic Pattern
ETSI	European Telecommunications Standards Institute
OCF	Open Connectivity Foundation
SSN	Semantic Sensor Network
IEC	International Electrotechnical Commission
ISO	International Organization for Standardization
W3C	World Wide Web Consortium

Content

VICINITY Consortium	2
Authors List	3
Reviewers List	3
Revision Control.....	4
Executive Summary	5
List of Definitions and Abbreviations	6
List of Tables	8
List of Figures.....	8
1. Introduction.....	9
1.1. Context within VICINITY	11
1.2. Objectives in Work Package 2 and Task 2.3.....	12
2. Ontology evaluation of technical quality	13
2.1. Semantic reasoners	13
2.2. Ontology Pitfall Scanner! (OOPS!)	13
3. Validation with regards to ontological requirements	16
3.1. Testing method	16
3.1.1. Test design.....	17
3.1.2. Test implementation	18
3.1.3. Test execution	18
3.2. Testing infrastructure	20
3.3. Testing results	23
4. Validation with regards to pilots' data	25
4.1. Methodology for validating VICINITY	27
4.2. Things Monitor implementation	28
4.3. Results of pilots' data validation.....	28
5. Validation with regards to standards	32
5.1. Coverage analysis method	32
5.1.1. Test results analysis.....	33
5.2. Coverage analysis infrastructure.....	33
5.1. Testing results	35
6. Conclusions	58
7. References	60
Keywords	62

List of Tables

Table 1: OOPS! Results for the VICINITY ontology network.....	15
Table 2: Supported test expressions.....	17
Table 3: Summary of requirements categorization for the VICINITY ontology network.....	23
Table 4 Tests executed for each VICINITY ontology.....	24
Table 5: Summary of requirements information for the IoT standards.....	36
Table 6: Summary of testing results for the IoT standards.....	37
Table 7: Testing results for SAREF ontology regarding to VICINITY requirements.....	38
Table 8: Testing results for SAREF and VICINITY ontologies.....	39
Table 9: Overlap between SAREF and VICINITY ontologies.....	39
Table 10: Testing results for W3C SSN ontology regarding VICINITY requirements.....	46
Table 11: Testing results for SSN and VICINITY ontologies.....	47
Table 12: Overlap between the SSN and VICINITY ontologies.....	48
Table 13: Testing results for the oneM2M ontology regarding to VICINITY requirements.....	52
Table 14: Testing results for oneM2M and VICINITY ontologies.....	52
Table 15 Overlap between the oneM2M and VICINITY ontologies.....	54
Table 16: Overview of the number of VICINITY requirements satisfied by each IoT standard ...	57
Table 17: Overview of the number of the IoT standards' requirements satisfied by the VICINITY ontology network.....	57

List of Figures

Figure 1 VICINITY ontology network overview.....	10
Figure 2 Ontology development methodology.....	11
Figure 3: OOPS! interface.....	14
Figure 4: Testing method with inputs and outputs.....	16
Figure 5: Test execution steps.....	20
Figure 6: Themis and its relationship with the testing method.....	21
Figure 7: Themis interface.....	22
Figure 8: Test case in RDF exported by Themis.....	22
Figure 9: Type of requirements considered by our system.....	25
Figure 10: Architecture of our validation approach.....	27
Figure 11: Things from pilots with detailed descriptions and interoperable.....	29
Figure 12: Interaction patterns.....	29
Figure 13: Accessibility of interaction patterns.....	30
Figure 14: Types from Adapters used by Pilots.....	30
Figure 15: Use of SAREF by the Pilots.....	31
Figure 16: Coverage analysis with inputs and output.....	32
Figure 17: Themis interfaces for loading test suites from files.....	34
Figure 18: Example of coverage report.....	35
Figure 20: VICINITY Requirements conceptualization of Device and Thing.....	46
Figure 19: SAREF Requirements conceptualization of Device and Thing.....	46
Figure 21: SSN Requirements conceptualization of Sensor and Thing.....	51
Figure 22: VICINITY Requirements conceptualization of Sensor and Thing.....	51
Figure 23: oneM2M Requirements conceptualization of Device and Thing.....	56
Figure 24: VICINITY Requirements conceptualization of Device and Thing.....	56

1. Introduction

The VICINITY interoperability approach relies on ontologies (i.e., semantic data models) that will be exploited throughout the VICINITY infrastructure. Such ontologies, which are developed as an ontology network, are explained in the VICINITY deliverable “D2.2 Detailed Specification of the Semantic Model” and should be evaluated regarding several criteria before they are released. The VICINITY ontology network consists of five ontologies so far, namely the VICINITY Core (Core), the Web of Things (WoT), the WoT mappings (Mappings), the VICINITY Adapters (Adapters), and the Ontology model for datatypes (Datatypes) ontologies. The Core ontology represents the information needed to exchange IoT descriptor data between peers through the VICINITY platform; this ontology is being created by following a cross-domain approach and implements requirements from different domain experts. The WoT ontology aims to model the Web of Things domain. The Mappings ontology represents the mechanism for accessing the values provided by web things in the VICINITY platform. The Adapters ontology aims to model all the different types of devices and properties that can be defined in the VICINITY platform. Finally, the Datatypes ontology aims to model the required and provided datatypes that are used in the interaction patterns of the platform. More information about these ontologies is available in the VICINITY ontology portal.¹ Figure 1 shows an overview of the VICINITY ontology network so far.

¹ <http://vicinity.iot.linkeddata.es>

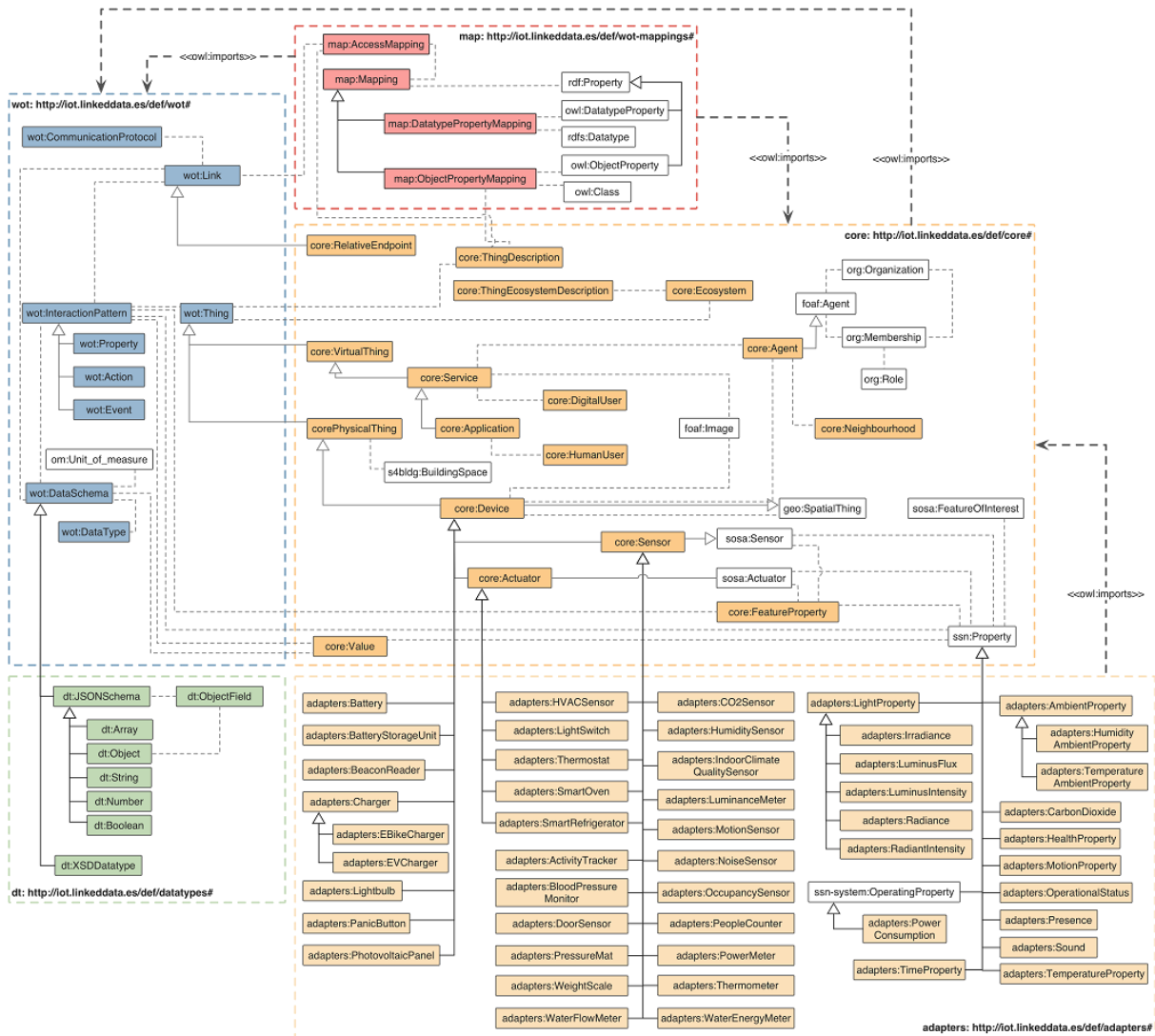


Figure 1 VICINITY ontology network overview

The development methodology process followed in this project is iterative and, therefore, several versions of the ontology with new requirements are released. Figure 2 present the development methodology steps that have to be performed and of the products resultant of them.

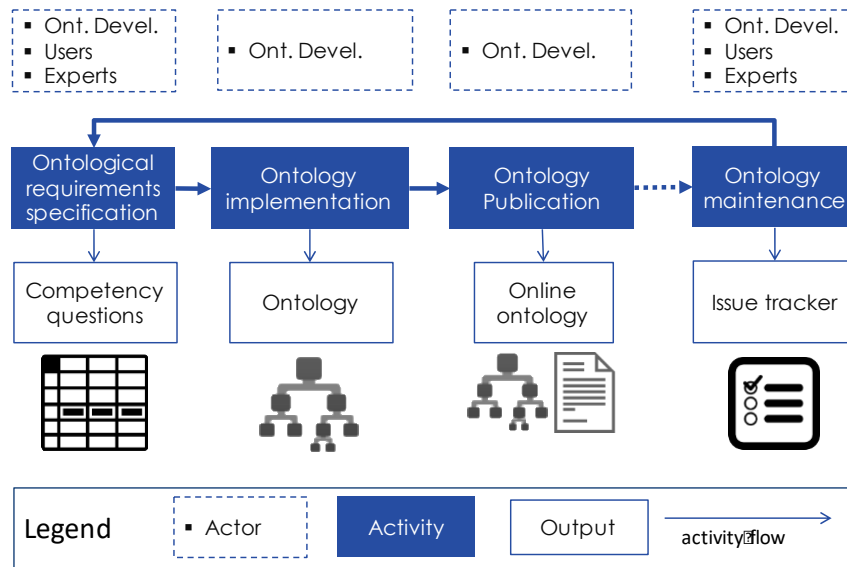


Figure 2 Ontology development methodology

Before publishing a release version of the ontology, during the ontology implementation the ontology should be evaluated. This evaluation process, as it is described in [1], refers to the activity of checking the technical quality of an ontology against a frame of reference. The goal of such activity is to guarantee the correctness and completeness of the generated ontologies. Ontology evaluation includes:

- **Ontology Validation**, which is the ontology evaluation that compares the meaning of the ontology definitions against the intended model of the world aiming to conceptualize.
- **Ontology Verification**, which is the ontology evaluation which compares the ontology against the ontology specification document (ontology requirements and competency questions), thus ensuring that the ontology is built correctly in compliance with the ontology specification.

In this document, the evaluation process of the VICINITY ontologies was carried out following the following criteria: (1) validation regarding the model, (2) verification regarding their ontological requirements, (3) validation regarding pilot data, and (4) verification regarding IoT standards.

The goal of this deliverable is to detail how the developed VICINITY ontologies are validated to assure the users that they are correctly built, complete, and ready to be used.

The rest of this deliverable is structured as follows:

- **Section 2** provides the validation of the ontologies regarding modelling issues.
- **Section 3** is devoted to validation of the ontologies regarding their ontological requirements.
- **Section 4** describes the validation regarding the pilot data.
- **Section 5** is dedicated to the description of the validation regarding several IoT standards.
- **Section 6** provides some conclusions and future lines of work.

1.1. Context within VICINITY

The D2.3 (Evaluation of the semantic model in real world scenarios) document is part of WP2 (Semantic model). It is derived from the D2.2 where the VICINITY semantic model was presented, in order to validate the generated VICINITY ontology. Additionally, D2.3 is also derived from the deliverables D3.4,

D3.5 and D3.6, as the Things Monitor described in this document will allow to check whether the infrastructures registered in the VICINITY platform satisfy the requirements established by the VICINITY semantic interoperability approach.

1.2. Objectives in Work Package 2 and Task 2.3

The objective of WP2 is to validate the ontologies and interfaces specified by the project and to contribute the results and experience to standardisation bodies. WP2 will define a semantic model for cross-domain IoT networks and demonstrate and validate it in real scenarios and will support the correlation between the proposed semantic model and existing IoT platforms and infrastructures. Finally, the evaluation of demonstrators will be used to validate the model and to produce the corresponding standardization recommendations. Within Task 2.3 the platform and the semantic model will be validated in real life situations with inputs from the project demonstration sites and that will serve as the basis for recommendations, amendments and extensions to standardization.

2. Ontology evaluation of technical quality

Along this section the evaluation of technical quality process of the VICINITY ontology network is described. This evaluation includes the detection of inconsistencies in the ontologies by using semantic reasoners and the detection of modelling errors by using the tool OOPS!.

2.1. Semantic reasoners

To ensure the quality of ontologies, there is a need for dealing with the inconsistency and uncertainty in the ontologies. Therefore, once an ontology is developed, it is needed to assure that there are not inconsistencies in it. An inconsistency refers to a severe error which implies that some of the classes in the ontology cannot have instances (OWL individuals), and no useful knowledge can be inferred from the ontology. The inconsistency will result in false semantic understanding and knowledge representation.

For this reason, semantic reasoners should be used during the ontology development process. A semantic reasoner is a program that infers logical consequences from a set of explicitly asserted facts or axioms and typically provides automated support for reasoning tasks such as classification, debugging and querying, in order to identify inconsistencies in the analysed ontology.

For the VICINITY project, the Hermit² reasoner was used due to the fact that it is a sound and complete open source reasoner that can be used as a Protégé plugin [2]. To check that the VICINITY ontology network is consistent, Hermit was installed as a Protégé plugin. After the execution of the reasoner on the five ontologies that belong to the ontology network so far, namely VICINITY Core, Web of Things (WoT), WoT Mappings, VICINITY adapters, and VICINITY datatypes, no inconsistencies or uncertainties were found.

