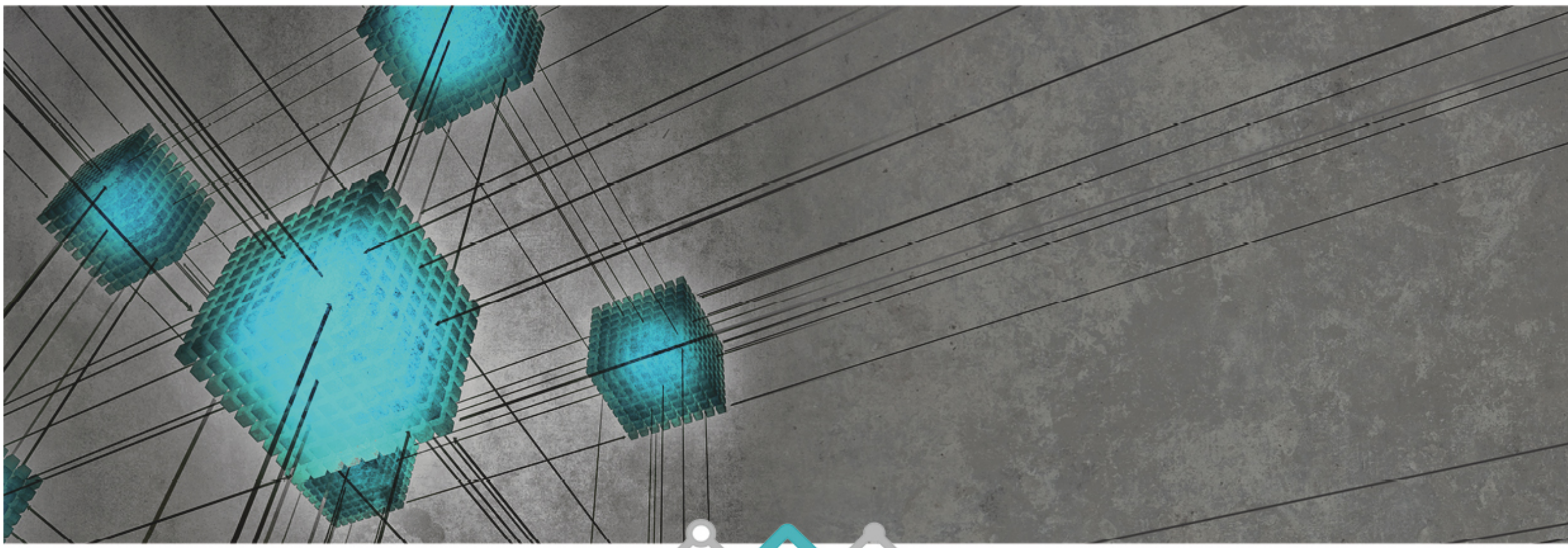

interiot

INTEROPERABILITY
OF HETEREOGENEUS
IOT PLATFORMS.

D4.2

Final Reference IoT Platform Meta-architecture and Meta-data Model

January 2018

INTER-IoT

INTER-IoT aim is to design, implement and test a framework that will allow interoperability among different Internet of Things (IoT) platforms.
Most current existing IoT developments are based on ”closed-loop” concepts, focusing on a specific purpose and being isolated from the rest of the world. Integration between heterogeneous elements is usually done at device or network level, and is just limited to data gathering. Our belief is that a multi-layered approach integrating different IoT devices, networks, platforms, services and applications will allow a global continuum of data, infrastructures and services that can will enable different IoT scenarios. As well, reuse and integration of existing and future IoT systems will be facilitated, creating a de-facto global ecosystem of interoperable IoT platforms.
In the absence of global IoT standards, the INTER-IoT results will allow any company to design and develop new IoT devices or services, leveraging on the existing ecosystem, and bring get them to market quickly.
INTER-IoT has been financed by the Horizon 2020 initiative of the European Commission, contract 687283.

INTER-IoT

Final Reference IoT Platform Meta-architecture and Meta-data Model

Version: Final

Security: Public

January 31, 2018

The INTER-IoT project has been financed by the Horizon 2020 initiative of the European Commission, contract 687283

Disclaimer

Executive Summary

The Deliverable D4.2 aims to provide the final version of the Reference Model, Meta-data and Reference Architecture for the INTER-IoT project. Therefore, it extends and updates the work previously done in D4.1.

One of the main issues of systems belonging to the domain of Internet of Things is *interoperability*. As the concept itself of IoT is very large, and spans over totally different technological sets and application domains, it is almost impossible to have a single vertical solution, even for a given domain. When development of IoT systems started, there was an emergence of the so-called INTRAnet of Things: providing a specific service was the key objective, while being interoperable was not even a requirement in the "could-do" list.

As there are currently a very large number of different platforms, and the situation is likely to stay the same for the foreseeable future, it's very important to develop systems that are able to make different platforms talk to each other seamlessly. As many times the differences between those platforms start from the modeling phase, and how they see the interaction between the real world and the digital one, it's necessary that our work will begin from the modeling of the interaction between different IoT platforms.

During the first stages of this project, one of the most relevant objectives that has been achieved is the definition of an initial reference meta-architecture and a reference meta-data model. This helped us to create the guidelines and the backbone for defining an interoperability system to be applied in a myriad of use cases. Due to its abstraction, these reference models can be used as a base to build interoperability solutions in a technologically agnostic manner. Hence, the solutions and products developed within the project have been designed following the baselines of these reference models. Additionally, the methodology and the pilots have utilized these reference models to feed its structures and technological choices. As the reference models have been widely defined in D4.1, in this deliverable we present a brief comment on how they have influenced or interacted with the other WPs of the project.

The INTER-IoT Reference Model (RM) has been initially defined in the Deliverable D4.1. It is based upon the results of the IoT-A project; however, unlike the RM proposed in IoT-A, in INTER-IoT the RM is fully focused on providing a model for interoperability of existing IoT Platforms. Although IoT-A also contains the concept of interoperability, this is considered a design constrain for the creation of new platforms. In spite, the approach followed in INTER-IoT is the provision of interoperability mechanisms for already existing platforms, which is more realistic for real scenarios in industry, health, smart cities, etc. This deliverable contains an update of the reference model, including the analysis of the existing platforms involved in INTER-IoT pilots (INTER-LogP and INTER-Health) and some improvements in the definition of terms in the different subsections. In particular, the domain model has been updated including a reformulation of some terms, bringing a more uniform definition of concepts such as services or resources. The Functional Model has been simplified, to make it more understandable and easy to use in practice.

The INTER-IoT Reference meta-data model is proposed in the form of an OWL ontology. The INTER-IoT ontology named GOIoTP (Generic Ontology for IoT Platforms) was engineered to fulfill the requirements put forth by INTER-IoT, but also to offer a core ontology for any artifact in the IoT domain. While reusing existing standardized and established ontologies used in IoT, GOIoTP also additionally proposes its own extensions, in order to provide a complete, modular ontology, useful in a wide range of IoT use cases. Because GOIoTP is a core ontology, it defines some stub classes – i.e. classes that do not have a robust definition, or subclasses. This is a design decision intended to enable different extensions, for those stub concepts, depending on particular

needs of an implementation, and follows the usual design patterns for top-level ontologies. Other than offering less restrictions on implementers, stub classes are also easier to align to and from other core IoT ontologies. In order to offer a more complete interoperability solution, we have also defined an extension to GOIoTP that expands the definitions and subclass structure for the stub classes. GOIoTPex (GOIoTP extended) is the multi-module ontology that extends the reference model of GOIoTP with concrete entities. It completes GOIoTP and is suitable for scenarios, in which INTER-IoT user does not have any preference, when it comes to the data model used internally by INTER-IoT components, i.e. when the user is looking for a complete modeling solution.

The Reference Architecture of INTER-IoT is derived from the IoT-A Reference Architecture and is a refinement of the previous work in Deliverable D4.1. The main target has been to simplify and to make it more usable by everyone. While the concepts remain largely the same, the way that they are laid out should help final users to get acquainted sooner with the different concepts and to use the RA as a tool for their daily design. As a reminder, a Reference Architecture is a blueprint for developing Concrete Architectures. Its main scope is to apply different views to functionality areas, in order to identify the relationships between different modules and the need of different functions.

Finally, we will show how to apply the different modeling and architectural work in a real environment. For that, real scenarios for the two application domains are described, in which it is analysed how the reference architecture can be applied.

List of Authors

Organisation	Authors	Main contributions
UPVLC	Eneko Olivares, Jara Suárez de Puga, Carlos Enrique Palau Salvador	Communication Model, complementarity with other WPs
UNICAL	Raffaele Gravina	Complementarity with other WPs
PRODEVELOP	Miguel Ángel Llorente Carmona	Coordination of Reference Model, Domain Model,
TU/e	Tim Van der Lee	Overall contributions
ASL TO5	Anna Costa	Guidelines
AFT	Moncef Semichi	Guidelines
RINICOM	Eric Carlson	Reference Model discussions
XLAB	Matevž Markovič, Flavio Fuart, Manja Gorenc Novak	Overall contributions
SRIPAS	Paweł Szmeja Wiesław Pawłowski, Katarzyna Wasielewska	Metadata Model, Internal Review
ABC	Alessandro Bassi	Overall coordination, Introduction, Functional Model, Reference Architecture, Appendixes
VPF	Pablo Giménez Salazar	Guidelines
SABIEN	Gema Ibáñez, Vicente Traver, Alvaro Fides	Ethics section. Internal Review

Change control datasheet

Version	Changes	Pages
0.0	Formatting, Inter-IoT template	
0.1	Table of content draft and basic assignments	5
0.2	Reference Model chapter defined	8
0.3	Metadata model added	30
0.4	Guidelines section drafted	35
0.5	Position of this Document within WPs	42
0.6	Reference Model "almost" complete	68
0.7	Reference Architecture Added	78
0.8	Completed Section 2, 3, 6	89
0.9	Ready for internal review	99
0.95	Reviewed by internal reviewers	99
1.0	Addressed review comments	99

Contents

List of Figures

Acronyms

PC	Project Coordinator
D#.#	Deliverable number #.# (D2.1 deliverable 1 of work package 2)
DoA	Description of Action of the project
INTER-IoT	Interoperability of Heterogeneous IoT Platforms
EC	European Commission
EU	European Union
GA	Grant Agreement
H2020	Horizon 2020 Programme for Research and Innovation
IoT	Internet of Things
IPR	Intellectual Property Rights
M#	#th month of the project (M1=January 2016)
WP	Work Package
IPR	Intellectual Property Rights
PCC	Project Coordination Committee
PIC	Project Implementation Committee
STPM	Scientific and Technical Project Manager
TL	Task Leader
WPL	Workpackage Leader

1 Introduction

One of the main issues of systems belonging to the domain of Internet of Things is *interoperability*. As the concept itself of IoT is very large, and spans over totally different technological sets and application domains, it is almost impossible to have a single vertical solution, even for a given domain. When development of IoT systems started, we saw the emergence of the so-called **INTRAnet** of Things: providing a specific service was the key objective, while being interoperable was not even a requirement in the "could-do" list. Looking at the numbers, it's undeniable that every single provider, vendor and even a large number of small players developed a platform for IoT. We are talking about 400 different platforms here, and the number is still growing. We are therefore in a situation where, instead of converging on a limited number of standard solutions, players are trying to capture the market, establishing their solution as superior to become a *de facto* standard for the IoT domain, or at least some of its subdomains.
As for the time being there is no winner, and the situation is likely to stay the same for the foreseeable future, it's very important to develop systems that are able to make different platforms talk to each other seamlessly. As many times the differences between those platforms start from the modeling phase, and how they see the interaction between the real world and the digital one, it's necessary that our work will begin from the modeling of the interaction between different IoT platforms.
This document is focused on the development of a modeling and architectural abstraction for the interoperability of different IoT systems, and serves as base for development of further modules within the INTER-IoT project.
As a general reminder, a Reference Model as per OASIS definition is

> *an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details. A reference model may be used as a basis for education and explaining standards to non-specialists.*

For what concerns the Reference Architecture definition, we use the one provided by the IoT-A project, which says that

> *A Reference Architecture is an architectural design pattern that indicates how an abstract set of relationships realizes a set of requirements. The main purpose of a RA is to provide guidance for the development of concrete architectures. More reference architectures may be derived from a common reference model*

For what concerns a Meta-data Model, the definition we use is the following one:

Meta-data modeling is the analysis, construction and development of the frames, rules, constraints, models and theories applicable and useful for the data modeling in a predefined class of problems.

The remainder of the document is organized as follows. Section 2 focuses on the position of this Deliverable, and the development of an Architectural Reference Model (ARM) for interoperability between IoT systems, within the INTER-IoT development. Section 3 develops the same section in the Deliverable D4.1, explaining the final INTER-IoT Reference Model. Section 4 focuses on the Meta-data model, while Section 5 explains the final Reference Architecture. Section 6 illustrates in practice how to use the INTER-IoT ARM. Finally in the Annex, we report the parts of Deliverable D4.1 that didn't change, together with an explanation of the IoT-A ARM as the main inspiration of this work. INTER-Layer

2 Relationships with the other WorkPackages

During the first stages of this project, one of the most relevant objectives that has been achieved is the definition of a reference meta-architecture and a reference meta-data model to create the guidelines and the backbone to define an interoperability system to be applied in a myriad of use cases. Due to its abstraction, these reference models can be used as a base to build interoperability solutions in a technologically agnostic manner. Hence, the solutions and products developed in this project have been created following the baselines of these reference models. Additionally, the methodology and the pilots have utilized these reference models to feed its structures and technological choices. Also, the pilots have been the proof of concept to create a feedback on the references models, being this feedback a way to improve them in order to contemplate more numbers of uses cases. As the reference model have been widely defined in D4.1 and in other sections of this deliverable, here, we present a brief comment on how these reference models have influenced or interacted with the other WPs on this project.

2.1 Relationship of the RA with WPs

This section aims to reflect the impact of the reference architecture (RA) and the reference meta-model (RMM) in the development carried out in the other WPs. We identify WP3 and WP5 to be directly impacted by the RA and RMM defined in WP4.

2.1.1 Relationship of RA with WP3: INTER-Layer

In the previous version of this deliverable (D4.1) an Initial Reference IoT Platform Meta-Architecture was presented and the different INTER-IoT solutions, provided by WP3, were defined following the Reference Model (RM) or 'meta-model' for IoT Platforms Interoperability and the Reference Architecture (RA), provided by WP4.
So, as said, the already defined RA has been used for the design of the INTER-Layer architecture, and both, the INTER-IoT RA and the INTER-Layer architecture (specifically the API design of the different interoperability layers) has been the foundation for the design of the INTER-FW architecture. The INTER-IoT Reference Architecture design has been inspired in the works performed in the Lighthouse project IoT-A (project ID 257521). As a first step, the creation of the different Functional Groups (FG) and Functional Components (FC) identified as relevant for our interoperability solutions was carried out. A list of these FG and its FC identified for the development carried out in WP3 is the following:

- Application or Management FG

  - Configuration
  - Fault
  - Reporting
  - Member
  - State

- Security FG
  - Authorization
  - Authentication

- Communication FG
  - Hop-to-Hop Communication
  - Network Interoperability
  - End-to-End Interoperability

- IoT Service FG
  - IoT Service
  - IoT Service Resolution
  - VE Management
  - Node Interoperability

- Service Interoperability FG
  - Service Orchestration
  - Service Composition
  - Service Resolution

- Semantics FG
  - Ontology Alignment
  - Ontology Resolution

Giving these basic Functional Components the architecture defining the interrelation between each other was created. Also, after defining the relationships between the components the characterization of the features needed to be implemented in each FC was performed, attending always to the collected requirements. And finally, the technological decisions were made in the last stage of the design process to be ready to start the implementation stage later on.

2.1.2 Relationship of RA with other activities in WP4: INTER-FW and INTER-API

The relation with the INTER-IoT framework has been analyzed in the document D4.3 Initial Reference IoT Platform Meta-Architecture and Meta Data Model Interoperable IoT framework Model and Engine v1 released in September of 2017 in the Section 3.2: INTER-IoT RA Instantiation. The other activities in WP4 provide a comprehensive framework to expand the interoperability infrastructures in INTER-IoT (software development framework), configure and manage interoperable IoT platforms (configuration and management framework) and program IoT based applications relying interoperable platforms (API). These features are composed on the top of the layer infrastructure developed in WP3 and thus have themselves a transversal nature, being reflected almost all the components of the reference architecture in the modules that compose the technical solution.

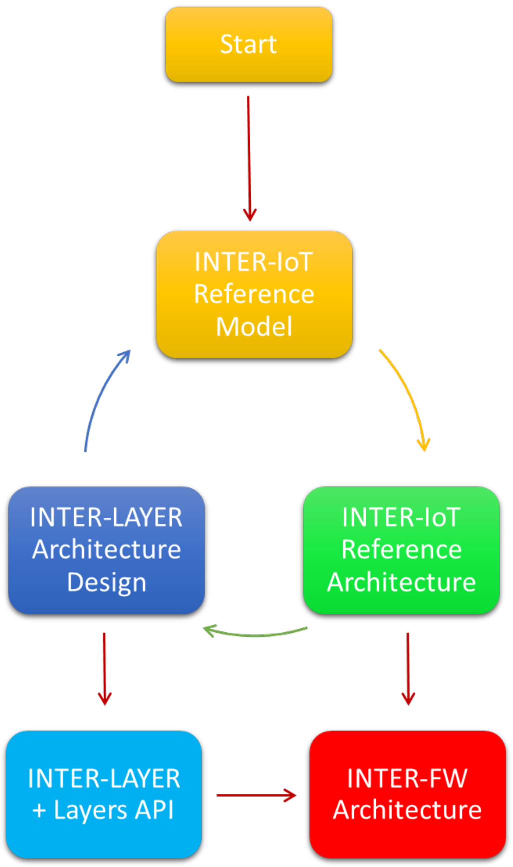
Start
INTER-IoT Reference Model
INTER-LAYER Architecture Design
INTER-IoT Reference Architecture
INTER-LAYER + Layers API
INTER-FW Architecture

Figure 2.1: Process followed for generating the INTER-FW architecture

More information about the implementation of the INTER-IoT RA in the WP4 activities can be found consulting the aforementioned deliverable, where it is explained how requirements are accomplished by the application of the designed RA and how the use cases leverage this architecture to find full-fledged interoperability at different layer levels.

2.1.3 Relationship of RA with WP5: INTER-Meth

INTER-METH is an engineering methodology that aims at supporting the integration process of heterogeneous IoT platforms to (i) obtain interoperability among them, and (ii) allow implementation and deployment of IoT applications on top of them. Specifically, INTER-METH is based on a process that defines and offers the following phases (or functionalities of a process):

1. Analysis phase supports the definition of the IoT platform integration requirements (both functional and non-functional).
2. Design phase produces the composition of the integration in terms of define artifacts (e.g. diagrams) for layer interoperability infrastructures and related interfaces (see INTER-Layer), and INTER-FW programming and management patterns, to fulfill the elicited requirements.
3. Implementation phase focuses on driving the implementation of the design work-product to obtain the full-working (hardware and/or software implemented) system.
4. Deployment phase involves the support to the operating set-up and configuration of the integrated IoT platform.
5. Testing phase defines the performance evaluation tests to validate the integrated platform according to the functional and non-functional requirements.
6. Maintenance phase manages the upgrade and evolution of the integrated system.