2.2. Ontology Pitfall Scanner! (OOPS!)

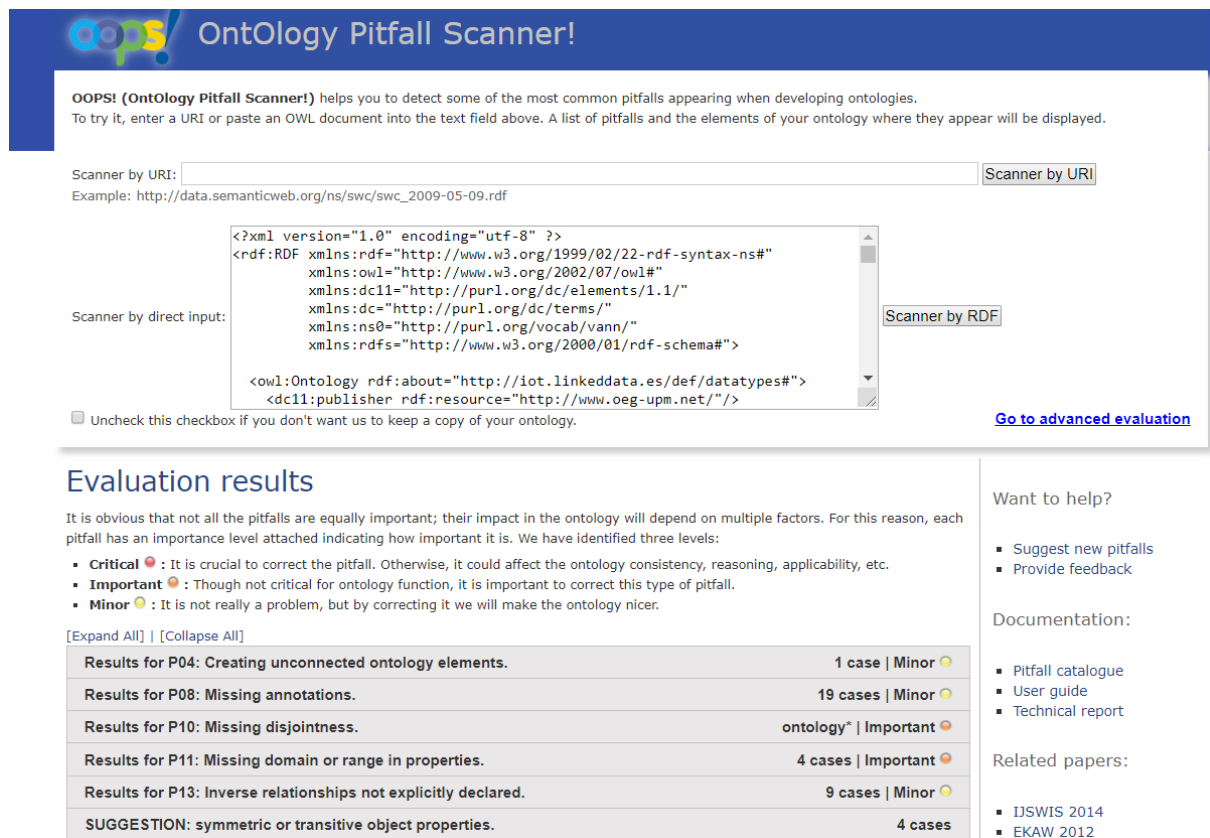
In order to identify if the ontologies are correctly built, it was decided to check whether there are modelling issues by using the OOPS! (Ontology Pitfall Scanner!) tool [3]. OOPS! represents a tool for diagnosing (semi-) automatically OWL ontologies. It allows to analyse ontologies to detect common pitfalls that appear during the ontology development process. Such catalogue of pitfalls is also available online, based on manually analysis of ontologies and literature reviews about ontology evaluation and Linked Data (LD). Each pitfall has an importance level attached indicating how important it is:

- **Critical:** It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- **Important:** Though not critical for ontology function, it is important to correct this type of pitfall.
- **Minor:** It is not really a problem, but by correcting it we will make the ontology in better form and understandable.

Figure 3 shows the OOPS! Interface, along with some of the obtained results after its scanner. OOPS! was used in all the VICINITY ontologies in order to identify modelling pitfalls. Table 1 shows the results of the OOPS! execution in the ontology network and the pitfalls related to the defined ontology. There

² <http://www.hermit-reasoner.com/>

are not critical pitfalls, even though there were found some important and minor pitfalls. These important pitfalls do not affect the consistency, reasoning or applicability of the ontology. Additionally, in this case they refer to “missing domain or range”, however, it was a modelling decision to not add domain or range to certain properties in order not to be restrictive with them. It is also shown the pitfall “Missing disjointness”, but it was also decided not to add disjoint classes since the restriction are not needed in the ontology. Regarding the minor pitfalls, they are mostly related to missing annotations and naming conventions and they will be corrected in future releases of the ontology network, together with the unconnected elements found in the ontology network. Regarding the pitfalls related to “Inverse relationships not explicitly declared”, the ontology developers will analyse each potential inverse relationship to add those considered necessary without adding too much overhead to the ontology.



OOPS! (Ontology Pitfall Scanner!) helps you to detect some of the most common pitfalls appearing when developing ontologies. To try it, enter a URI or paste an OWL document into the text field above. A list of pitfalls and the elements of your ontology where they appear will be displayed.

Scanner by URI: Scanner by URI
 Example: http://data.semanticweb.org/ns/swc/swc_2009-05-09.rdf

Scanner by direct input: Scanner by RDF

```
<?xml version="1.0" encoding="utf-8" ?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:owl="http://www.w3.org/2002/07/owl#"
  xmlns:dc11="http://purl.org/dc/elements/1.1/"
  xmlns:dc="http://purl.org/dc/terms/"
  xmlns:ns0="http://purl.org/vocab/vann/"
  xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#">
  <owl:Ontology rdf:about="http://iot.linkeddata.es/def/datatypes#">
    <dc11:publisher rdf:resource="http://www.oeg-upm.net/">
```

Uncheck this checkbox if you don't want us to keep a copy of your ontology. [Go to advanced evaluation](#)

Evaluation results

It is obvious that not all the pitfalls are equally important; their impact in the ontology will depend on multiple factors. For this reason, each pitfall has an importance level attached indicating how important it is. We have identified three levels:

- Critical** 🚨 : It is crucial to correct the pitfall. Otherwise, it could affect the ontology consistency, reasoning, applicability, etc.
- Important** ⚠️ : Though not critical for ontology function, it is important to correct this type of pitfall.
- Minor** 🟡 : It is not really a problem, but by correcting it we will make the ontology nicer.

[Expand All] | [Collapse All]

Results for P04: Creating unconnected ontology elements.	1 case Minor 🟡
Results for P08: Missing annotations.	19 cases Minor 🟡
Results for P10: Missing disjointness.	ontology* Important ⚠️
Results for P11: Missing domain or range in properties.	4 cases Important ⚠️
Results for P13: Inverse relationships not explicitly declared.	9 cases Minor 🟡
SUGGESTION: symmetric or transitive object properties.	4 cases

Want to help?

- Suggest new pitfalls
- Provide feedback

Documentation:

- Pitfall catalogue
- User guide
- Technical report

Related papers:

- IJSWIS 2014
- EKAW 2012

Figure 3: OOPS! interface

Table 1: OOPS! Results for the VICINITY ontology network

Ontology	OOPS! results		
	Critical	Important	Minor
WoT	-	P11: Missing domain or range	P04: Creating unconnected ontology elements P08: Missing annotations P13: Inverse relationships not explicitly declared P22: Using different naming conventions
Core	-	-	-
WoT Mappings	-	P11: Missing domain or range	P04: Creating unconnected ontology elements P08: Missing annotations P13: Inverse relationships not explicitly declared P22: Using different naming conventions
VICINITY adapters	-		P04: Creating unconnected ontology elements P08: Missing annotations
VICINITY datatypes	-	P10: Missing disjointness P11: Missing domain or range	P04: Creating unconnected ontology elements P08: Missing annotations P13: Inverse relationships not explicitly declared

3. Validation with regards to ontological requirements

The validation of ontologies before the online publication is a crucial part in the ontology development process, since it will guarantee that all the requirements asked by the domain experts are satisfied by the ontology after it is released online. In VICINITY, the testing process described in [4] was followed to check all the proposed requirements, in order to assure that the VICINITY ontologies are complete with regards to the asked requirements. Along this section such testing process and the tool to support it are described.

3.1. Testing method

The testing process followed in VICINITY is based on three activities, i.e., test design, test implementation and test execution. In the test design activity, the knowledge intended to be produced by every requirement is identified, e.g., from the requirement “A device can have a status” is expected a relation between two concepts in the ontology. To formalize such knowledge into a test case a collection of test expressions is used according to the requirements behaviour. During the test implementation activity such test expressions are implemented in order to be able to execute them on our ontologies to validate the associated requirement. The implementation of the test cases is based on the analysis of the behaviour of the ontology in different situations to verify that certain knowledge is modelled in the ontology, rather than simply checking the presence or absence of axioms using semantic reasoners. This is because the use of semantic reasoners is not enough to validate if a requirement is satisfied.

In addition to the testing activities, we propose an RDF vocabulary to store the generated test cases and to provide traceability between them and their associated requirements. Figure 4 summarizes the activities carried out in this testing process, together with their inputs and outputs. The following subsections describe in detail the activities in this testing process.

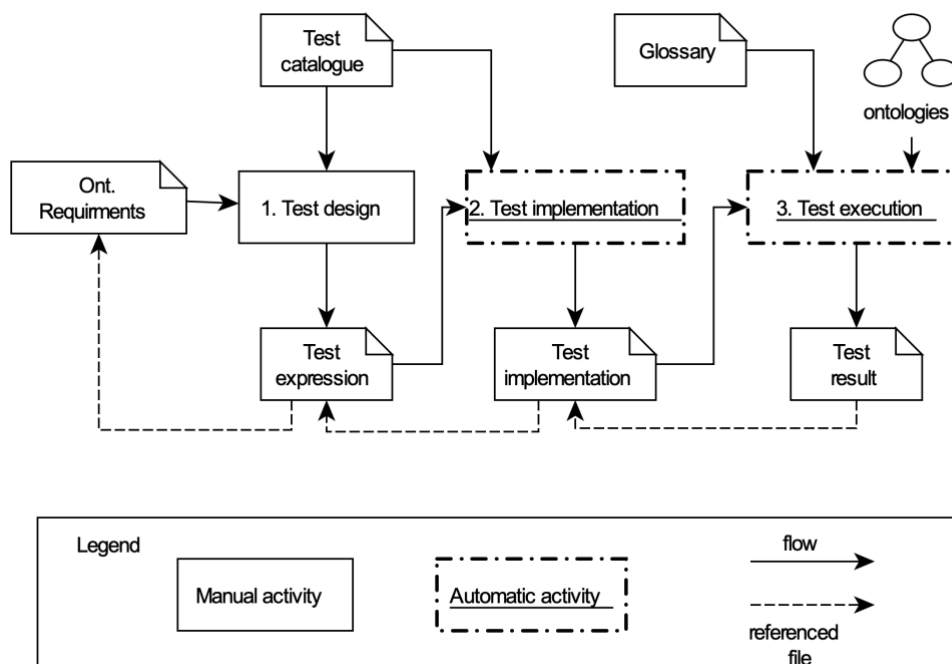


Figure 4: Testing method with inputs and outputs

3.1.1. Test design

During this activity the desired behaviour, i.e., the knowledge intended to be produced by every requirement in an ontology, of each requirement is extracted. In order to carry out this extraction, we provide a set of possible test expressions, extracted from the CORAL Corpus [5], which represent the requirements desired behaviour. This corpus is based on the NeOn modelling components [6], and analyses 834 ontology requirements in order to identify lexico-syntactic patters (LSPs) based on the goal that each requirement has regarding its implementation in an ontology, e.g., a relation between two concepts. Therefore, these LSPs indicate the implementation in the ontology associated with a requirement template. Each of these types of requirements is associated with a test expression, which represents the desired behaviour in a formal language based on the OWL Manchester Syntax. Table 2 lists the supported test expressions. To give an example, if the tester wants to check an equivalence relation between two terms, then the corresponding test expression to be used is *A EquivalentTo B*, where A and B represents the equivalent terms.

Table 2: Supported test expressions

Test goal	Test expression syntax
T1 Equivalence	A EquivalentTo B
T2 Subsumption	A SubClassOf B
T3 Disjointness	A disjointWith B
T4 Property between two concepts	A SubClassOf P some B
T5 Universal restriction	A SubClassOf P only B
T6 Multiple inheritance	A SubClassOf B and C
T7 Symmetry	A Symmetric(P) B
T8 Maximum cardinality	A SubClassOf P max [num] B
T9 Minimum cardinality	A SubClassOf P min [num] B
T10 Cardinality	A SubClassOf P max [num]B
T11 The ontology contains the individual	I type A
T12 Subsumption and relation between classes	A SubClassOf [ClassB] that [PropertyP] Some C
T13 Minimum cardinality and relation between classes	A SubClassOf [PropertyP] min [num]B and B SubClassOf [PropertyP] some C
T14 Minimum cardinaly and universal restriction	A SubClassOf [PropertyP] min [num]B and B SubClassOf [PropertyP] only C
T15 Definition of a disjoint set of classes	A SubClassOf B and C SubClassOf B that disjointWith A
T16 Participation of a concept in an event	A SubClassOf [participates] some B
T17 Co-Participation of two concepts in an event	A and B SubClassOf participatesIn some C

The output of this activity is an RDF document where the test cases are stored using the proposed testing vocabulary. In this vocabulary, each test case design stores the associated requirement URI,

the description of the requirement, and the desired behaviour specified by the test expressions allowing traceability between the artefacts in the ontology development process.

3.1.2. Test implementation

In order to implement the tests to verify if a desired behaviour is satisfied, a procedure is proposed. In this procedure the test design is formalized into a precondition, a set of auxiliary term declarations and a set of assertions to check the behaviour. The procedure to implement each test expression defined in Table 2 is further described in [4].

The precondition is a SPARQL query which checks whether the terms involved in the ontology requirement are defined in the ontology. In order to execute the tests, these terms need to be declared in the ontology. Otherwise, the test fails, and the requirement is not satisfied. The axioms to declare auxiliary terms are a set of temporary axioms added to the ontology to declare the auxiliary terms needed to carry out the assertions. After the addition of these axioms the reasoner is executed and, in order to be able to check the behaviour, the ontology needs to be consistent. Finally, the assertions to check the behaviour are a set of pairs of axioms and expected results that represent different ontology scenarios. For each pair, the axiom is temporary added to the ontology to force a scenario, after which the reasoner is executed. The expected result determines if the ontology status after the addition (i.e., inconsistent ontology, unsatisfiable class or consistent ontology) is the expected one in case the requirement was satisfied. If all the status concurs with the expected status, then the requirement is satisfied.

As an example, to check equivalence between two concepts ($A \text{ EquivalentTo } B$), we define a set of auxiliary terms, i.e., the classes that complement A ($\neg A$) and B ($\neg B$). After their definition, we define a set of assertions that force the ontology to present unsatisfiable classes or inconsistencies. First, it is generated a class A' that is defined as a subclass of class B and $\neg A$. If the ontology satisfies the requirement, this addition causes an unsatisfiable class due to the fact that the reasoner would infer that A' is subclass of A and $\neg A$. Then, it is generated a class A' that is defined as a subclass of class A and $\neg B$. If the ontology satisfies the requirement, this addition causes an unsatisfiable class due to the fact that the reasoner would infer that A' is subclass of B and $\neg B$. The last assertion, generates a class A' that is defined as a subclass of class A and B . If the ontology satisfies the requirement, this assertion causes a consistent ontology due to the fact that there is no problem if A' is subclass of A and B .

The output of this activity is an RDF document where the test cases are stored using the proposed vocabulary. In this vocabulary, each test case implementation stores the associated test design; the test preparation, which represents auxiliary terms declaration; and the corresponding test assertions.

3.1.3. Test execution

Finally, the test execution activity consists in three steps, namely, the execution of the query which represents the preconditions, the addition of the axioms which declare the auxiliary terms, and the addition of the assertions. After the addition of each axiom, the reasoner is executed to report the status of the ontology. The addition of the auxiliary axioms needs to always lead to a consistent ontology. In the case of the assertions, the agreement between the reasoner status after the addition

of all the axioms and the status indicated in the test implementation determines whether the ontology satisfies the desired behaviour. We distinguish four possible results:

- **Undefined terms**, if the ontology does not pass the preconditions.
- **Passed**, if the ontology passes the preconditions and the results of the assertions are the expected ones.
- **Absent relation**, if the ontology passes the preconditions, the results of the assertions are not the expected ones, but the addition of the requirements would not lead to an inconsistency in the ontology.
- **Conflict**, if the ontology passes the preconditions, the results of the assertions are not the expected ones and the addition of the requirements would lead to an inconsistency in the ontology.

It is worth mentioning that during this activity it is also carried out a mapping between the term identified in the test implementation and the actual term in the ontology where the ontology is going to be executed. The mapping at this stage of the testing process allows to execute the same tests on multiple ontologies. Figure 5 shows the steps carried out in this activity.

As an example, if we want to check equivalence between two classes Sensor and Actuator, the test will result in *undefined terms* if the ontology does not define a class named Sensor or a class named Actuator. Additionally, the test will result in an *absent relation* if the ontology defines the classes named Sensor and Actuator but does not define any relation between them, and the test will result in a *conflict* if the defined relation in the ontology between the classes Sensor and Actuator is a disjoint instead of an equivalence. However, if the ontology to be tested defines an equivalence relation between the classes Sensor and Actuator, then the test will be *passed*.

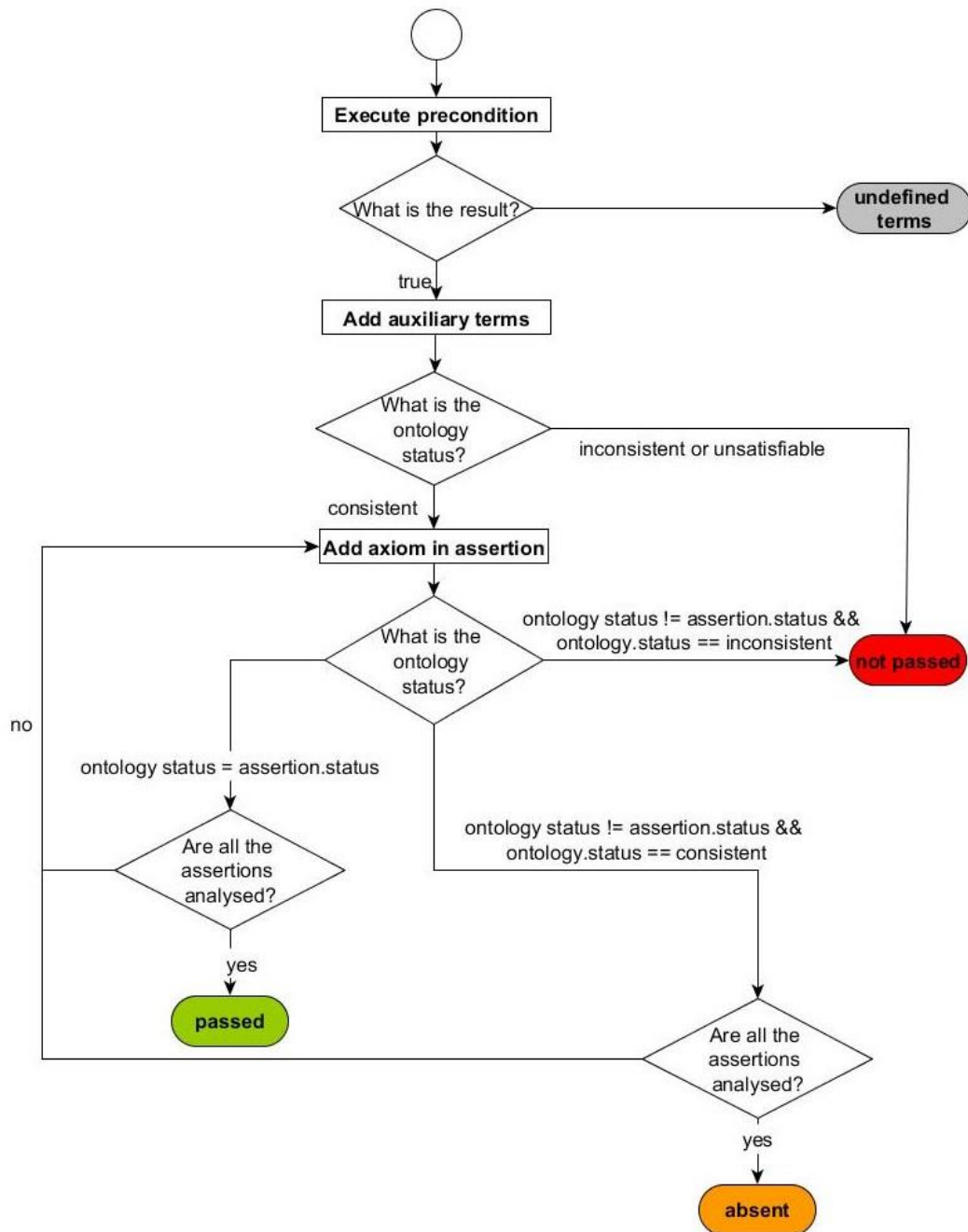


Figure 5: Test execution steps

3.2. Testing infrastructure

This testing process is supported by the tool Themis, which is a web application that is available online.³ As it is shown in Figure 6, Themis supports and automates the test implementation and execution activities.

³ <http://themis.linkeddata.es/>

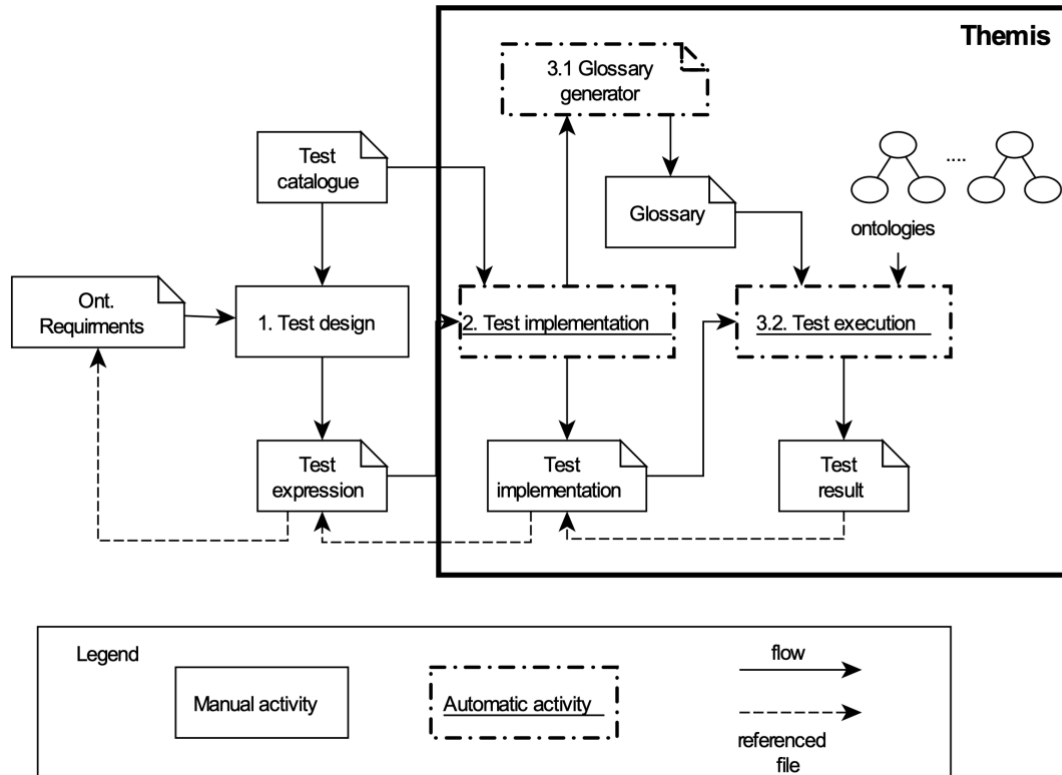


Figure 6: Themis and its relationship with the testing method

This tool allows to validate a requirement by using the test expressions listed in Table 2. It is possible to add the test expression to the interface by means of two mechanisms: (1) by adding directly the test expression to the interface or (2) by loading an RDF file that is available online with the test suite. As mentioned before, to execute the tests on an ontology it is required a mapping between the terms in the tests and in the actual ontology. To do so, Themis proposes a glossary of terms where the terms are extracted from the URI fragments of each concept. If needed, the users can modify this glossary to adapt it to their use case.

The interface also shows the results for each of the test expressions. The possible results are the same as the results described in the previous section, namely, undefined terms, passed, absent relation, and conflict. Figure 7 shows an example of the possible results.

Themis Home How to use it REST Service

Execute tests. Validate your ontology!

This tool helps you to validate your ontology model by executing tests. To try it, you need first to load the ontology to be validated and then define and check the tests. The result of each test will appear in the "Test results" section.

- Load the ontology**
Add the URI of the ontology to be validated:

 [http://iot.linkeddata.es/def/core](#) [Glossary of terms](#)
- Check the tests**
You can load a test suite from a URL:

Or you can add the tests directly. To add more than one test separate them by using ",".
 The following link shows all the supported tests. In this other link you can also find some examples that can be useful to propose tests.

Tests results

Test	Result	Problem
Device subclassOf Building	Absent relation	The ontology does not implement the requirement associated to the test
Device disjointWith Sensor	Conflict	The ontology has a relation which causes a conflict with the one define in the test
SmartHouse subclassof Thing	Undefined terms	The terms in the test are not defined in the ontology
Thing subclassOf providesInteractionPattern only InteractionPattern	Passed	None

Figure 7: Themis interface

Themis also allows to export the test expressions added in the interface as RDF files, which can be uploaded online to reuse them over the same or other ontologies.

```

### http://www.semanticweb.org/untitled-ontology-53#testDesign1
:testDesign1 rdf:type <http://w3id.org/def/vtc#TestCaseDesign> ,
              owl:NamedIndividual ;
              <http://w3id.org/def/vtc#desiredBehaviour> "Device subclassOf Building"^^xsd:string .

### http://www.semanticweb.org/untitled-ontology-53#testDesign2
:testDesign2 rdf:type <http://w3id.org/def/vtc#TestCaseDesign> ,
              owl:NamedIndividual ;
              <http://w3id.org/def/vtc#desiredBehaviour> "Device disjointWith Sensor"^^xsd:string .

### http://www.semanticweb.org/untitled-ontology-53#testDesign3
:testDesign3 rdf:type <http://w3id.org/def/vtc#TestCaseDesign> ,
              owl:NamedIndividual ;
              <http://w3id.org/def/vtc#desiredBehaviour> "SmartHouse subclassof Thing"^^xsd:string .
    
```

Figure 8: Test case in RDF exported by Themis

3.3. Testing results

Table 3 summarizes the number of requirements defined for each ontology in the VICINITY ontology network, as well as the provenance and the number of requirements which were implemented, discarded or pending. These requirements represent the needs asked by the domain experts in order to model the VICINITY platform. As shown in the table, there are no pending requirements, which means that all the asked needs were implemented. Moreover, it shows that several requirements were discarded. This deletion of requirements was done because those requirements were no longer needed in the ontology or their definition was not correct. As an example of deletion, the requirement “What is a WoT interface?” for the WoT ontology was considered obsolete and WoT interface term replaced by the Endpoint term. The discussion regarding this deletion is stored in the Github issue tracker. ⁴

Table 3: Summary of requirements categorization for the VICINITY ontology network

Ontology	Extracted from	Ontological requirements ⁵			
		Defined	Implemented	Discarded	Pending
Core	<ul style="list-style-type: none"> D1.5 VICINITY Bratislava and later meetings/emails with partners Partners’ devices characterizations 	190	50	140	0
WoT	W3C Web of Things IG	35	16	19	0
WoT Mappings	<ul style="list-style-type: none"> Gateway API Developers 	16	15	1	0
WoT adapters	<ul style="list-style-type: none"> Partners’ meetings Partners’ devices characterizations 	154	154	0	0
WoT datatypes	Partners’ meetings	11	11	0	0
		406	246	160	0

From the set of ontological requirements defined for each ontology, test cases were extracted in order to validate the ontology. Such test cases were generated by using the test expression catalogue provided in Section 3. Each ontological requirement is translated to one or more test expressions, selecting the more appropriate ones from the test catalogue. Several test cases can be related to the same requirement. Table 4 displays the number of tests generated for each ontology.

⁴ See <https://github.com/mariapoveda/wot-ontology/issues/5> for the discussion of the deletion of the WoT interface term.

⁵ See <http://vicinity.iot.linkeddata.es/vicinity/testing.html> for the online version of the VICINITY ontologies requirements.