In WP5, INTER-METH is formalized according to two levels of refinement. At high level, an abstract INTER-METH process is defined and it is independent from other INTER-IoT products. At a more refined level, INTER-METH is instantiated in the INTER-IoT Framework and particularly it is related with the reference architecture (RA) discussed in D4.1 as well as the multi-layered interoperability architecture of INTER-IoT.

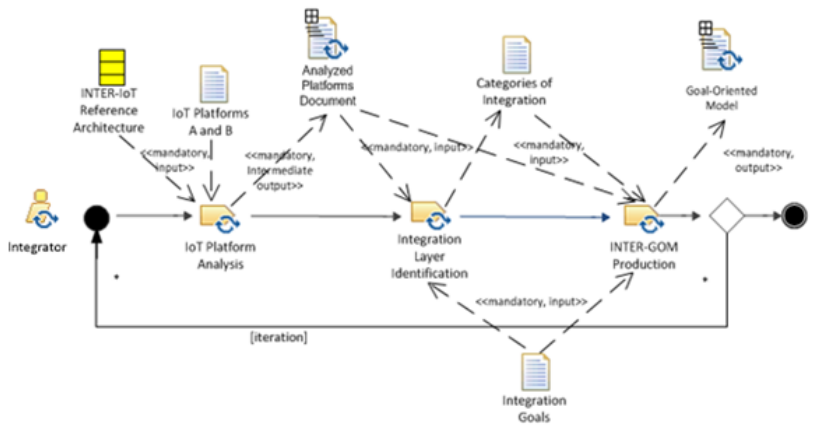

Figure 2.2: The workflow of tasks of the instantiated (Inter-IoT-based) Requirement Analysis activity.

In particular, the RA is used during the analysis phase (see the workflow depicted in Figure 2.2) as the platforms to be integrated are analyzed and compared according to the RA. Identifying the feasible and possibly most appropriate integration points is a non trivial task and having an homogeneous view of the platforms architecture is a fundamental prerequisite for this task. It is possible to define a general and a particular level (if desired) of refinement for the identification of integration points, in according to the RA: the coarse level identifies the Functional Groups, while the fine grained level identifies the Functional Components.

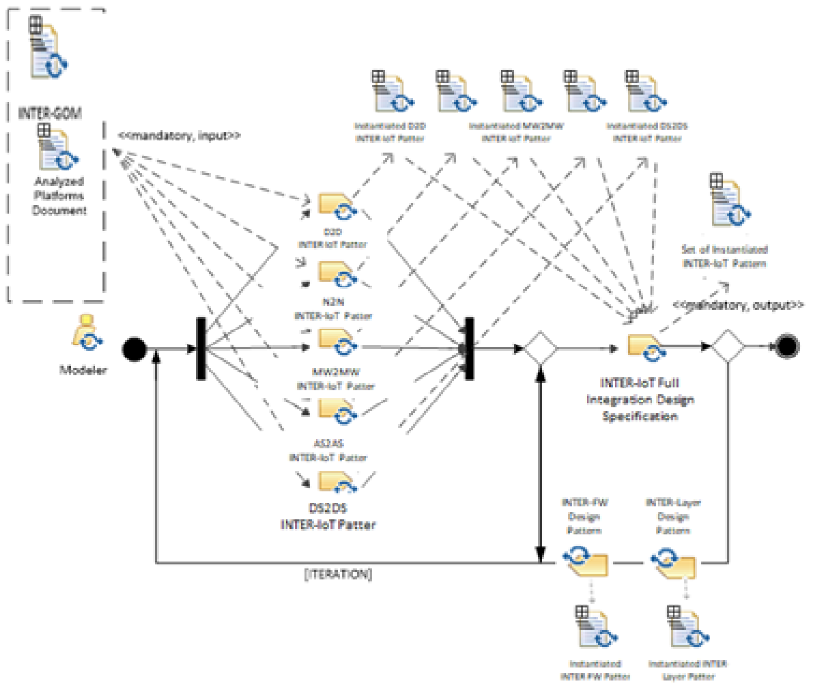
INTER-GOM
Analyzed Platforms Document
<<mandatory, input>>
Instantiated D2D INTER IoT Patter
Instantiated MW2MW INTER IoT Patter
Instantiated DS2DS INTER IoT Patter
Set of Instantiated INTER-IoT Pattern
<<mandatory, output>>
D2D INTER IoT Patter
N2N INTER-IoT Patter
MW2MW INTER-IoT Patter
AS2AS INTER-IoT Patter
DS2DS INTER-IoT Patter
Modeler
INTER-IoT Full Integration Design Specification
INTER-FW Design Pattern
INTER-Layer Design Pattern
[ITERATION]
Instantiated INTER FW Patter
Instantiated INTER Layer Patter

Figure 2.3: The workflow of tasks of the instantiated (Inter-IoT-based) Design Integration activity.

The RA comes also into play during the design integration phase (see the workflow depicted in Fig. 2.3) in terms of the Analyzed Platforms Document (in fact analyzed according to the RA) which is one of the inputs of this phase.

2.1.4 Relationship with WP6: Pilots

2.1.4.1 INTER-LogP

There is a very close relationship between the RA and INTER-LogP. Although each of the actors involved in the transport pilot had its own system, it could not be considered IoT. That is why it has been necessary to adapt these systems following the indications of the RA. Following the INTER-IoT Functional view, the platforms have many of the modules described for access to sensors, interoperability of sensors, interoperability with other platforms, data ontology, etc. In addition, INTER-LogP uses the solutions provided by INTER-Layer to achieve interoperability, both at the device level and with other platforms.
A more detailed description of this process will be defined in Section 6. Using several Use Cases it will be exemplified the application of the RA into the Pilot deployment.

2.1.4.2 INTER-Health

The relationship between RA and INTER-Health pilot is a full-duplex connection. In one direction the RA influences indirectly the pilot as through the actual implementation of INTER-Layer this pilot was designed. Given the scenarios and the quandary, INTER-Layer solutions were disposed to solve the interoperability problems of the system. And this was following the guidelines that the RA provided. As there was no pre-existing IoT infrastructure in the INTER-Health pilot, the IoT interoperability

mechanisms were designed to accommodate INTER-IoT according therefore to the RA. Analyzing the needs of the scenarios and applying the restrictions provided by the RA, different architectural decisions were made on the pilot. For example, while the platform is primarily designed with a server-based or cloud-based deployment, the security and legal restrictions imposed by the INTER-Health require that all equipment is executed locally at the pilot premises, without accessing external services. While not initially conceived for this kind of deployment, the RA, and its implementation, allows for this configuration.
Moreover, in the other direction, we understand the INTER-Health pilot as a proof of concept for the validity of the RA applied into real scenarios. With this approach and having the KPIs defined previously for the pilots we can assure a specific level of success for our defined RA when applied into solving complex IoT interoperability issues.
One of the keys of the INTER-Health pilot is the need to a full availability of the system, since it is bound to a clinical trial and thus it is fundamental that the data from the patients arrives whenever is generated to the central systems of the pilot. Other critical aspect is the need of protecting the data of the patients (classified as) to be compliant with the European laws, something also enforced by the Ethical Comittee. These two factors suggest a technological deployment highly tested and the complete avoidance of data flows out of ASLTO5 premises. For these reasons, the INTER-Health pilot explores a defined subset of interoperability mechanisms of the range of developments performed in INTER-IoT. These two are, at the time of writing this report and for the convenience of the scientific and technical outcomes of the pilot, the middleware-level interoperability and the device-level interoperability, and, consequently, the components of the RA instantiated in this pilot are limited to these two interoperability means. However, key aspects of the reference architecture are tested in such experience, in particular non-functional requirements as security, privacy or reliability related.[1]
To test other interoperability layers also described in this reference architecture in the domain of healthcare, the INTER-Domain pilot is planned, where data is less critical in terms of availability, allowing to experiment all the technological solutions developed in the framework of the INTER-IoT solution without exposing privacy or jeopardizing valuable scientific data.
In the same way as in INTER-LogP, a more detailed application of the RA into this pilot will be defined in Section 7 of this deliverable.

2.2 Relationship of the meta-data model with Work Packages

2.2.1 Relationship of meta-data model with WP3: INTER-Layer

2.2.1.1 MW2MW

As an introduction to this section, it is important to define some idioms related with data representation and processing in INTER-MW.
Semantic web provides a common framework, which promotes the common data formats and exchange protocols. It is used for two purposes, for common formatting and combining the data drawn from different sources and it is a language for recording the data relationships to real world object.
At the project level, a common INTER-IoT ontology (GOIoTP) has been defined which provides a common understanding of the IoT meta-structure, enables semantic interoperability and represents all messages routed through the system.
In the context of the INTER-MW implementation the following technologies are used in order to implement support for the INTER-IoT ontologies i.e. GOIoTP and GOIoTPex:

[1]See deliverable D2.3 Sections 3.1.1, 3.5.1 and 3.6.1 for further details

1. *RDF.* Resource Description Framework forms one of the basic building blocks for forming the web of semantic data and is used to define a type of database, called *triplestore*. The edges in a RDF graph represent named links between resources or values, and these are represented by the graph nodes. RDF uses URIs to name the relationships between entities, as well as the entities themselves. This connection in a graph (an edge and two nodes) is usually referred to in RDF as a triple. Each triple is comprised of a subject (a resource), a predicate (a relation), and an object (a value or a resource) .
2. *Triplestore.* Triplestore data bases have been built as subject-predicate-object as engines from scratch. Parliament[2] is a free and open source triplestore database and was designed for Semantic Web. The Parliament database (see D3.2, Section 2.3.5) is used for the INTER-IoT Middleware services which provides advanced querying mechanisms. It is used for the Service subsystems for platform and source registry, discovery and routing & roaming. Model for the Parliament instance follows GOIoTP and GOIoTPex.
3. *JSON-LD.* JSON-LD is inspired by JavaScript standard object literal syntax and is currently most frequent used data formats. It is a perfect language for formulating any kind of "self-describing" messages, which in addition can contain meta-information to specify the content of the messages.
4. *SPARQL RDF.* SPARQL can be used to express queries across diverse data sources, whether the data is stored natively as RDF or viewed as RDF via middleware[3]. SPARQL RDF is custom - tailored for complex queries.