It is worth mentioning that the test suites, i.e., the sets of test cases associated to each ontology, were exported to RDF files and uploaded to the VICINITY ontology portal. These online RDF files can be reused in future releases of the ontology to assure that all the previous requirements are still satisfied. The test cases are stored using the Verification Test Case ontology⁶ to describe their properties. Each RDF file includes, in addition to the set of test cases, the provenance of the test suite, i.e., the ontology from which the test cases were extracted. Each test case is also linked to the URI of the associated requirement and can also include the competency question of such associated requirement in order to improve the readability of the test case.

Once all the tests were defined, Themis was executed during the ontology development process in order to identify if there are tests that are not passed by the ontology. Table 4 summarizes the last obtained results of such execution, showing that all the requirements are satisfied. As the table shows, most of the requirements are passed by the ontology. However, there is one test whose results are absent relation, which means that the ontology passes the preconditions, the results of the assertion are not the expected ones but there are no conflicts in the ontology. This result warns the ontology developers that there is a test that, even though they do not cause any conflict in the ontology, they are not implemented, at least completely, in the ontology. After analysing such absent relation, which is related to the requirement “*A thing implements security*” in the ontology, the ontology developers finally decided to maintain the ontology as it was, due to the fact that they do not want to add restrictions to the property *implementsSecurity*. It is worth mentioning that sometimes, due to modelling decisions, absent relation results can be obtained, and that does not represent that the requirements are not satisfied by the ontology, since the developers decided not to add that restriction to the ontology.

Table 4 Tests executed for each VICINITY ontology

Tests results						
Ontology	Number of tests	Undefined terms	Passed	Conflict	Absent relation	
WoT	16	0	15	0	1	
Core	50	0	50	0	0	
WoT Mappings	15	0	15	0	0	
WoT adapters	154	0	154	0	0	
WoT datatypes	13	0	13	0	0	
	248	0	247	0	1	

⁶ <https://w3id.org/def/vtc#>

4. Validation with regards to pilots' data

In the context of the VICINITY project we have devised an ontology that defines how data must be modelled. Nevertheless, the data used in the platform can use terms from our ontology or even use other terms that are not defined in our ontology; in that cases it is relevant to check whether the terms are made up or actually the ones from the specified ontology. In addition, there are other facts that should be checked or quantified relying on the data.

Figure 9 summarizes the two identified main types of requirements that are necessary to consider and analyse when validating the data in VICINITY relying on the VICINITY ontology network:

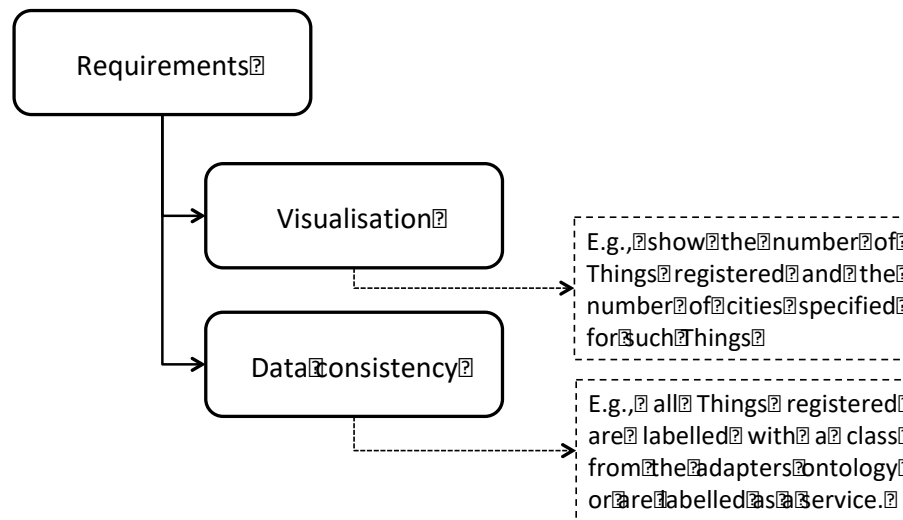


Figure 9: Type of requirements considered by our system

- **Visualisation requirements:** these requirements are meant to display their results in a chart or a table. Usually these requirements are related to data completion. For instance, in VICINITY an IoT infrastructure can be registered without specifying in which city is located, although it is mandatory to specify that information (unless due to some privacy policies such information should not be disclosed).
- **Data consistency requirements:** these requirements are related to how data is modelled considering an ontology. The result of these requirements is a boolean value. For instance, in VICINITY we can specify as requirement that all the terms used in data must belong to the ontology devised in VICINITY. The output of this requirement can only be true or false.

Bearing in mind these types of requirements we have devised a validation system called Things Monitor. Our system aims at checking and validating a set of requirements provided by an expert. In addition, we have implemented a dashboard manager that shows the results of all the requirements established and maintains the privacy restrictions of the VICINITY platform by implementing an authentication system.

Figure 10 recaps the system that we have developed. First, an expert provides a set of requirements. Next, such input is computed by our Monitor component, which stores in a database the results of the

validation. Finally, the dashboard manager component reads the results and draws the corresponding charts. Our Things Monitor also provides the specification required to draw the charts.

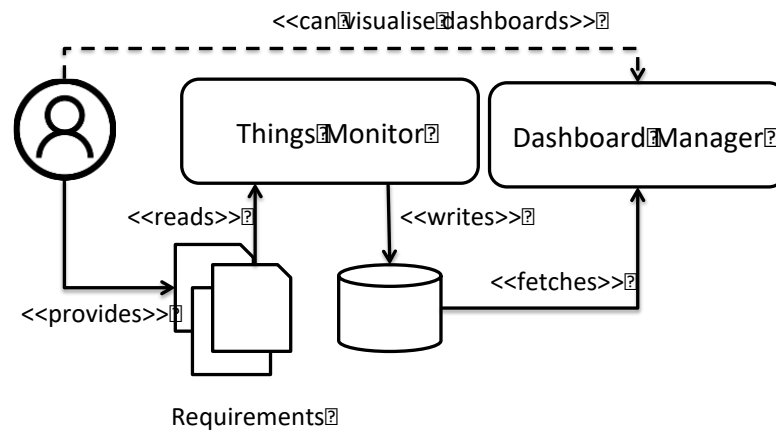


Figure 10: Architecture of our validation approach

4.1. Methodology for validating VICINITY

To validate VICINITY, on the one hand, we provided a set of general requirements that the platform must fulfil and, on the other hand, we provided a set of requirements that the data of each account in VICINITY must meet; for instance, the accounts of the pilots. The requirements provided are namely for visualisation.

For the overview of the platform we have the following requirements:

- Evolution of Things in VICINITY: is meant to display how the platform evolves showing the Things registered over time, the adapters, services, and other elements related to those Things.
- Things in VICINITY: is meant to display the number of Things registered in VICINITY, and from those, which are labelled as adapters or services, or on the contrary have a too generic type. In addition this requirement shows the number of Things that were correctly described in order to be used by the Semantic Interoperability services.
- Things owned by an organisation: is meant to display for each account in the VICINITY platform the number of Things owned by such account.
- Accessibility of interaction patterns: is meant to report the number of Things that are accessible, either to read or write.
- Types of interaction patterns: is meant to report the type of interactions that the Things have, i.e., if they expose data (properties), trigger events (event), or an order can be written in the thing (actions).
- Things with contextual information: is meant to report which Things were registered in the platform with contextual data, i.e., the building or building space where are located, their city or country, and their organisation.
- Type of Things in VICINITY: is meant to display the type of Things, i.e., which kind of sensors and devices were registered in the platform.
- Properties observed in VICINITY: is meant to report the physical magnitudes observed by the IoT infrastructures registered in VICINITY.
- Organisations in VICINITY: is meant to report the organisations registered, as well as the contracts in which they are involved to share or provide data.

For the each of the accounts we have the following requirements:

- Organisation's Things: is meant to report the number of Things registered by this organisation, and which of those where labelled with a type from the adapters ontology or as a service. In addition, this requirement report as well if the Things have the proper annotations so they can be used in the Semantic Interoperability Services.
- Type of Things: is meant to report the type of things registered by this organisation.
- Contracts held: is meant to report the contracts held by this organisation, either to expose data, or to read data.
- Things with contextual information: is meant to report from the Things registered in the VICINITY belonging to this organisation in particular, how many have data about their location, i.e., building, building location, city, or country.

To validate our requirements, we provided a manual to the partners so they can check whether their accounts meet the requirements. The validation is done by the partners, who need to check that their charts have the optimal values (specified in the manual). In this case, we have no strong constraints that can be evaluated as passed or failed; on the contrary the requirements encode desirable values that the platform should meet and due to this reason the validation depends as well by the partners.

4.2. Things Monitor implementation

The Things Monitor is a VICINITY component, and due to this reason, is located in the GitHub of the project <https://github.com/vicinityh2020/tmonitor>; due to privacy issues, the manual is attached as a confidential document. Finally, the deployed version of the dashboard manager can be found at <http://monitor.vicinity.linkeddata.es>. However the access is not public due to privacy policies.

4.3. Results of pilots' data validation

In this section we show the results of analysing how the pilots used the ontology. Our analysis is divided in several parts, first we are interested in know how many Things registered by the pilots where described using detailed types (from the Adapters ontology) instead of rather general (from the core). In addition to this we aimed at knowing how many of such Things where registered with the necessary data to be accessible through the semantic interoperability services. For this matter Figure 11 shows that out of 564 Things 387 where labelled with types from the Adapters ontology and 121 as services (which have no more detailed type). In conclusion only 56 Things were labelled using generic types, which is not necessary mistaken depending on the devices. Answering our second question 43 out of 564 objects were labelled with the required information to allow such Things to be accessed by the semantic interoperability services.

In second place, we aimed at analyse the different interaction patters used by the registered things. We wished to know how many of them are either properties, actions, events or they were not labelled properly. In addition, we aimed at knowing how many properties have an observation from the ontology, and how many reported units of measure. On the one hand, Figure 12 shows that out of 1158 interaction patterns 1021 are properties, 26 actions, and 111 events; however no interaction pattern reports any unit. Regarding the accessibility of data, and the observations Figure 13 shows that out of 1158 interaction patterns 1132 were actually labelled with an observation from the Adapters

ontology. In addition, 1040 were labelled as readable, 190 as writable, and 1047 as accessible relying on the WoT ontology.

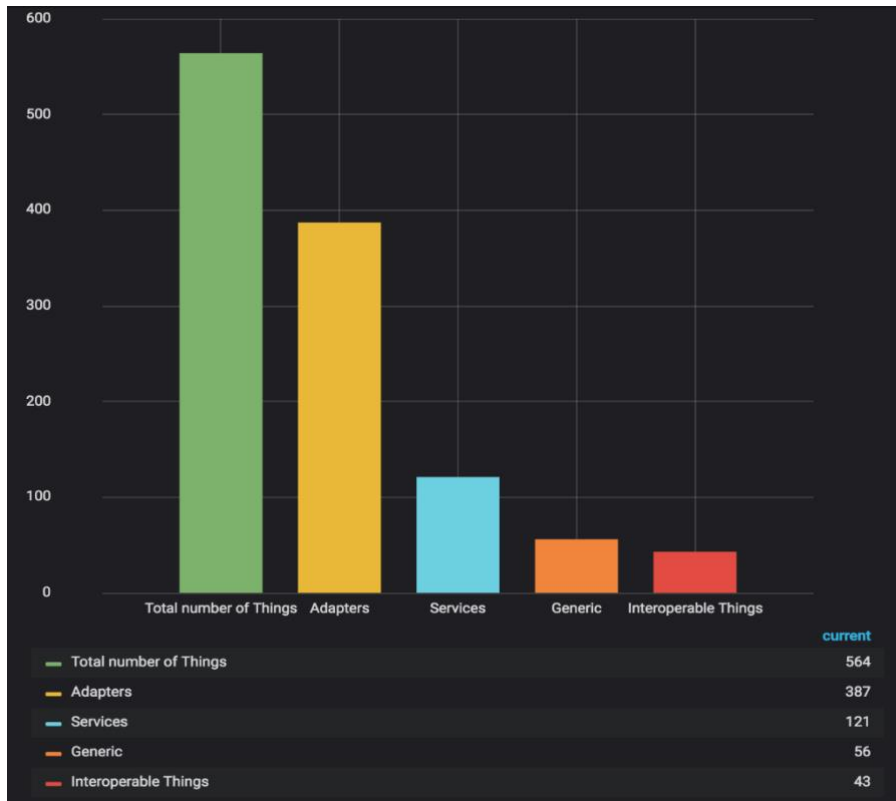


Figure 11: Things from pilots with detailed descriptions and interoperable

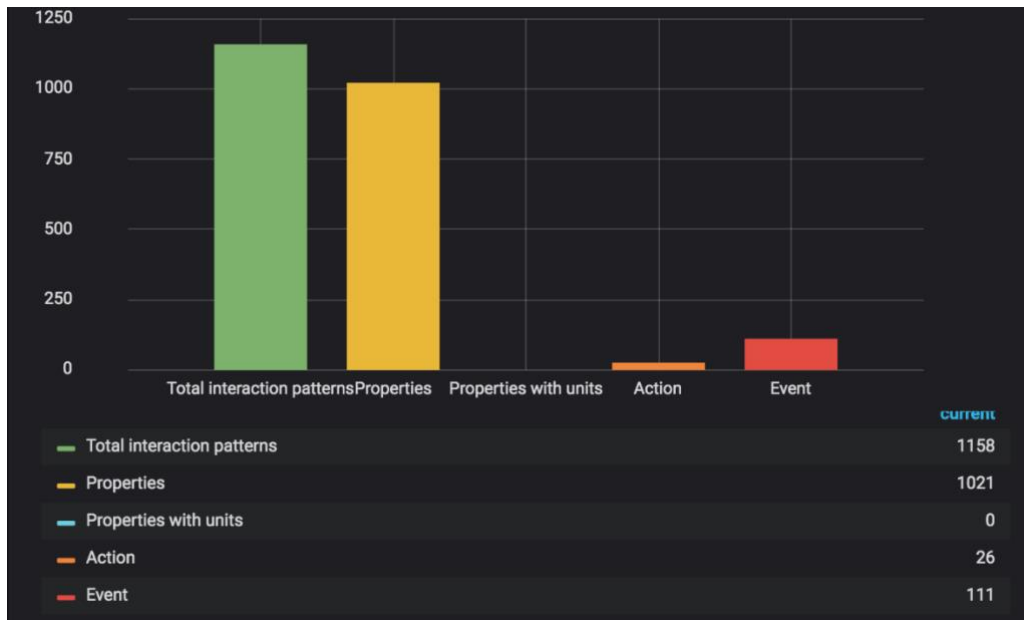


Figure 12: Interaction patterns

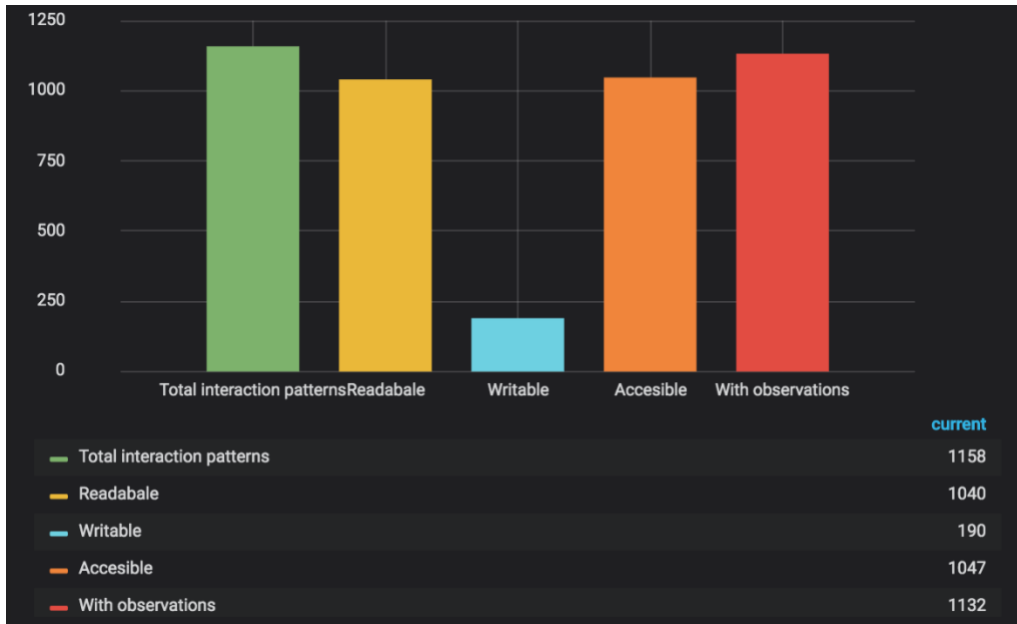


Figure 13: Accessibility of interaction patterns

In third place we aimed at listing the types used from the Adapters ontology, for this matter Figure 14 shows the listing of types used by the Pilots.