Messages

The essence of the INTER-MW data model are messages expressed in the common ontology (see D3.2, Section 4.3.2.3). Each message is composed of a payload and message metadata. The latter one contains information, which is needed for the message routing within INTER-MW. There could be also the case when the payload contains essential information for routing, which is transferred to the metadata part after followed the message processing and finally routing within INTER-MW. The metadata should contain information about: message type, destination platform(s), source platform, message id and conversation id. For more details see D3.2, Section 4.3.6.
Messages are converted to JSON-LD (see D3.2, Section 2.3.5) for transportation and interfacing purposes, which is further divided into two RDF graphs: metadata and payload graph. **Metadata graph** uses INTER-MW RDF vocabulary to route and manage messages. In the D3.2, Section 4.3.2.3 is example of JSON-LD platform registration message, where can be seen the fields of the metadata instance: InterIoTMsg:messageID, InterIoTMsg:conversationID, InterIoTMsg:dateTimeStamp, InterIoTMsg:query, InterIoTMsg:errCode, etc.. **The payload graph** contains RDF code. The payload is uses the GOIoTPex semantics and utilizes IPSM to translate to and from it. Below are described the main components that are part of INTER-MW communication processes.

Data Flow Management

Since the essence of the INTER-MW data model are messages, it is also important to represent the conversation between INTER-MW, which uses messages for communication between its components. INTER-MW components communicate through channels (queues) with the purpose to connect

[2] http://parliament.semwebcentral.org/
[3] https://www.w3.org/TR/rdf-sparql-query/

with each other. The full list of queues (INTER-MW inter-component communication topics) can be seen in D3.2, Table 27, Section 4.4.2.2. Some of the queues represent messages sent between bridges and IPSRM. Conversations can be determined as permanent channels between API clients and middleware components. Initially, the conversation starts at the API Request Manager (ARM) level and creates a new *Conversation Id* and after return it to the caller. Components like *Bridges* and *Services* keep track of active conversation and identify responses to a appropriate conversation.

Bridges

Implementation of bridge means interoperability with a new platform and thus consequently means the implementation of communication protocol and syntactic translation of the messages (translated in JSON-LD data-interchange format, see D3.2, Section 2.3.5). Bridges in INTER-MW have a role of a middleman between INTER-MW and IoT platform. A bridge creates and processes the message using the information created by the IoT platform in the case platform wants to send a message upstream towards INTER-MW. Bridges also handles syntactic translation of payload between syntax in the data model and the IoT platform format (semantic translation performed in IPSM is done independently of the syntactic conversion performed by bridges), to sum up, communication through INTER-MW, which consists of the data models of platforms, uses a two step approach: syntactic and semantic translation. The data models from heterogeneous IoT platforms do not need to be the uniform, as long the bridges are developed for these platforms. The central ontologies in IPSM instances (see Section 2.2.1.2) used by pilot and demo INTER-MW deployments all used GOIoTP and GOIoTPex ontologies.

Services

Services are connected to Parliament triple store database, which allows fast and optimized execution of SPARQL RDF query language (by using message types such as QUERY and DISCOVERY, see Table 4.3 in D3.2, where the INTER-MW Message types are listed). The latter is very important, because the QUERY and DISCOVERY messages enable applications from outside as well as components within middleware to dynamically collect information about resources (connected platforms, things, etc.). Furthermore, QUERY and DISCOVERY types of messages allow INTER-MW users/developers to figure out the specifics of different connected platforms and discovery mechanisms they may support. The model in Parliament, as well as the SPARQL queries for discovery, use the INTER-IoT ontologies.

2.2.1.2 DS2DS

IPSM (Inter Platform Semantic Mediator) is a software developed for the DS2DS layer that enables semantic translation. IPSM design involves translation channels that connect two communicating artifacts, and translate the passing messages to the semantics declared by the receiver. IPSM communication architecture assumes the existence of a (potentially empty) central ontology (see D3.2 for more details about IPSM). The central ontology should be modular, and is defined independently per IPSM instance. It serves as a “middleman” semantics. Each message in a translation channel is first (semantically) translated to the central ontology semantics, and only then, to the receiver semantics. This architecture simplifies the process of adding more clients to an existing ecosystem, in which IPSM is used. Any new participant needs only to define alignments to and from the central ontology, and does not need to do this for any other artifact it wants to communicate with.

IPSM works well with any ontology, and is, in this sense, ontology-agnostic. Nevertheless for IoT applications GOIoTPex is the INTER-IoT recommended central ontology. It is modular, which simplifies engineering of new alignments, and covers a broad range of entities relevant to any IoT software. It was also successfully used as central ontology in IPSM instances used by demo and pilot INTER-MW deployments.

2.2.1.3 AS2AS

Customized Node-RED software is used in the AS2AS layer of INTER-IoT. With it, a user can visually define flows between so-called nodes, which represent services offered by platforms (or any other software). Every node has requirements regarding the input (e.g. data format and number of parallel inputs), and offers (optionally) some output in a given format, after processing the input data. Both input and output are accessible via a given protocol. The available services are known before-hand, meaning that each service needs to be registered, before it becomes available to the end-user. It is then displayed in a node catalog.
The service registry and discovery are based on the same technology used in INTER-MW - Parliament and SPARQL (see Section 2.2.1.1). The service module of GOIoTPex is used for description and searching/filtering of available registered services. Input and output descriptions of the services allows for matching compatible nodes, and thus assists in creation of Node-RED flows.

2.2.2 Relationship of meta-data model with WP6: Pilots

2.2.2.1 INTER-LogP

During the definition of the pilots in INTER-LogP, several sources of data have been identified that will be used during the pilots, which include port authority, haulier companies, terminal operators and others (see D6.2). This data will be shared with other companies and therefore it is necessary that they be able to understand them. That is why the pilot implementations of INTER-LogP use GOIoTPex as a base ontology, and extend it with port specific modules.
The main entities that can be extracted from the Meta Data Model that will be used in INTER-LogP are: truck access control data, environmental data, trucks data, and lighting data. While GOIoTPex provides a suitable base, the port environment is very specific and requires its own set of ontological entities.
However, any modification on the data syntax has been performed over the legacy systems. Hence, the INTER-IoT solution has adapted the Meta Data Model to allow the integration of all parts participating in the pilot. A further example will be itemized in Section 7. The same case as in the RA it will be shown how the Meta Data Model has influenced the development of the pilots.

2.2.2.2 INTER-Health

Similarly to INTER-LogP, the INTER-Health pilot implementation uses GOIoTPex as a base, and extends it with custom eHealth module. Even though the medical domain can be very complicated, when it comes to ontologies, we have found that GOIoTPex, and especially its User, Observation and Units modules cover the basic needs, without any extension. Using just those ontologies we were able to send smart weighing scale observations, and attach information about the measurement value or the patient being observed. Additional ontological entities needed to be included in order to cover all the possible information values required by the pilot, in particular for less general observation types such as blood pressure or medical questionnaires.

However, there is an additional value required for INTER-Health use cases that had not been initially covered: The identification of the doctor “ordering” the measurement. The inclusion of *iiot:orderedByUser* property in GOIoTP was the result of analysis of this INTER-Health requirement. For the medical domain it was crucial that the information about the doctor ordering a measurement is included in the meta-data.

3 INTER-IoT Reference Model

3.1 Introduction

Interoperability has been indentified as one of the major challenges in IoT in multiple forums by a comprehensive set of stakeholders[1] [2] [3] [4]. The INTER-IoT project proposes a wide range of solutions in order to cover as many aspects of the interoperability problems identified in the dawn of the connected objects. All these solutions are identified, planned and organized in a framework specially created for the domain of the interoperability of IoT artifacts during the execution of the project. This is the INTER-IoT Reference Model. The reference model helps in the identification of key concepts, defining them in a univoque way. The most extended and accepted definition of Reference Model comes from the OASIS association [5], which defines it as:

an abstract framework for understanding significant relationships among the entities of some environment, and for the development of consistent standards or specifications supporting that environment. A reference model is based on a small number of unifying concepts and may be used as a basis for education and explaining standards to a non-specialist. A reference model is not directly tied to any standards, technologies or other concrete implementation details, but it does seek to provide a common semantics that can be used unambiguously across and between different implementations.

The INTER-IoT Reference Model is proposed in section 3 of the document D4.1, released in January of 2017. The INTER-IoT RM description and fundamentals are based in the successful results of the IoT-A project, however, unlike the RM proposed in IoT-A, in INTER-IoT RM is fully focused in providing a model for interoperability of existing IoT Platforms. Although IoT-A also contains the concept of interoperability, this is considered a design constraint for the creation of new platforms. In spite, the approach followed in INTER-IoT is the provision of interoperability mechanisms for already existing platforms, which is more realistic for real scenarios in industry, health, cities, etc. The INTER-IoT RM proposed in document D4.1 and revised in this one, proposes a reference model for IoT interoperability mechanisms in real scenarios.