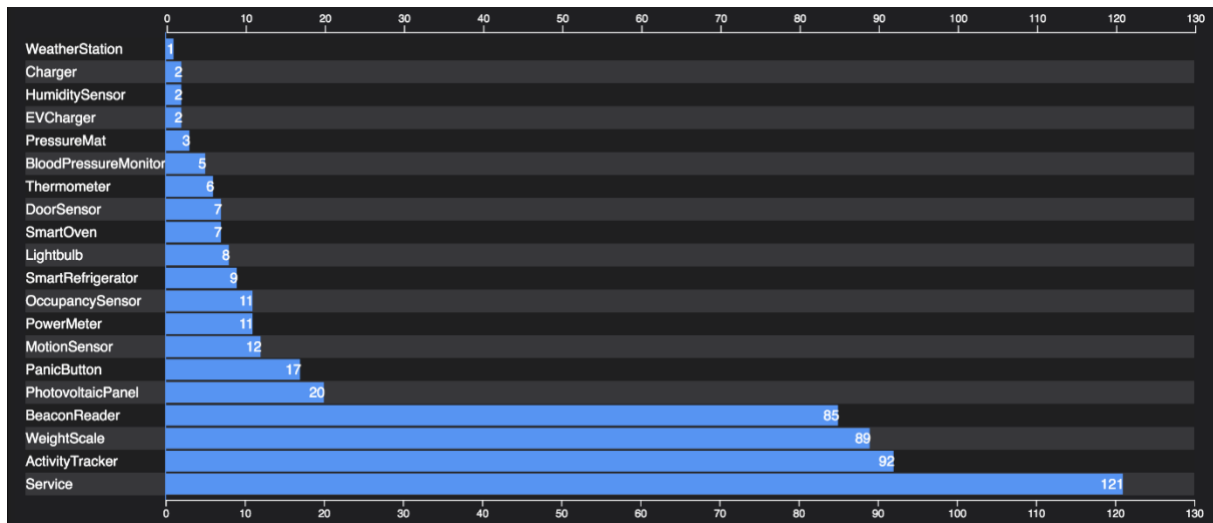


Figure 14: Types from Adapters used by Pilots

Finally, we aimed at analysing the usage of the ontology to describe contextual data, i.e., SAREF for Buildings and SAREF for City. Figure 15 shows that the pilots out of 564 Things used Saref for Buildings only in 18 Things, whereas SAREF for City 42. In addition, they used the core ontology to specify organisations in 45 Things out of 562.

As a conclusion, we see that the pilots relied namely on the Core, WoT, and Adapters ontologies. Instead the ontologies of SAREF and the WoT mappings were scarcely used.

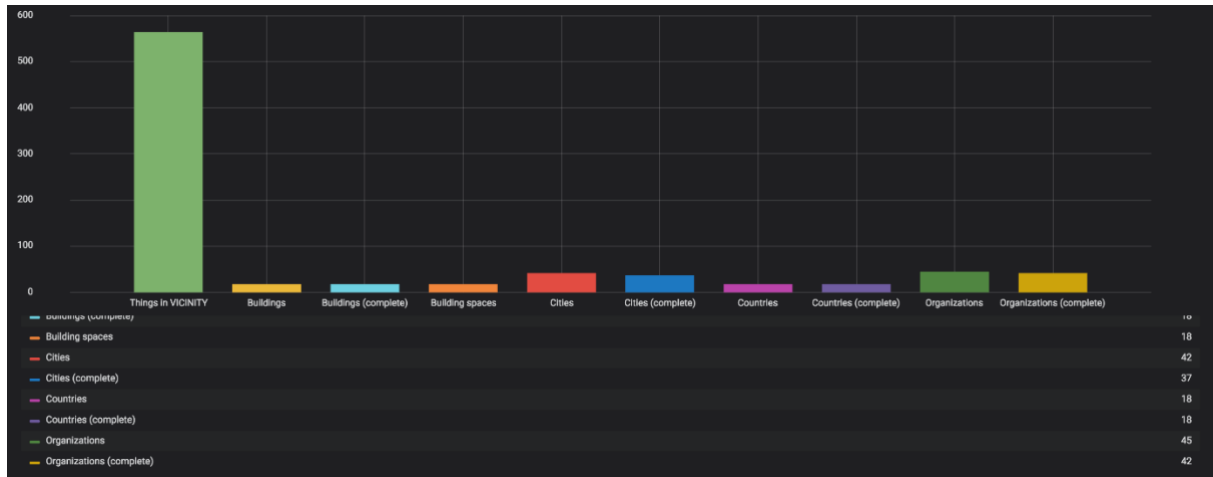


Figure 15: Use of SAREF by the Pilots

5. Validation with regards to standards

In addition to the validation of the requirements asked by the domain experts and users, the VICINITY ontology network was validated against the requirements of several IoT standards, in order to reuse concepts and patterns of well-known resources, namely: (1) the ETSI SAREF ontology [7], (2) the W3C SSN ontology,⁷ (3) OCF standards,⁸ (4) the oneM2M ontology [7], and (5) ISO/IEC 30141:2017 [8]. This validation allows to analyse the coverage regarding the ontological commitments of such standards and the VICINITY ontology.

Along this section the description of the method to carry out this coverage analysis is presented, in addition to the results analysis to check the coverage between the VICINITY ontologies and the already mentioned standards.

5.1. Coverage analysis method

To carry out such coverage analysis, a systematic approach has been followed. First, the set of models is selected according to the domain to be analysed. Then, the ontological requirements of these selected standards are gathered. Afterwards, the requirements must be translated into test expressions, as it is described in Section 3, in order to be implemented and executed.

As it is shown in Figure 16, this coverage method includes the already defined testing process (see Section 3), even though it goes further in the validation by executing different ontological requirements coming from different documents and including a test results analysis activity. Therefore, once the results are obtained and it is possible to be aware of which requirements from each input model are satisfied by the ontology, an analysis of the obtained results should be accomplished.

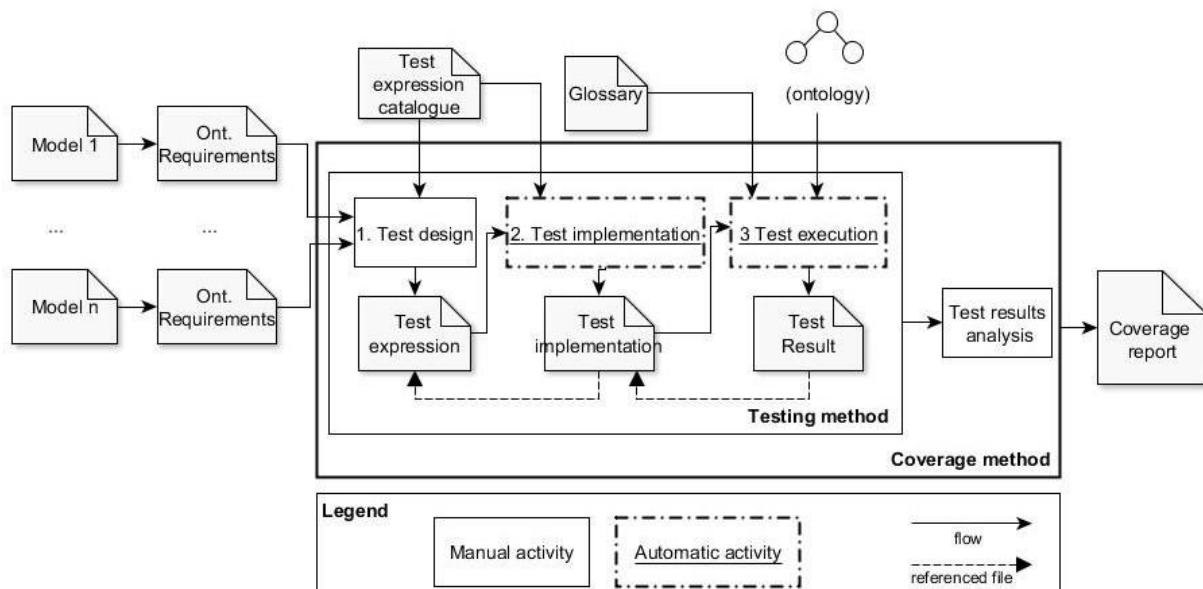


Figure 16: Coverage analysis with inputs and output

⁷ <http://www.w3.org/ns/ssn>

⁸ <https://openconnectivity.org>

5.1.1. Test results analysis

Once the results of the testing method are available, i.e., which requirements are passed or not passed, an analysis of such results should be carried out. This analysis includes a set of steps that should be followed:

1. Identify the topics of potential overlap between the input models and the ontology to be analysed, e.g., devices or users.
2. Identify for each pair input model and ontology: (1) the similarities in the requirements, (2) the additional information that is in the ontology but not in the input models and vice versa, and (3) the incompatibilities between the ontology and the input models.

This analysis of the results is accomplished in order to provide the users, domain experts and ontology developers with some indicators, including:

- The number of requirements passed and not passed by the ontology, emphasizing which requirements generate a conflict between a standard and the ontology.
- The concepts where there is an overlap between the input models and the ontology to be analysed.
- The relation and identification of incompatibilities in the concepts where there is an overlap.

These results can be used by these actors, i.e., users, domain experts and ontology developers, to provide feedback for the developed ontology, by means of requesting changes or by means of identifying inconsistencies between input models and the developed ontology. Moreover, it can also be used to identify mappings and potential terms for reuse.

5.2. Coverage analysis infrastructure

For this coverage analysis we have also proposed Themis as the tool to execute the tests on the ontologies. Themis allows to load a test file and execute the test suite on multiple ontologies, as shown in Figure 17. Therefore, it is possible to load the test suites for each of the input models and to execute all of them on the ontologies to be analysed, i.e., the VICINITY ontology network.

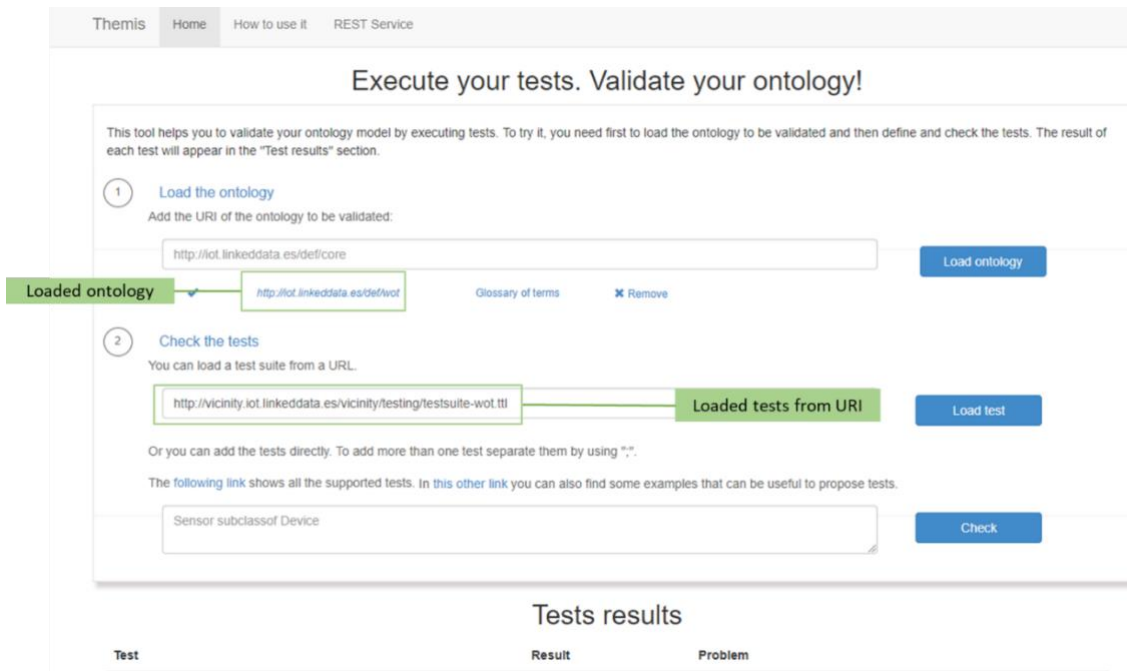


Figure 17: Themis interfaces for loading test suites from files

However, as it is shown in Figure 6, Themis only supports the test implementation and test execution activities and, therefore, the test results analysis is out of scope of this tool. To carry out such analysis a manual procedure is proposed where the results are visualized in a table. In this table it is needed to show the following fields:

- The test case
- The provenance of each test case, i.e., the requirement associated
- The testing result associated to the test case
- The topic associated to the test case

An example of this type of report is shown in Figure 18, which was also stored as an HTML file in order to improve the visualization and to publish it online.