The following chapter contains an update of the reference model, including the analysis of the existing platforms involved in INTER-IoT pilots (INTER-LogP and INTER-Health) and some improvements in the definition of terms in the different subsections. In particular, the domain model has been updated including a reformulation of some terms, bringing a more uniform definition of concepts such as *services* or *resources*. These and other terms have been updated thanks to the analysis, design and

[1] http://blogs.ptc.com/2015/07/23/whos-going-to-service-all-those-things-in-the-iot/
[2] https://www.itu.int/en/ITU-T/academia/kaleidoscope/2017/Documents/special/ss03.pdf
[3] https://www.ietf.org/blog/2016/01/an-interoperable-internet-of-things/
[4] https://iot.ieee.org/newsletter/march-2017/three-major-challenges-facing-iot
[5] https://www.oasis-open.org/

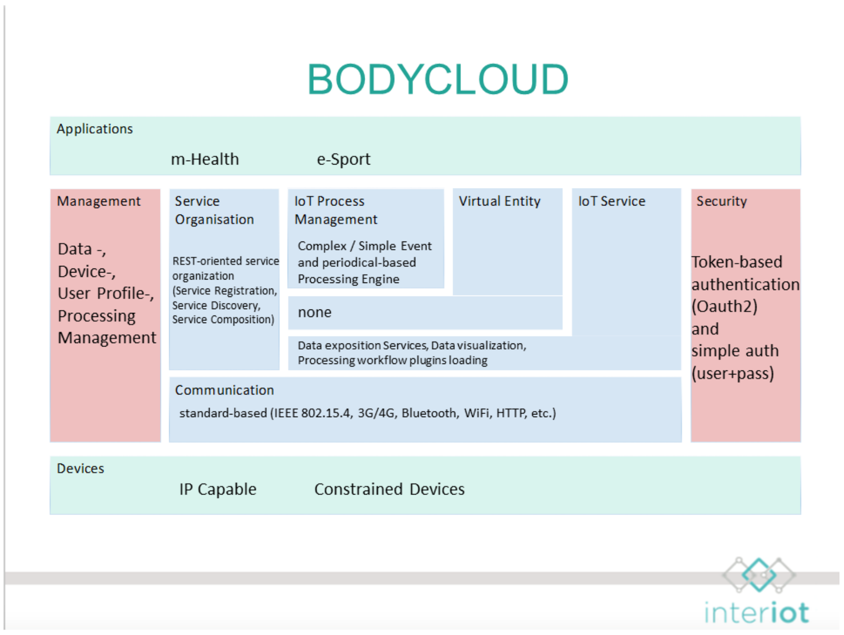
BODYCLOUD
Applications
m-Health
e-Sport
Management
Data -, Device-, User Profile-, Processing Management
Service Organisation
REST-oriented service organization (Service Registration, Service Discovery, Service Composition)
IoT Process Management
Complex / Simple Event and periodical-based Processing Engine
Virtual Entity
IoT Service
none
Data exposition Services, Data visualization, Processing workflow plugins loading
Security
Token-based authentication (Oauth2) and simple auth (user+pass)
Communication
standard-based (IEEE 802.15.4, 3G/4G, Bluetooth, WiFi, HTTP, etc.)
Devices
IP Capable
Constrained Devices
interiot

Figure 3.1: The workflow of tasks of the instantiated (Inter-IoT-based) Design Integration activity.

implementation works in the different work packages of the project for the different layers and the global framework (INTER-FW). Finally, a security model for IoT interoperability is proposed, offering general guidelines for interoperable secure deployments, as the used in the INTER-IoT pilots.

3.1.1 BodyCloud

The management of a multitude of body sensor networks (BSN) and their gathered data cannot be autonomously accomplished with their limited resources. BodyCloud (`http://bodycloud.dimes.unical.it`) tackles this problem by exploiting a Cloud computing infrastructure and providing an integrated platform, namely a Cloud-enabled BSN infrastructure, that offers:

- capabilities of using heterogeneous sensors through mobile devices acting as gateways,
- scalability of processing power for different kinds of analysis,
- scalability of data stream storage,
- ubiquitous and global access to the processing and storage infrastructure,
- easy sharing of results, and
- pay-as-you-go pricing for using BSN services.

In particular, BodyCloud is a distributed software framework for the rapid prototyping of large-scale BSN applications. Currently based on Google App Engine, it is designed as a Software-as-a-Service (SaaS) architecture to support the storage and management of sensor data streams and the processing and analysis of the stored data using software services hosted in the Cloud. BodyCloud aims to support several cross-disciplinary applications and specialized processing tasks. It enables

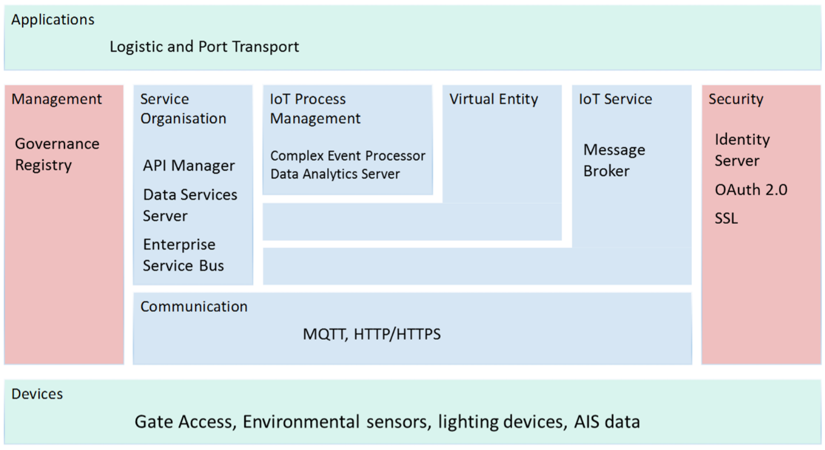

Figure 3.2: Valenciaport-Platform functional model

large-scale data sharing and collaborations among users and applications in the Cloud, and delivers Cloud services via sensor-rich mobile devices. BodyCloud also offers decision support services to take further actions based on the analyzed BSN data. Thanks to its design and implementation choices, BodyCloud can be flexibly tailored, in a very effective manner, for supporting a broad range of application domains, including m-Health, e-Sport, and environmental monitoring. In figure 6.2 it's possible to see an analysis of BodyCloud for the IoT-A Functional Model.

3.1.2 Valenciaport-Platform

The IoT platform of the port of Valencia, analysed in figure 3.2 aims to be a link between data and applications. On the one hand, there are many elements that monitor each of the elements in the port, such as sensors (environmental, light meters, etc.), automatic access gates, AIS (Automatic Identification System), etc. On the other hand, there are applications that offer services to the users of the port, such as Port Community System or Port Management System.
The IoT platform also processes, enriches and displays the data, so that the port managers can make decisions based on what is happening at each moment. The main modules that make up the IoT platform of the port are:

- **Governance and registry** – of the different element registered in the platforms such as sensors, devices, users. All this elements should be managed and updated.

- **API manager** – for publishing and management of all APIsare published and managed.

- **Data Service Server** – to manage the different access rights (reading and writing) to the different databases available.

- **Enterprise Service Bus** – for the exchange of data among the different components of the platform.

- **Complex Event Processor** – to define and analyze rules that allow conclusions to be drawn from the data received.

- **Data Analytics Server** – to analyze the data and detect patterns that can be used to optimize processes.
- **Message Broker** – to provide the publish/subscribe messaging application style that allows access to data in real time.
- **Identity Server** – to jointly manage the access security of all the components of the platform.
- **High Availability Scalable Database** – to provide a long term database for storing large amounts of data, ready to be used in data analytics.

3.1.3 SEAMS 2

The SEAMS 2 platform is an IoT platform aimed to gather data from machinery in an industrial environment, process it (calibrate, transform, homogenize) and obtain operational and business KPIs valuable for decision making and process optimization. To enable its basic features the platform comprises a set of modules:

- **Device Manager** – to perform basic operations with IoT Devices (Register/Unregister) via an easy-to-use UI, as well as more complex tasks such as defining calibration operations, accuracy of the sensors or groups of different machines.
- **Enterprise Service Bus** – to manage platform services and set up basic service interoperability with other platforms.
- **Calibration Unit** – to perform basic calibration operations over the raw data from the field machines.
- **Complex Event Processor** to enable stream processing, allowing KPIs composition, data transformation, correlation, and sequence detection.
- **Business Process Unit** – a domain-specific module to detect and calculate the business processes in the terminal operation, allowing flexible cost allocation and optimization in real time.
- **Search Engine** – to store and quickly explore the short-term history data.
- **High Availability Scalable Database** a long term database ready to be used in data analytics, big data techniques, and machine learning.
- **Advanced visualization module** – a tool to reveal patterns and bottlenecks, represent KPIs, perform geospatial-based analysis and control of assets, for both short and long-term data.
- **REST API** – to enable new developments and integration with other internal or external systems. The API is prepared to retrieve both short term and long term data in responsive, quick operations.

The SEAMS 2 Platform for Terminal Operations Analytics supports the acquisition of data in MQTT, AMQP, Kafka and HTTP transport protocols and is prepared to provide a high available to tolerate the harsh conditions for wireless communications present in a cargo sea terminal.

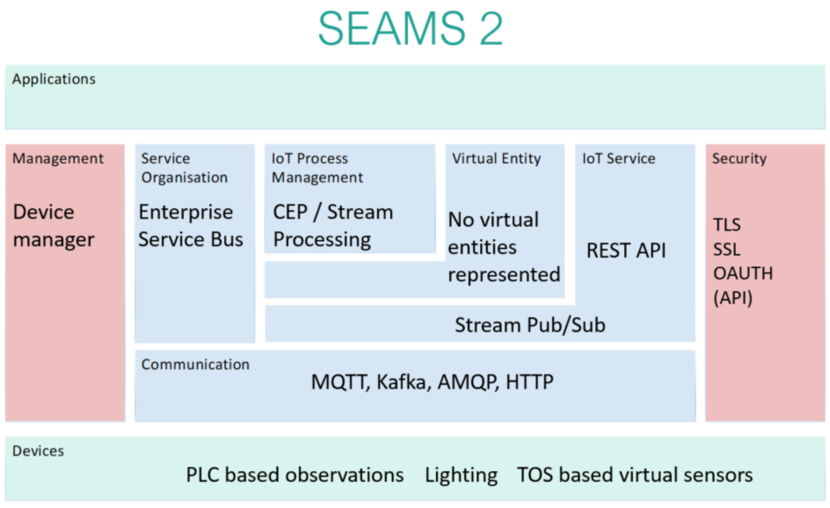

Figure 3.3: The workflow of tasks of the instantiated (Inter-IoT-based) Design Integration activity.