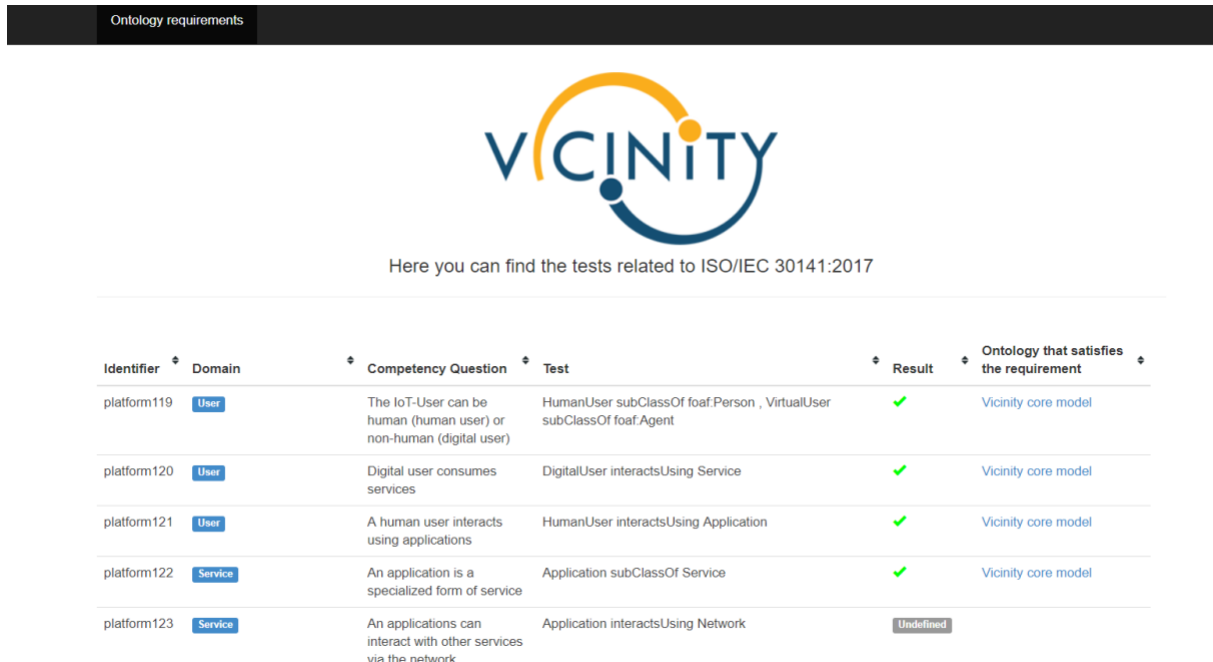


Figure 18: Example of coverage report

From this report it is easy to extract additional information, such as the topic of the requirements that are passed, which requirements conflict with the ontology, as well as the percentage of requirements that are passed.

Once the report is generated, a table with more detailed information about the topics where there is an overlap can be generated. Such tables can include relations between the requirements, e.g., a requirement is contained in another or two requirements are incompatible, and the relation between concepts in several ontologies.

5.1. Testing results

This section will describe the obtained testing results related to the conformance between the VICINITY ontology network and several IoT standards. Two analyses have been made: (1) the coverage of the VICINITY ontology network with respect to the IoT standards, and (2) the coverage of the IoT standards with respect to the VICINITY network.

As previously mentioned, in addition to the VICINITY requirements asked by the partners, the VICINITY ontology network should satisfy the requirements of several standards in IoT, namely: (1) the SAREF ontology, (2) the SSN ontology, (3) OCF standards, (4) the oneM2M ontology, and (5) ISO/IEC 30141:2017. Therefore, the requirements related to these standards were collected, and the associated tests were defined by using the test expression catalogue as it is depicted in Table 2. Table 5 shows the list of standards, the provenance of the gathered requirements associated to such standards, and the number of defined ontological requirements per standard. As it is shown, some of the requirements were extracted from official documentation, e.g., Technical Reports, while others were extracted from web sites. It is worth mentioning that some requirements, due to their complexity, were translated into several tests.

Table 5: Summary of requirements information for the IoT standards

Ontology	Version	Extracted from	Defined requirements	Defined tests
ETSI SAREF	v2.1.2	SAREF extension investigation Technical Report (TR 103 411) [7]	70	70
W3C SSN	V2.0	W3C SSN Specification ⁹	24	34
OCF	v2.0.2	OCF Core Specification [9]	27	27
OneM2M	v3.6.0	SAREF extension investigation Technical Report (TR 103 411) [7]	33	33
ISO/IEC 30141:2017	-	ISO/IEC 30141:2017:Internet of Things (IoT) – Reference Architectures [8]	36	36

Themis was also used in order to check if the VICINITY ontology network satisfies the test expressions defined for the requirements related to these IoT standards. As the test cases of such standards were generated following the testing method proposed in Section 3, the test expressions do not have any information of ontologies such as URIs, making them independent of the ontology from which they are extracted. Therefore, these test expressions can be executed on the VICINITY ontology network. Table 6 shows the testing results of the requirements related to each standard after Themis execution on the VICINITY ontology network. All the tests are stored in the VICINITY ontology portal.¹⁰

⁹ See <https://www.w3.org/TR/vocab-ssn/> for the W3C documentation.

¹⁰ See <http://vicinity.iot.linkeddata.es/vicinity/conformance.html> for the online version of the requirements and tests related to the IoT standards.

Table 6: Summary of testing results for the IoT standards

Analysed standard	Tests results					Relevant concepts
	Total number of tests	Number of test with undefined terms result	Number of test with passed result	Number of test with conflict result	Number of test with absent relation result	
ETSI SAREF ¹¹	70	66	4	0	0	Device
W3C SSN ¹²	34	10	9	0	15	Sensor, System, Procedure, Property, Feature of Interest, Actuator, Deployment
OCF ¹³	27	21	6	0	0	Device
OneM2M ¹⁴	33	28	4	0	1	Device and Thing
ISO/IEC 30141:2017 ¹⁵	36	22	14	0	0	Thing and User
	200	147	37	0	16	

From the execution of Themis, it could be deduced that the VICINITY ontology network satisfies several tests, but did not take into consideration some concepts related to these standards because the developers did not consider it necessary for the project domain definition. As an example, several concepts defined in the SAREF ontology are considered out of scope in the VICINITY domain, such as the *Command* or *Function* concepts, and, therefore, they are not included in the ontology. However, it could also be concluded that there were no conflicts between the VICINITY ontology network and these ontologies and standards. This last statement refers to the fact that, even though there are several undefined terms in VICINITY ontologies, there are no inconsistencies between the domain

¹¹ See <http://vicinity.iot.linkeddata.es/vicinity/testing/report-saref-results.html> for the online version of the SAREF ontology requirements.

¹² See <http://vicinity.iot.linkeddata.es/vicinity/testing/report-ssn-results.html> for the online version of the SSN ontology requirements.

¹³ See <http://vicinity.iot.linkeddata.es/vicinity/testing/report-ocf-results.html> for the online version of the OCF requirements.

¹⁴ See <http://vicinity.iot.linkeddata.es/vicinity/testing/report-onem2m-results.html> for the online version of the One2M2M ontology requirements.

¹⁵ See <http://vicinity.iot.linkeddata.es/vicinity/testing/report-isoiec30141-results.html> for the online version of the ISO IEC 30141 ontology requirements.

defined in the standards and the domain defined in the VICINITY ontologies. More detailed results related to the execution of the test cases of these standards is available online in the VICINITY ontology portal.¹⁶

In addition to the coverage of the VICINITY ontologies regarding the IoT standards, it has also been analysed the coverage of the IoT standards regarding the VICINITY ontology network with the aim of identifying whether there is knowledge defined in VICINITY that is not defined in none of the standards, and to identify overlaps. Therefore, the test associated to the five ontologies that belong to the VICINITY network, i.e., the VICINITY Core (Core), the Web of Things (WoT), the WoT mappings (Mappings), the VICINITY Adapters (Adapters), and the Datatypes (Datatypes) ontologies, were executed on the IoT standards. Since to follow the coverage method process depicted in Figure 16 **Error! Reference source not found.** it is needed to have an ontology to execute the test cases, we could only analyse those standards that have an ontology associated. These standards are the SAREF, the W3C SSN and the oneM2M ontologies. The following tables summarize the testing results of such standards with the tests of the VICINITY ontology network.

First, Table 7 shows that the SAREF ontology satisfies 12 requirements from the VICINITY ontology network, which are related to Device and Thing concepts.

Table 7: Testing results for SAREF ontology regarding to VICINITY requirements

Analysed standard	Tests results					Relevant concepts
	Total number of tests	Number of test with undefined terms result	Number of test with passed result	Number of test with conflict result	Number of test with absent relation result	
WoT	14	13	1	0	0	Thing
Core	50	43	7	0	0	Device
WoT Mappings	15	15	0	0	0	-
WoT adapters	154	150	4	0	0	Adapter, Property
WoT datatypes	13	13	0	0	0	-
	246	234	12	0	0	

¹⁶ See <http://vicinity.iot.linkeddata.es/vicinity/conformance.html> for the online version of the requirements and tests related to the IoT standards.

To identify in more detail what is the overlap and in which concepts there is a relation between SAREF and the VICINITY ontology network the analysis steps proposed in Section 5.1.1 **Error! Reference source not found.** were followed. Additionally, such analysis was completed with the information obtained from Table 6, which provides the results of the tests cases associated to the standards that are passed by the VICINITY ontologies. With all this information, it was extracted which are the relevant concepts where there is a potential overlap between the ontologies, i.e., Device and Thing. Therefore, the tests related to Device and Thing topics of both the VICINITY and the SAREF ontologies are analysed separately. Table 8 summarizes the number of tests and the tests results for SAREF and VICINITY ontologies together, joining the results of Table 6 and Table 7. Additionally, it shows the relevant concepts between these SAREF and VICINITY ontologies.

Table 8: Testing results for SAREF and VICINITY ontologies

Analysed standard	Tests results					Relevant concepts
	Total number of tests	Number of test with undefined terms result	Number of test with passed result	Number of test with conflict result	Number of test with absent relation result	
SAREF and VICINITY	316	300	16	0	0	Adapter, Device, Property, Sensor and Thing

Such requirements related to the relevant concepts Device and Thing of both ontologies are detailed in Table 9, together with the testing results for each ontology in order to analyse the overlap in these concepts. It is worth mentioning that the grey line indicates that the terms in the requirements are not considered in the ontology, that is, they are undefined. The green check mark (✓) represents that the ontology passes the test, while the orange check mark (✓) represents that the test returns the absent relation result, that is, the requirement is not implemented in the ontology and the addition of it would not lead to an inconsistent ontology. Additionally, the red cross (✗) indicates that the requirement is not implemented in the ontology and the addition of it would lead to an inconsistent ontology.

Table 9: Overlap between SAREF and VICINITY ontologies

Topic	Provenance	Requirement	SAREF ontology	VICINITY ontology network
Adapter	VICINITY	A lightbulb is a device	—	✓
Adapter	VICINITY	A smart oven is a type of device	—	✓

Adapter	VICINITY	A light switch is a type of device	✓	✓
Adapter	VICINITY	The battery storage unit is a type of device	—	✓
Adapter	VICINITY	The photovoltaic panel is a type of device	—	✓
Adapter	VICINITY	The power meter is a type of device	—	✓
Adapter	VICINITY	A battery storage unit is a type of device	—	✓
Adapter	VICINITY	A photovoltaic panel is a type of device		✓
Device	SAREF	A device can be characterized by a profile	✓	—
Device	SAREF	What is a device?	✓	✓
Device	SAREF	A device performs one or more functions	✓	—
Device	SAREF	Multiple devices can offer the same service	✓	—
Device	SAREF	A service shall specify the device that is offering the service	✓	—
Device	SAREF	A device shall have a model property	✓	✓
Device	SAREF	Examples of devices are a light switch, a temperature sensor, an energy meter, a washing machine	✓	—
Device	SAREF	A device can optionally have a description	✓	✓
Device	SAREF	A device shall have a model property	✓	✓
Device	SAREF	A device may consist of other devices	✓	✓
Device	SAREF	A device can be found in a corresponding state	✓	✓
Device	SAREF	The devices can belong to several categories	✓	—
Device	SAREF	The devices can be classified into categories: FunctionRelated, EnergyRelated and BuildingRelated	✓	—

Device	SAREF	A device offers a service	✓	—
Device	SAREF	A device can be used for measuring a property	✓	✓
Device	SAREF	A device can be used for the purpose of sensing	✓	✓
Device	SAREF	A device can be used for the purpose of offering a commodity	✓	—
Device	VICINITY	What is an IoT device?	✓	✓
Device	VICINITY	A device has a unique identifier	✓	✓
Device	VICINITY	Which attributes can have a device? Device deviceName only string, Device avatar Image, Device serialNumber only string	—	✓
Device	VICINITY	What is a device profile? Device deviceName only string Device avatar Image, Device deviceDescription only string, Device serialNumber only string	—	✓
Device	VICINITY	A device can have a status	✓	✓
Device	VICINITY	A device can have a location	—	✓
Device	VICINITY	Which are the social relationships a device can be involved in?	—	✓
Device	VICINITY	Which devices are there?	—	✓
Property	VICINITY	The temperature is a property	✓	✓
Property	VICINITY	The position of the valve is a property	—	✓
Property	VICINITY	The current mode is property	—	✓
Property	VICINITY	The door status is a property	—	✓
Property	VICINITY	The light status is a property	—	✓

Property	VICINITY	The light of a freezer is a property	—	✓
Property	VICINITY	The temperature of the refrigerator is a property	—	✓
Property	VICINITY	The status of the freezer door is a property	—	✓
Property	VICINITY	The temperature of the freezer is a property	—	✓
Property	VICINITY	The light is a type of property	✓	✓
Property	VICINITY	The baking temperature is a type of property	—	✓
Property	VICINITY	The baking time is a type of property	—	✓
Property	VICINITY	The alarm time is a type of property	—	✓
Property	VICINITY	The bake elapsed time is a type of property	—	✓
Property	VICINITY	The bake remaining time is a type of property	—	✓
Property	VICINITY	The baking start time hour is a type of property	—	✓
Property	VICINITY	The baking start time minute is a type of property	—	✓
Property	VICINITY	CO2 is a type of property	—	✓
Property	VICINITY	Luminance is a type of property	—	✓
Property	VICINITY	The presence is a type of property	—	✓
Property	VICINITY	Power consumption is a type of property	—	✓