3.2 Domain Model

As in D4.1, the Domain Model is based upon the IoT-A's Domain model. After the release of D4.1, some changes have been added to the Domain Model. On one hand, a generalization has been added to the Service entities related to IoT Platforms. On the other hand, more entities have been identified related to the interoperability services that may appear when two or more platforms need to be interconnected, and also for modeling the semantic interoperability among platforms.
In Figure 3.4 the proposed Domain Model for interoperability of IoT Platforms is shown. IoT-A's colors have been extended to include two new types of entities:

- Purple: new entities that are intrinsic to each IoT Platform.
- Light brown: new entities that are outside the scope of an IoT Platform, and which are necessary for the interoperability of IoT Platforms.

Regarding the new entities related to IoT Platforms, a new abstract **Service** entity has been created. A *Service* represents an *Active Digital Artifact* that provides a generic service of an IoT Platform. The already existing *Platform Service* and *IoT Service* extend this *Service* to provide two different types of services, as was described in D4.1, which in short are:

- **IoT Service**: Mechanism to interact with specific *Resources* related to *Virtual Entities*, like query, subscribe, insert, etc.
- **Platform Service**: Complex services provided by the platform not directly related to a specific *Resource* or *Virtual Entity*, usually processing, monitoring, etc.

Regarding the new entities necessary for the interoperability of IoT Platforms, the following entities have been identified:

- **Interoperability Service**: The Platform Interoperability Service defines the abstract concept of making interoperable different Services at different platforms. They include interoperability mechanisms among homogeneous Services available at the IoT Platforms, like interoperability

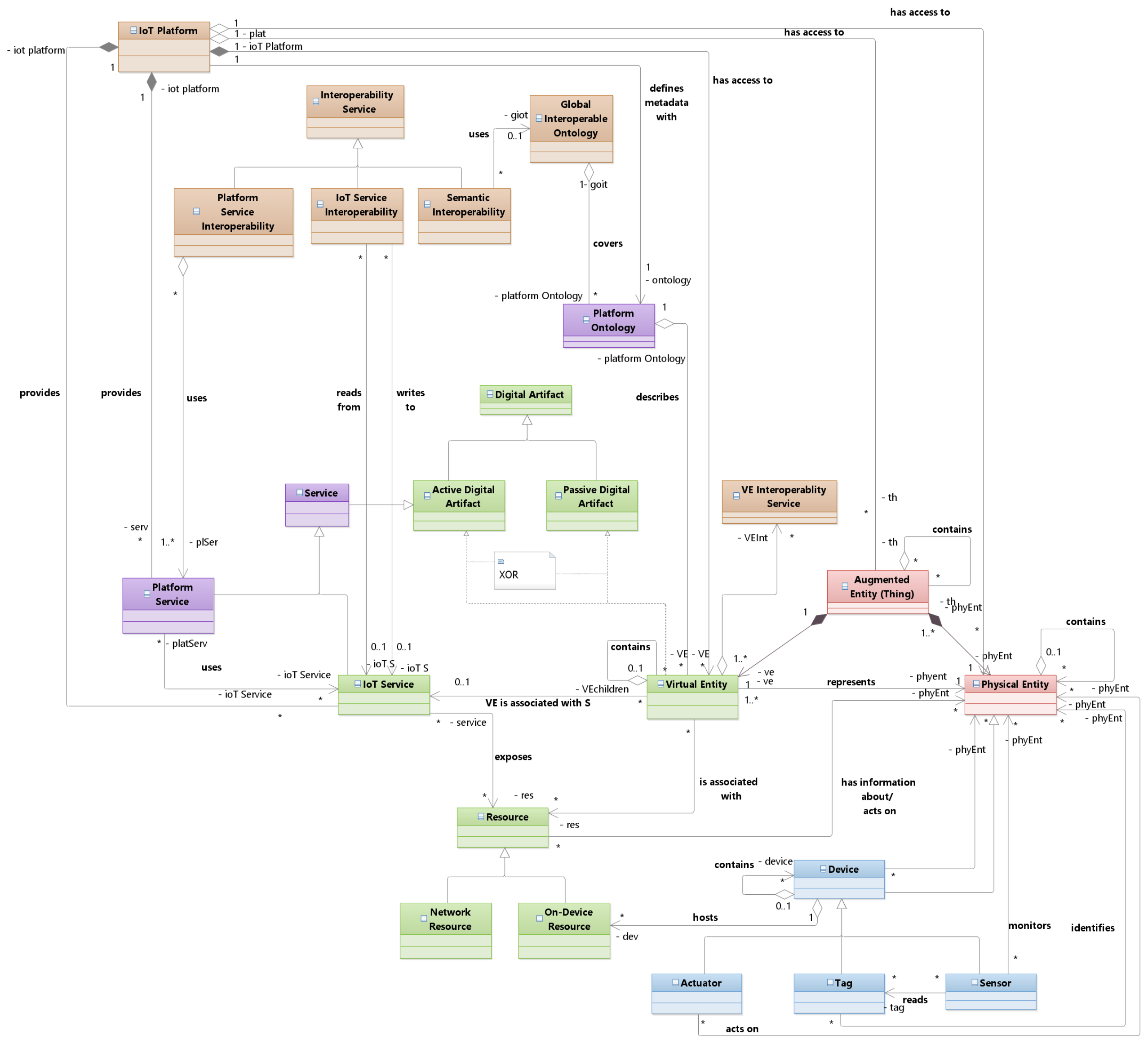

Figure 3.4: INTER-IoT generic domain model.

among IoT Services (e.g. data access) or interoperability among Platform Services (e.g. stream processing). The Platform Interoperability Service is extended in the following entities which model the different types of interoperability among Services:

- **IoT Service Interoperability**: An *IoT Service Interoperability* is a specific type of *Platform Interoperability Service* which provides interoperability among *IoT Services* which already exist at different IoT Platforms. This can be used, for instance, to read sensor measurements from a origin IoT Platform through a IoT Service, and inserting those measurements into a destination IoT Platform using an existing IoT Service at the destination IoT Platform.
- **Platform Service Interoperability**: A *Platform Service Interoperability* is a specific type of *Platform Interoperability Service* which enables the creation of new derived services that are made up of existing *Platforms Services* or that allow the interaction with *Platforms Services* that exist in different IoT Platforms.
- **Semantic Interoperability**: A *Semantic Interoperability* is the entity responsible for performing the translation of content among the different IoT Platforms that are to be interoperated. It uses the *Generic Ontology of IoT Platforms* and manages the definition of the ontology alignments for making the semantic translation from a source ontology to a target ontology.

• **Generic Ontology of IoT Platforms**: The *Generic Ontology of IoT Platforms* is a dynamic concept for achieving semantic translation among IoT Platforms. Rather than defining a brand-new full ontology, that may be broad enough to deal with any ontology, INTER-IoT approach for the semantic interoperability relies on a dynamic ontology. It should be an extendible ontology design, that extends a core ontology, with the ability to grow for adding new ontologies and features from the different IoT Platforms that need to be interoperated. Thus, when adding new IoT Platforms to a scenario where there are other platforms with other ontologies, the **Generic Ontology of IoT Platforms** should be extended to cover all the features of the different ontologies.

After the release of D4.1, it was also identified that a new entity could be created for representing controllers, understanding them as a specific type of device that controls a set of basic devices with no communication capabilities, typically legacy or simple sensors. Nevertheless, the auto-relationship of a *Device* in Figure 3.4, which contains a set of *Devices* can be used for modeling these controllers, so no new entity has been considered necessary. Modeling of controllers or physical gateways can be done through the *Device* entity.

3.3 Functional Model

During this year, we refined the previous INTER-IoT Functional Model. As a reminder, the INTER-IoT Functional Model main purpose is to break up complexity in systems by grouping similar Functional Components (FC) together. The Functional Decomposition (FD) is the process by which different FCs are identified and related to one another into Functional Groups (FG).
Our Functional Model philosophy changed: instead of identifying with precision all different groups, for simplicity sake we layered the different FC into specific FG, which are superimposed.

• At the bottom, we put the FG **Device**. This FG regroups all the Devices according to the definition in the Domain Model.

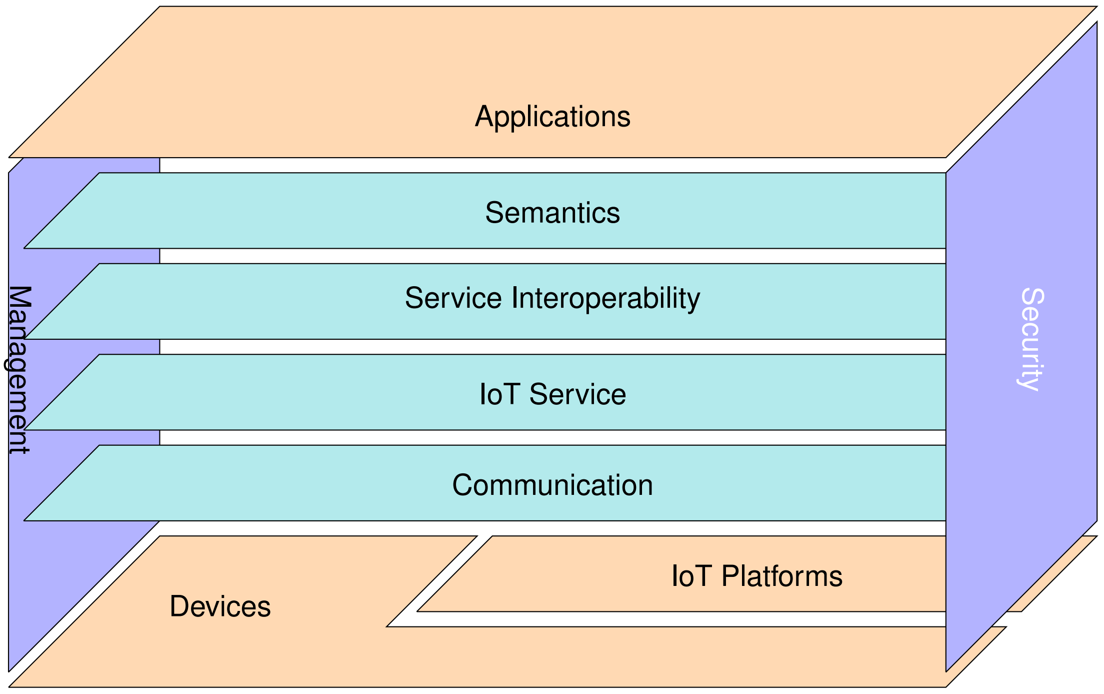

Figure 3.5: Overall INTER-IoT Functional Model.