Property	VICINITY	The device status is a type of property	—	✓
Property	VICINITY	An operational status is a type of property	—	✓
Property	VICINITY	Sound level is a type of property	—	✓
Property	VICINITY	The weight is a type of property	—	✓
Property	VICINITY	The systolic blood pressure is a type of property	—	✓
Property	VICINITY	The diastolic blood pressure is a type of property	—	✓
Property	VICINITY	The heart rate is a type of property	—	✓
Property	VICINITY	The activity tracker is a type of property	—	✓
Property	VICINITY	The steps are a type of property	—	✓
Property	VICINITY	The distance walked is a type of property	—	✓
Property	VICINITY	The calories burned are a type of property	—	✓
Property	VICINITY	The sleep is a type of property	—	✓
Property	VICINITY	The battery is a type of property	—	✓
Property	VICINITY	The panic button is a type of property	—	✓
Property	VICINITY	Nominal electric vehicle current is a type of property	—	✓
Property	VICINITY	An energy supply system is a type of property	—	✓

Property	VICINITY	The number of connectors is a type of property	—	✓
Property	VICINITY	The maximal power per connector is a type of property	—	✓
Property	VICINITY	The simultaneous charging is a type of property	—	✓
Sensor	VICINITY	A humidity sensor is a type of sensor	—	✓
Sensor	VICINITY	A motion sensor is a type of sensor	—	✓
Sensor	VICINITY	A door sensor is a type of sensor	—	✓
Sensor	VICINITY	A window sensor is a type of sensor	—	✓
Sensor	VICINITY	A window sensor observes whether a window is opened	—	✓
Sensor	VICINITY	A thermostat is a type of sensor and actuator	—	✓
Sensor	VICINITY	A CO2 sensor is a type of sensor	—	✓
Sensor	VICINITY	The luminance sensor is a type of sensor	—	✓
Sensor	VICINITY	A noise sensor is a type of sensor	—	✓
Sensor	VICINITY	An indoor climate quality sensor is a type of sensor	—	✓
Sensor	VICINITY	A people counter is a type of sensor	—	✓
Sensor	VICINITY	The motion sensor is a type of device	—	✓
Sensor	VICINITY	A sound sensor is a type of sensor	—	✓

Sensor	VICINITY	An HVAC sensor is a type of sensor	—	✓
Sensor	VICINITY	The weight scale is type of sensor	—	✓
Sensor	VICINITY	The blood pressure monitor is a type of sensor	—	✓
Sensor	VICINITY	A thermometer is a type of sensor	✓	✓
Thing	VICINITY	What is a thing in the web thing context?	✓	✓
Thing	VICINITY	Each thing has at least an interaction pattern	—	✓
Thing	VICINITY	Security is associated with things	—	✓

From Table 9, it is possible to extract some information regarding the overlap between the ontologies. Firstly, both ontologies include the concepts Device and Thing and have the following similarities:

- Devices in VICINITY and SAREF have a model.
- Devices in VICINITY and SAREF have a status.
- Devices in VICINITY and SAREF measure properties.

However, there are some definitions of such concepts that are not included in both ontologies:

- The relation between a device and a service, which is present in SAREF but not in VICINITY.
- The relation between a device and a commodity, which is present in SAREF but not in VICINITY.
- The category of devices, which is present in SAREF but not in VICINITY.
- The relation between a device and its functions, which is present in SAREF but not in VICINITY.
- The relation between a device and its owner, which is present in VICINITY but not in SAREF.
- The relation between a thing and the interaction patterns, which is present in VICINITY but not in SAREF.
- The relation between a thing and a thing and the security, which is present in VICINITY but not in SAREF.
- The relation between a device and a spatial thing, which is present in VICINITY but not in SAREF.

Moreover, from this table it is also possible to see that the conceptualization of the Profile concept defined in the SAREF ontology is different from what it is conceptualized as a Profile in the VICINITY ontologies, even though there is no logical incompatibility. Figure 19 and Figure 20 show the conceptualization of each ontology regarding the requirements. In these figures, green rectangles and grey arrows represent the concepts and relationships that are implemented in both ontologies.

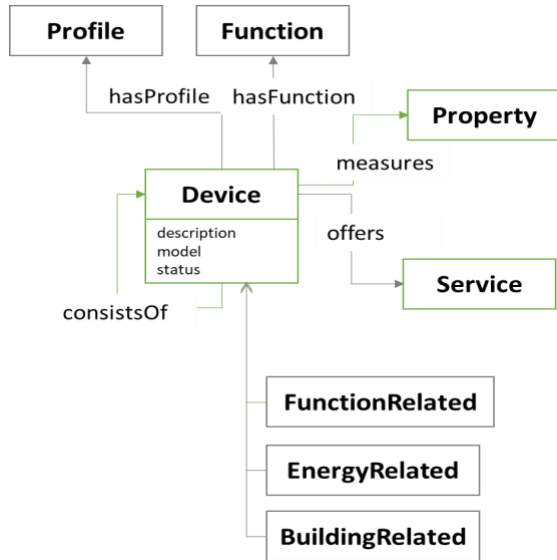


Figure 20: SAREF Requirements conceptualization of Device and Thing

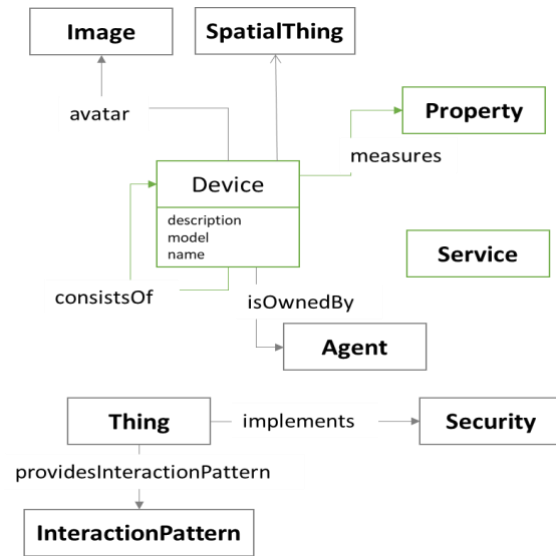


Figure 19: VICINITY Requirements conceptualization of Device and Thing

Table 10 exposes that the SSN ontology only satisfies one requirement of the VICINITY ontology network and that this requirement is related to the Thing concept.

Table 10: Testing results for W3C SSN ontology regarding VICINITY requirements

Analysed standard	Tests results					Relevant concepts
	Total number of tests	Number of test with undefined terms result	Number of test with passed result	Number of test with conflict result	Number of test with absent relation result	
WoT	14	13	1	0	0	Thing
Core	50	50	0	0	0	-
WoT Mapping s	15	15	0	0	0	-
WoT adapters	154	154	0	0	0	-
WoT datatypes	13	13	0	0	0	-
	246	245	1	0	0	

As it was previously done with the SAREF ontology, to identify in more detail what is the overlap between the analysed ontologies and in which concepts there is a relation between W3C SSN and the VICINITY ontology network, the information summarized in Table 6 is also included in the analysis. The tests of both ontologies related to the topics where there is a potential overlap, i.e., those relevant concepts, between both the VICINITY and the W3C SSN ontologies are analysed separately. It is worth mentioning that, as shown in Table 6 and Table 10, the VICINITY ontologies satisfy several requirements of the SSN ontology, while the SSN only satisfies one requirement of the VICINITY ontologies. This occurs because the VICINITY ontology imports the SOSA ontology,¹⁷ which is a module of the W3C SSN ontology. Table 11 summarizes the number of tests of both ontologies, together with their results and the identified relevant concepts, joining the results obtained in Table 6 and Table 10.

Table 11: Testing results for SSN and VICINITY ontologies

Analysed standard	Tests results					Relevant concepts
	Total number of tests	Number of test with undefined terms result	Number of test with passed result	Number of test with conflict result	Number of test with absent relation result	
SSN and VICINITY	280	255	10	0	15	Sensor, System, Procedure, Property, Feature of Interest, Actuator, Deployment and Thing

The requirements related the relevant concepts where there might be a potential overlap are detailed in Table 12, together with the testing results.

¹⁷ <https://www.w3.org/ns/sosa>

Table 12: Overlap between the SSN and VICINITY ontologies

Topic	Provenance	Requirement	W3C SSN ontology	VICINITY ontology network
Actuation	SSN	An actuation has a result	✓	✓
Actuator	SSN	An actuator made only actuations	✓	✓
Actuator	SSN	What is an actuator?	✓	✓
Deployment	SSN	What is a deployment?	✓	✓
Deployment	SSN	Deployment is associated to a property	✓	—
Feature of Interest	SSN	What is feature of interest?	✓	✓
Platform	SSN	Platform has a deployment	✓	—
Platform	SSN	A platform is deployed by some system	✓	—
Platform	SSN	A deployment is deployed in a platform	✓	—
Procedure	SSN	A procedure has an output	✓	—
Procedure	SSN	A procedure is implemented by a system	✓	—
Procedure	SSN	A procedure has an input	✓	—
Procedure	SSN	What is a procedure?	✓	✓
Property	SSN	What is a property?	✓	✓
Sensor	SSN	What is a sensor?	✓	✓
Sensor	SSN	A sensor is a type of system	✓	✓

Sensor	SSN	Sensor detects a stimulus		
Sensor	VICINITY	A humidity sensor is a type of sensor		
Sensor	VICINITY	A motion sensor is a type of sensor		
Sensor	VICINITY	A door sensor is a type of sensor		
Sensor	VICINITY	A window sensor is a type of sensor		
Sensor	VICINITY	A window sensor observes whether a window is opened		
Sensor	VICINITY	A thermostat is a type of sensor and actuator		
Sensor	VICINITY	A CO2 sensor is a type of sensor		
Sensor	VICINITY	The luminance sensor is a type of sensor		
Sensor	VICINITY	A noise sensor is a type of sensor		
Sensor	VICINITY	An indoor climate quality sensor is a type of sensor		
Sensor	VICINITY	A people counter is a type of sensor		
Sensor	VICINITY	The motion sensor is a type of device		
Sensor	VICINITY	A sound sensor is a type of sensor		
Sensor	VICINITY	An HVAC sensor is a type of sensor		
Sensor	VICINITY	The weight scale is type of sensor		
Sensor	VICINITY	The blood pressure monitor is a type of sensor		

Sensor	VICINITY	A thermometer is a type of sensor	—	✓
System	SSN	A system implements a procedure	✓	✓
System	SSN	A system can have subsystems	✓	✓
System	SSN	What is a system?	✓	✓
VICINITY	Thing	What is a thing in the web thing context?	✓	✓
VICINITY	Thing	Each thing has at least an interaction pattern	—	✓
VICINITY	Thing	Security is associated with things	—	✓

From Table 12, it is possible to extract some information regarding the overlap between the SSN ontology and the VICINITY ontology network. Firstly, both ontologies include the concepts Thing, Procedure, Sensor, System, Procedure, Property, Feature of Interest, Actuator and Deployment and have the following similarities:

- The relation between a property and a feature of interest.
- The relation between a sensor and a system.

However, there are some definitions of such concepts that are not included in both ontologies:

- The relation of a procedure and its output, which is in the SSN ontology but not in the VICINITY ontologies.
- The relation between a platform and its deployment, which is in the SSN ontology but not in the VICINITY ontologies.
- The relation between a sensor and its stimulus, which is in the SSN ontology but not in the VICINITY ontologies.
- The relation between an actuator and its actuations, which is in the SSN ontology but not in the VICINITY ontologies.
- The relation between a procedure and its inputs, which is in the SSN ontology but not in the VICINITY ontologies.
- The relation between a deployment and its platform, which is in the SSN ontology but not in the VICINITY ontologies.
- The relation between a system and its procedure, which is in the SSN ontology but not in the VICINITY ontologies.
- The relation between a systems and other systems, which is in the SSN ontology but not in the VICINITY ontologies.
- The relation between a deployment and a property, which is in the SSN ontology but not in the VICINITY ontologies.

- The relation between an actuation and a result, which is in the SSN ontology but not in the VICINITY ontologies.
- The relation between a sensor and its observable properties, which is in the SSN ontology but not in the VICINITY ontologies.
- The relation between security and things, which is in the VICINITY ontologies but not in the SSN ontology.
- The relation between things and its interaction pattern, which is in the VICINITY ontologies but not in the SSN ontology.
- The hierarchy of sensors, which is in the VICINITY ontologies but not in the SSN ontology.

Even though these concepts, i.e., Thing, Procedure, Sensor, System, Procedure, Property, Feature of Interest, Actuator and Deployment are not completely aligned in VICINITY and SSN ontologies, it can be concluded that no incompatibilities were found. Additionally, it was also found that there are some absent relations in the ontology. This can be deduce from Table 12, where the two first requirements have an orange mark. This mark means that even though the terms Actuator and Result exist in the ontology, the relation between such terms are not defined.

Figure 22 and Figure 21 show the conceptualization of each ontology regarding the requirements, where green rectangles and arrows represent the concepts and relationships implemented in both ontologies.

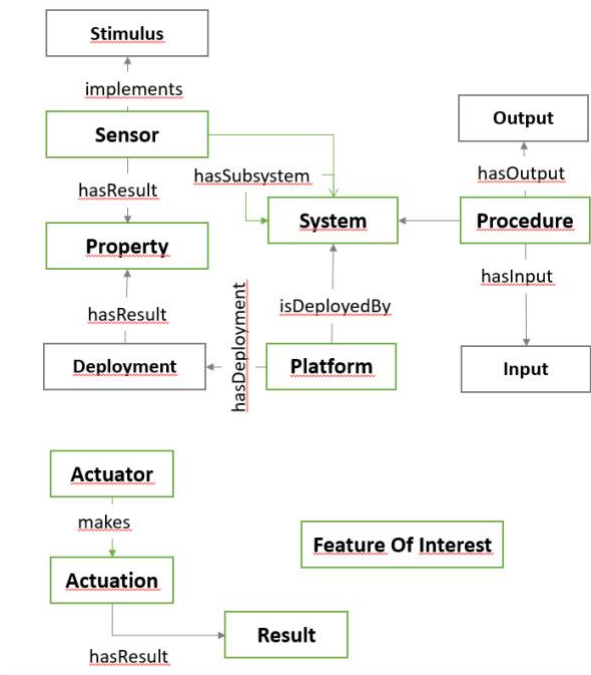


Figure 21: SSN Requirements conceptualization of Sensor and Thing

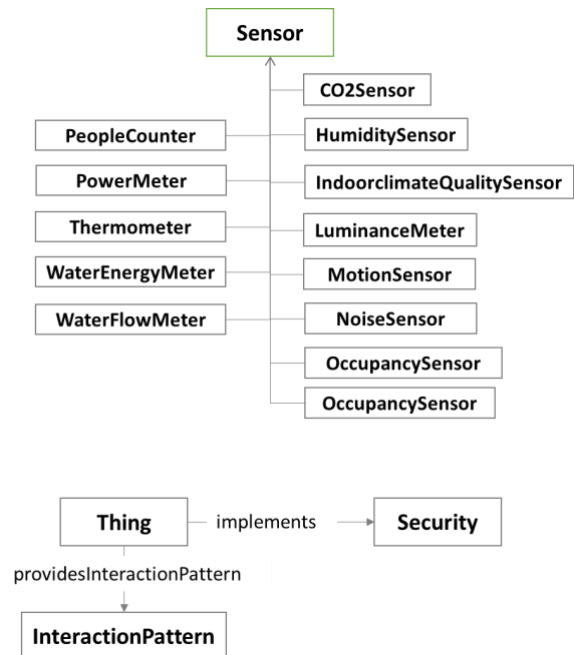


Figure 22: VICINITY Requirements conceptualization of Sensor and Thing

Finally, Table 13 exposes that the oneM2M ontology only satisfies 1 requirement of the VICINITY ontology network, and that this requirement is related to the Thing concept.

Table 13: Testing results for the oneM2M ontology regarding to VICINITY requirements

Analysed standard	Tests results					Relevant concepts
	Total number of tests	Number of test with undefined terms result	Number of test with passed result	Number of test with conflict result	Number of test with absent relation result	
WoT	14	13	1	0	0	WoT
Core	50	50	0	0	0	-
WoT Mapping s	15	15	0	0	0	-
WoT adapters	154	154	0	0	0	-
WoT datatypes	13	13	0	0	0	-
	246	245	1	0	0	

As it was previously done with the SAREF and the SSN ontologies, to identify in more detail what is the overlap and in which concepts there is a relation between oneM2M and the VICINITY ontology network, the tests related to Device and Thing topics of both VICINITY and oneM2M are analysed separately. Table 14 summarizes the number of tests of both ontologies, together with their results and the identified relevant concepts, joining the results obtained in Table 6 and Table 13.

Table 14: Testing results for oneM2M and VICINITY ontologies

Analysed standard	Tests results					Relevant concepts
	Total number of tests	Number of test with undefined terms result	Number of test with passed result	Number of test with conflict result	Number of test with absent relation result	
oneM2M and VICINITY	270	264	5	0	1	Device and Thing

The requirements related the relevant concepts identified are added to Table 15, together with the testing results for both ontologies.

Table 15 Overlap between the oneM2M and VICINITY ontologies

Topic	Provenance	Requirement	oneM2M ontology	VICINITY ontology network
Device	OneM2M	A device performs one or more functionalities in order to accomplish a particular task	✓	—
Device	OneM2M	A device may be a physical or non-physical entity	✓	✓
Device	OneM2M	A device can be composed of several (sub-)devices	✓	✓
Device	OneM2M	A device has one or more services that expose in the network its functionalities	✓	—
Device	VICINITY	What is an IoT device?	✓	✓
Device	VICINITY	A device has a unique identifier	—	✓
Device	VICINITY	Which attributes can have a device? Device deviceName only string, Device avatar Image, Device serialNumber only string	—	✓
Device	VICINITY	What is a device profile? Device deviceName only string Device avatar Image, Device deviceDescription only string, Device serialNumber only string	—	✓
Device	VICINITY	A device can have a status	—	✓
Device	VICINITY	A device can have a location	—	✓
Device	VICINITY	Which are the social relationships a device can be involved in?	—	✓
Device	VICINITY	Which devices are there?	—	✓
Device	VICINITY	A device profile indicates the type of device, e.g: sensor or actuator	—	✓
Thing	OneM2M	A thing is an entity that can be identified in the oneM2M System	✓	✓

Thing	OneM2M	A thing can have relations to other things	✓	✓
Thing	OneM2M	A thing may have properties	✓	—
Thing	VICINITY	What is a thing in the web thing context?	✓	✓
Thing	VICINITY	Each thing has at least an interaction pattern	—	✓
Thing	VICINITY	Security is associated with things	—	✓

From Table 15 it was shown that both ontologies include the concepts Device and Thing and have the following similarities:

- A relation between a device and other devices.
- The interaction between things.

However, there are some definitions of such concepts that are not included in both ontologies:

- The relation between a device and a task, which is in the OneM2M ontology but not in the VICINITY ontology.
- The relation between a device and a service, which is in the OneM2M ontology but not in the VICINIY ontology.
- The relation between a thing and its properties, which is in the OneM2M ontology but not in the VICINIY ontology.
- The attributes of a device, which is in the VICINITY ontologies but not in the OneM2M ontology.
- The relation between a thing and its interaction patterns, which is in the VICINITY ontologies but not in the OneM2M ontology.
- The relation between security and things, which is in the VICINITY ontologies but not in the OneM2M ontology.
- The definition of a device profile, which is in the VICINITY ontologies but not in the OneM2M ontology.

Moreover, from this table it is also possible to see that the conceptualization of the Device concept defined in the OneM2M ontology is different from what it is conceptualized as a Device in the VICINITY ontologies, since in OneM2M a device can be a physical or non-physical entity while in VICINITY a device can only be a physical entity. Figure 23 and Figure 24 show the conceptualization of both ontologies regarding the defined requirements, where green rectangles and grey arrows represent the concepts and relationships that are implemented in both ontologies. It is worth noting that, even though the conceptualization of Device is different in both ontologies, there is no logical incompatibility between them.

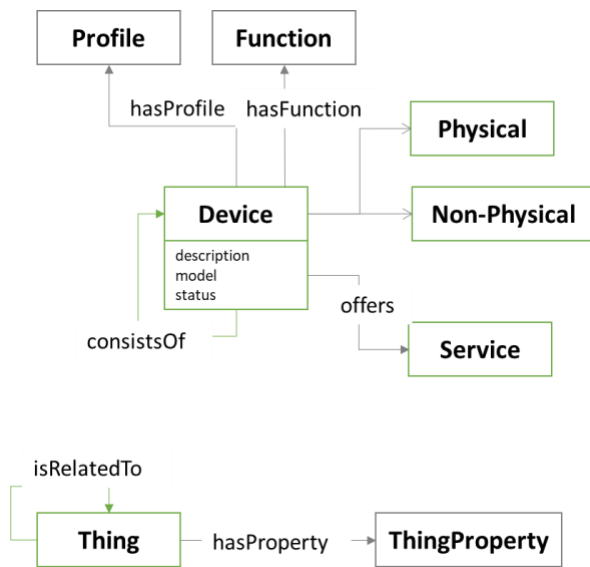


Figure 23: oneM2M Requirements conceptualization of Device and Thing

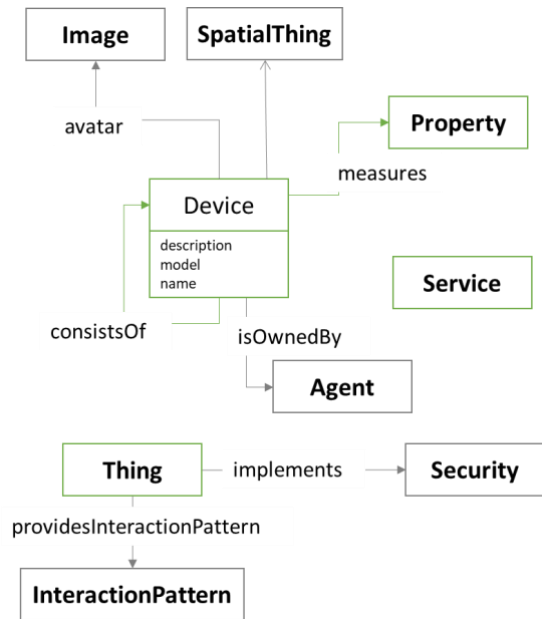


Figure 24: VICINITY Requirements conceptualization of Device and Thing

Finally, Table 16 and

Table 17 summarize the results obtained from the coverage analysis. Table 16 shows the number of requirements defined for the VICINITY ontology network that are satisfied for each of the analysed IoT standard. Additionally,

Table 17 exposes the number of requirements defined for each of the IoT standards that are satisfied for the VICINITY ontology network. From these tables it can be observed that VICINITY is partially align with the IoT standards, e.g., 14 requirements of ISO/IEC 30131:2017 or 9 requirements of SSN ontology were satisfied. However, the VICINITY ontology is out of scope of such standards. This fact can be observed in

Table 17 where only the SAREF ontology satisfies more than one ontology. Such results could be justified due to the fact that the VICINITY ontology network was created for a particular project and domain with requirements for particular partners, while the IoT standards are more generic. However, it will be analysed if the VICINITY ontology network could be completed with more concepts that are defined in the IoT standards, in order to increase the conformance between them.

Table 16: Overview of the number of VICINITY requirements satisfied by each IoT standard

Standard	Tests results	
	Number of VICINITY requirements passed	Percentage of VICINITY requirements passed
ETSI SAREF ¹⁸	12	4.87%
W3C SSN ¹⁹	1	0.4%
OneM2M ²⁰	1	0.4%

Table 17: Overview of the number of the IoT standards' requirements satisfied by the VICINITY ontology network

Ontology	Standard test cases				
	Percentage of SAREF requirements passed	Percentage of SSN requirements passed	Percentage of OCF requirements passed	Percentage of OneM2M requirements passed	Percentage of ISO/IEC 30141:2017 requirements passed
VICINITY ontology network	5.71% (4 reqs. passed)	26.47% (9 reqs. passed)	22.22% (6 reqs passed)	12.12% (4 reqs. passed)	38.89% (14 reqs. passed)

¹⁸ See <http://vicinity.iot.linkeddata.es/vicinity/testing/report-saref-results.html> for the online version of the SAREF ontology requirements.

¹⁹ See <http://vicinity.iot.linkeddata.es/vicinity/testing/report-ssn-results.html> for the online version of the SSN ontology requirements.

²⁰ See <http://vicinity.iot.linkeddata.es/vicinity/testing/report-onem2m-results.html> for the online version of the One2M2M ontology requirements.

6. Conclusions

Along this document the methodology and infrastructure of the evaluation of the VICINITY ontology has been detailed. Such evaluation includes several criteria: (1) validation regarding the model, (2) verification regarding their ontological requirements, (3) validation regarding pilot data, and (4) verification regarding IoT standards.

From the evaluation presented in this document it can be concluded that the VICINITY network does not have inconsistencies or modelling errors, and that it covers all the requirements given by the partners. The testing process allows the ontology developers to check that after each iteration whether all the requirements were satisfied and that it covers all the partners' expectations. Additionally, we have performed an ontology-driven data validation of the VICINITY platform relying on our Things Monitor component. As a result, the quality of data according to a set of requirements can be guaranteed.

Apart from checking that the developed ontologies satisfy all the project requirements, we also analyse how the VICINITY network is aligned with well-known IoT standards. From this coverage analysis we could conclude that, even though the VICINITY ontology network does not cover all the requirements in those standards, there are no inconsistencies between the standards and the VICINITY ontologies. This document also identifies the terms shared by the standards and the VICINITY ontologies. The coverage analysis also shows that VICINITY ontology network has partial conformance with the IoT standards, e.g., 14 requirements of ISO/IEC 30131:2017 or 8 requirements of SSN ontology were satisfied. However, this coverage analysis also shows that the VICINITY ontology network is out of scope of the analysed IoT standards, which is expected because the IoT standards are more generic than the VICINITY ontology network.

In addition, we analysed the elements from the ontology used by the pilots to describe their infrastructures. As a result, we concluded that most of the data in vicinity is properly labelled with specific types from the Adapters ontology; pilots are mostly registering properties, but there are also some actions and events. The properties are correctly link to a physical magnitude observed by such property however units are not been used. In addition another lack identified is the use of contextual data such as buildings, rooms, cities, or countries. Finally, the number of Things registered with mappings, and thus interoperable, is not very high. Therefore, although everything is suitable to be discovered, not all discoverable infrastructures will allow the semantic interoperability services to be accessed. This matter with the contextual data and the mappings is not necessarily something wrong, since depends on the level of privacy that the pilots want to specify in their Things.

As future work, the minor pitfalls detected from the execution of OOPS! will be corrected, in order to improve the readability of the ontology. Additionally, a further analysis regarding whether more concepts should be reused or imported from the IoT standards will also be accomplished, in order to improve the conformance between the VICINITY ontology network and the analysed standards. Finally, as the VICINITY ontology network development methodology is iterative, the tests and their results will be updated in order to support the verification for such future versions.

In summary, this deliverable presents the evaluation of the VICINITY ontology network. It is worth mentioning that during the project lifetime, new requirements could appear and the VICINITY ontology

could be updated. All the results presented in this document related to the validation regarding the model and the verification regarding their ontological requirements are available online in the VICINITY ontology portal²¹ and will be updated over time if some of the ontologies change.

²¹ <http://vicinity.iot.linkeddata.es/test>

7. References

- [1] M. C. Suárez-Figueroa y A. Gómez-Pérez, «NeOn Methodology for Building Ontology Networks: a Scenario-based Methodology,» de *Proceedings of the International Conference on SOFTWARE, SERVICES & SEMANTIC TECHNOLOGIES*, Sofia, Bulgaria, 2009.
- [2] S. Abburu, «A Survey on Ontology Reasoners and Comparison,» *International Journal of Computer Applications*, 2012.
- [3] M. Poveda-Villalón, A. Gómez-Pérez y M. C. Suárez-Figueroa, «OOPS! (OntOlogy Pitfall Scanner!): An On-line Tool for Ontology Evaluation,» *International Journal on Semantic Web and Information System*, 2014.
- [4] A. Fernández-Izquierdo y R. García-Castro, «Requirements Behaviour Analysis for Ontology Testing,» de *European Knowledge Acquisition Workshop*, Nancy, 2018.
- [5] A. Fernández-Izquierdo, M. Poveda-Villón y R. García-Castro, «Fernández-Izquierdo, A., Poveda-Villalón, M., & García-Castro, R. (2019, June). CORAL: A Corpus of Ontological Requirements Annotated with Lexico-Syntactic Patterns,» de *European Semantic Web Conference*, 2019.
- [6] M. C. Suarez-Figueroa, S. Brockmans, A. Gangemi, A. Gomez-Perez, J. Lehmann, H. Lewen, V. Presutti y M. Sabou, «NeOn Modelling Components. Deliverable 5.1.1 NeOn Project,» 2007.
- [7] ETSI, «SAREF extension investigation Technical Report (TR 103 411),» 2016.
- [8] ISO, «ISO/IEC 30141:2017:Internet of Things (IoT) - Reference Architectures,» 2017.
- [9] O. C. Foundation, «OCF SPECIFICATION 2.0.2,» [En línea]. Available: <https://openconnectivity.org/developer/specifications> .