- However, as INTER-IoT also deals with Platforms that opaquely cover some of the devices, another FG has been made: this is the the **IoT Platform** FG. Those two FG are outside the scope of the INTER-IoT FM, and are outlined only to show where the original data and interaction are based upon.

- As both Devices and IoT Platforms need to transfer information, a FG **Communication** has been identified.

- A number of main abstractions identified in the Domain Model have been put into the **IoT Service** FG, namely Service, Platform Service, Digital Artifact, Virtual Entities, Resources.

- The **Service Interoperability** FG Includes the Interoperability Service FG and the VE Interoperability Service.

- In the **Ontology** FG there are the Platform Ontology and the Global Interoperability Ontology.

- The **Application** FG corresponds to the upper layers. As defined in the requirements, the users of INTER-IoT will be also applications or systems willing to access the different platforms.

- To address consistently the concern expressed about IoT Trust, Security and Privacy in the interoperability realm, the need for a transversal **Security** FG is identified.

- Finally, the transversal **Management** FG is required for the management of and/or interaction between the functionality groups.

3.3.1 Communication

The Communication FG has a fundamental importance in any IoT modeling, as it provides the capability for different entities to transfer information. In INTER-IoT case, entities may be either devices or IoT platforms. Basically, this FG is represented by the Communication Model (see Section 3.4). In this domain, there is the capability of translating between different communication protocols, or to encapsulate some communication with envelopes able to carry the data towards Internet-capable proxies. As well, given the fact that in the IoT domain there are constrained devices that may not be able to offer typical End-to-End properties, like encryption, this FG is instrumental in setting up appropriate schemes and abstractions for overcoming those issues - for instance, by using a specific proxy or gateway that masks the limited capabilities of a constrained device, acting as the end node towards external nodes communicating with it.

3.3.2 IoT Service

The IoT Service FGs include functions that relate to interactions on the Virtual Entity, IoT Service and Digital Artifacts abstraction levels, respectively. To give a practical example, we can say that In the physical world there are a number of Devices that sense and modify the environment. The Resources associated to these Sensors and Actuators are exposed as IoT Services on the IoT Service level. Interactions between applications and the IoT system on this abstraction level are about reading sensors data, or setting specific actuators. In order to apply these services, there must be a communication scheme already in place between the devices and the final users; furthermore, any Application can only interact with these Services if they already know the semantics of the values. Therefore, on this level no semantics is encoded in the information itself, nor does the IoT system have this information, it has to be a-priori shared between the Sensor and the application.
In INTER-IoT Model, we decided to simplify the original IoT-A model by including the Virtual Entity FG in this FG. In general, Virtual Entity (VE) level models higher-level aspects of the physical world, and these aspects can be used for discovering Services. However, as there always must be a relation between a VE and an IoT Service, for simplicity we can assume that the IoT Service always relies on a VE. This level of abstraction is below the Service Interoperability one, as the latter is about having independent Services cooperate towards a goal.
The IoT Service Functional Group contains IoT Services as well as functionalities for discovery, look-up, and name resolution of IoT Services.

3.3.3 Service Interoperability

The Service Interoperability FG is central for any kind of interaction between different entities, in particular between IoT Platforms. Its main goal is to provide the abstractions necessary for all interaction, based on the IoT Services and according to a specific Ontology. It expands in many ways the Service Organization FG from the IoT-A Functional Model. In IoT-A, the Service Organization FG is responsible for resolving and orchestrating IoT Services and also deal with the composition and choreography of Services. Service Composition is a central concept within the architecture, since IoT Services are very frequently capable of rather limited functionality due to the constraints in computing power and battery life that are typical for WS&ANs or embedded Devices comprising the IoT. Service Composition then helps combining multiple of such basic Services in order to answer requests at a higher level of Service abstraction (e.g. the combination of a humidity sensing Service and a temperature Service could serve as input for an air-conditioning). Service Choreography is a concept that supports brokerage of Services so that Services can subscribe to other Services available in the

system. Within INTER-IoT, the Service Interoperability FG allows not only a choreography, orchestration and composition between different services belonging to the same system, but also between different systems.
Therefore, the Service Interoperability FG relates to the need to interoperate different IoT Platforms at the Service layers. Interoperability between IoT Platforms can be handled at different layers (e.g. device, middleware, service, etc.). The Service Interoperability FG works at the service layer of each platform, regardless of their underlying infrastructure. The overall goal of the Service Interoperability FG is to provide a uniform service view to the Semantics FG. An example of this would be a service aimed at offering access to historical data about traffic intensity in a region, when that service needs to access a historical data service from different government organizations (e.g. National Roads, Regional Road and Local Road Agencies). Therefore, the Service Interoperability FG is responsible for accessing and using services that already exist in heterogeneous IoT Platforms. It also needs to provide a means to design the new ”compound” services, based on the composition, orchestration and choreography of existing services. The Service Interoperability FG interacts with the Semantics FG for requesting semantics features needed for the execution of the services, like, for instance, aligning different ontologies used by different IoT Platforms.
It’s important to highlight that the Service Interoperability FG is the responsible for talking with the IoT Platforms, but not for implementing any of the features that the IoT Platforms provide (what, in IoT-A’s Functional Model, is described in the IoT Process, IoT Service or Virtual Entities FGs). The Service Interoperability FG has three main functions:

- To enable the access to different Entities. This includes the use of the appropriate protocols and APIs at middleware level that each platform exposes.
- To keep track of the interconnected IoT Platforms and their devices, so that they can easily be found, when needed. This allows the remaining groups to not to know about the location of the platforms, or how the devices are connected to them.
- To perform device and platform interactions, like querying data from different devices and platforms in a common way, mapping sensor data flows from a source to a destination, offering subscriptions to sensor data, etc.

This FG makes use of the Semantics FG, for instance, to translate ontologies from data flowing from heterogeneous IoT Platforms with its own ontology, into a common one to be provided to a user at the Application FG.

3.3.4 Semantics

The role of the Semantics FG is to deal with the management of ontologies that are needed for making IoT Platforms interoperable. Traditional interoperability designs leave semantics tasks to the Application side, but this approach lacks the necessary interoperability features. For instance, no common data processing can be made at any component, as data ontologies are unknown. Compound services are then very limited without semantic support, as the data from different platforms is not compatible due to the lack of ontology. We consider semantics essential for interoperating IoT Platforms without transferring responsibilities to the end user. The Semantics FG is the responsible for providing support to all the management of ontologies needed at INTER-IoT. It defines the core ontology used for interoperating a specific set of IoT Platforms, each with its own ontology. It is also able to identify the ontologies used at the different platforms interoperated for the different devices or services providing information. One of the main functions of the Semantics FG is to perform, so called, ontology alignment, which means to perform the translation from an origin ontology (maybe from an IoT Platform) to a target ontology (maybe needed by a destination IoT Platform). This ontology alignment process is just a step needed to perform the semantic translation of content among IoT Platforms, which is the final goal of the Semantics FG. The semantic translation among platforms, provided by the Semantics FG offers the following functions:

- Identify or define the origin or destination ontologies of the data involved in a data communication between IoT Platforms.
- Perform the ontology alignment from these origin ontologies to a common ontology.
- Perform the ontology alignment from the common ontology to the destination ontology. The Semantics FG can provide its capabilities to several FGs with different purposes:
- Service Interoperability FG. It allows the Service Interoperability FG to perform alignment of data ontologies from different IoT Platform services so that common service processing can be done.
- Platform Interoperability FG. The Platform Interoperability FG can use this FG when particular services need to translate ontologies from data flowing from heterogeneous IoT Platforms with its own ontology into a common one to be provided to a user at the Application FG or, for instance, to interconnect sensor data from one to another platform each one of them having different ontologies.
- Device Interoperability FG. It allows this layer to perform ontology translation of data between devices, when making Device to Device interconnections, if data format or data ontology is different.
- Application FG. Although users, at the Application FG, will usually need to use the Service Interoperability FG, Platform Interoperability FG or Device Interoperability FG to make IoT Platforms interoperable in different ways, there is a possibility that the services provided by the Semantics FG can be of high value to an external user. This is a secondary functionality of interoperable IoT Platforms, but it's considered interesting when, for instance, a user wants to orchestrate its own services using raw data from different IoT Platforms and this data needs to be semantically homogenized.

3.3.5 Security

The Security FG is responsible for ensuring all the security aspects involved in the interoperability of IoT Platforms. The security in our realm has two faces:

- Management of the security aspects related to the connection with underlying IoT Platforms. This implies to accomplish with the different security features that the platforms require. INTER-IoT will need to tackle the user authentication for connecting to a platform, the authorization management (e.g. use of authentication tokens) and the encryption of some communications. Moreover, the access to the different IoT Platforms maybe user-based or anonymized depending on the decision of platform owners, so it must be handled by INTER-IoT with flexibility for each scenario.
- Management of the internal security of INTER-IoT. The connection to INTER-IoT must be secured, with appropriate authentication capabilities, and authorization management. The identity

of each user must be preserved, so much for keeping the identification until the IoT Platform, as to keep track of the anonymization when talking with the IoT Platforms. This internal security also implies the permission assignment to specific IoT Platforms and its resources (devices, services, etc.) under certain conditions. for instance, a platform owner may will to give access to a subset of devices to a set of user roles, but only within a time range, or when mobile devices are at a certain location.

The Security FG interacts with all the different groups and will allow that certain accesses are made, or that certain interconnections between two platforms are authorized or not.

3.3.6 Management

The Management FC considers all the functionalities needed to rule the interoperability among different IoT Platforms. The Management FC is thus, responsible for initializing, monitoring and modifying the operation of the interoperability among IoT Platforms. The main reasons for needing management fall within the following groups:

- **Cost Reduction** Users, obviously want to operate a system at the lowest possible cost. This implies that the design of the solution should satisfy a great number of potential users and situations so that the cost can be recovered among many users. To achieve this, the design should be as multipurpose as possible. It means that the system should parametrized to a wide range of scenarios and user needs. The Management FC will be responsible for setting up these parameters for any final deployment of INTER-IoT.

- **Lack of design experience** We cannot assume that users of INTER-IoT will always be high-skilled IT engineers that can easily understand all the concepts and apply them right, finding good solutions for each and every problem. Some of the problems that will face our end users will arise during the operation phase of the system, not during the design phase. For instance, an IoT Platform can decelerate its performance or even shutdown completely, some devices can have malfunctions overloading with irrelevant data, some external component can inject too much traffic in form of requests, like a DDoS attack or a service become unavailable at a certain moment.

  To address this reality, the Management FC will need to include capabilities to mitigate the impact of these issues without a necessary good design of the interoperability made by an INTER-IoT user. Some examples of this would be to monitor IoT Platforms state or to handle incoming requests.

- **Fault Handling** Failures are inherent to any operating system. They can have many causes, not being possible to prevent all the failures. As the consequences of these failures can be very severe, it's necessary that the Management FC includes strategies and actions to control the operation of the interoperability solution. Such control implies the monitoring of the whole system, the prediction of potential failures, the detection of existing failures, the mitigation of their effects and, if possible, to repair them. The Management FC is responsible for addressing these features, through monitoring capabilities and the possibility to change operational parameters during run time, such as platform and device registries, communication channel re-mapping, service catalog status, etc.

- **Flexibility** Traditional interoperability design is based on specific user requirements, which drive the design of a specific solution by, for instance, defining specific communications or translations between two IoT Platforms. The danger of this approach is that, on one hand, requirements can change in time, affecting the already deployed solution, and on the other hand, each interoperability scenario may have its own specific requirements. Instead of designing a new interoperability solution each time, it is better to include some flexibility in INTER-IoT, so that the Management FC can adapt to different situations and react to changes during the operational phase. Some flexibility features have been included in the requirements. The Management FC is responsible for supporting these features during the operational phase. Some examples of this is the ability to use different ontologies for the same IoT Platform and change them during runtime, to be able to define new services and have them available, or to define new rules for making devices interoperable among them.

3.4 Communication Model

3.4.1 Communication protocols on IoT Platforms

As presented in the previous release of this deliverable (D4.1), the Communication Model aims at defining the paradigms for connecting the elements that compose any IoT system. These elements have to be previously defined by the Domain Model (DM). Also, this model is certainly less critical in some application scenarios than in others, and thus, not strictly mandatory. First of all, a research about protocols that are most frequently used in IoT systems was carried out. This information was basic at the time of applying to our own deployments or pilots. INTER-Layer Table 3.1 shows the communication protocols of the main platforms used in the INTER-Health and INTER-LogP pilots. Other platforms will be used in the context of INTER-Domain and the Open Call pilots.

Table 3.1: Communication protocols on IoT Platforms used in the pilots.

	Application Layer						Network/Link Layer			
	HTTP	MQTT	CoAP	LWM2M	AMQP	UPnP	BLE(802.15)	ZigBee(802.15)	WiFi(802.11)	Others
FIWARE	✓	✓	✓	✓						IoT-Agents
OM2M	✓		✓					✓		KNX and HUE
UniversAAL	✓							✓		ZWave, KNX
SensiNact[6]	✓	✓	✓			✓		✓		
Body Cloud	✓						✓		✓	
SEAMS2	✓	✓			✓				✓	Apache Kafka
Valenciaport-Platform	✓	✓							✓	Sigfox, 3G
Hauler Platform		✓								3G

. At the Communication Model level, we study the different communication protocols that are usually utilized by any IoT system to exchange information between elements belonging to a certain layer of the IoT stack. Thus, we consider protocols ranging from the device level to the application level. Even at the data and semantic level, although no communication protocols are used, data formats (as JSON-LD) have to be understood by both sides in order to exchange information in a meaningful way.

Additionally, as noticed in the previous deliverable, the Communication Model involves those Users that have to exchange information. These Users can be divided in; Human Users, Services or Artifacts. Moreover, the communication between these Users has to support two paradigms; Unicast and Multicast. And also, three different environments can be covered in the CM; when both Users belong to a normal (non-constrained) network, when just one User belongs to a constrained network and when both Users belong to a constrained network.

Taking this three environments, the Communication Model indicates that in the first one there is no need of translator or gateways to be implemented in the communication. Moreover, a non-constrained protocol is used for the USer to communicate. However in the second case there is a need to a gateway to translate the protocols and the information from the non-constrained network to the constrained one. Finally, in the third environment there is also the need of use a gateway of constrained protocols (e.g., 6LoWPAN, UDP/CoAP, etc.) as both Users belong to a constrained network and they want to exchange information.
After analyze the Communication Model environments and make the detailed research on platforms and elements that INTER-IoT desire to interoperate, we adopt the use of gateways or brokers to communicate and exchange information between elements presented in different IoT systems, so that, in different networks. Simplifying the INTER-Layer solutions, and being the aim of these the creation of interoperability between elements from the same of different IoT system, it easy to see that the three environments are perfectly covered as mediators that understand several protocols are implemented at each of the layers

- Device Layer; Implementation of a gateway.
- Network Layer; Implementation of an SDN network (with controller) to make the translation from constrained sensor networks to IP-based networks with the collaboration of the gateway.
- Platform Layer; Implementation of a middleware that acts as a broker or intermediary between platform communication.
- Application Layer; Implementation of a service compose modeler to orchestrate the composition of services.
- Data and Semantic Layer; Implementation of a channel-based semantic mediator.

However, once we have the solutions implemented at different layers (INTER-Layer), the communication of these solutions with other systems, e.g. using INTER-FW, is carried out without any kind of constrain. That is, both sites or artifacts that want to communicate are located in a non-constrained environment, so that, the communication between INTER-IoT interoperability solutions and INTER-FW is performed over common non-constrained protocols e.g. HTTP with REST, and without the use of any broker, mediator, gateway or intermediary.

3.4.2 INTER-IoT Domain Model element communications

3.4.2.1 Device to Device Interactions

Components from the Domain Model that involve device interaction have not changed, consequently, the same section from Deliverable 4.1 is still relevant.

3.4.2.2 Network to Network Interactions

In this interaction the elements or entities involved in the communication at network level are the same as in D4.1, but we have to add an extra use case or example. This example shows the communication between elements within the network (e.g service in the network with a network element as a virtual switch).
Example: Resource on the network interacts with a virtual element

In this example there is a network resource, that connects to an entity, specifically virtual entity located on the network. This example seems simple but involves quite important process that is the control and monitoring of network resources.

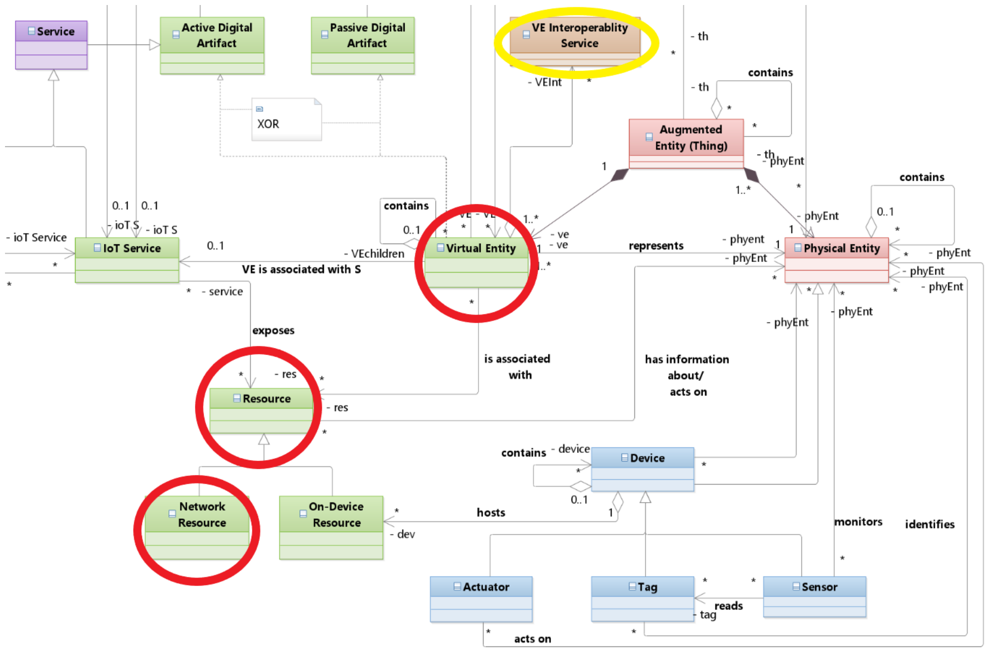

Figure 3.6: Domain Model entities involved in Network-to-Network communication.

In some particular cases, an inclusion of another entity of the Domain Model, that is the Virtual Entity Interoperability Service might be needed. This element is required when an Interoperability Service is running in some node of the network and performs gathering of information from elements located on different network domains.

3.4.2.3 Middleware to Middleware Interactions

Due to the changes that have been made to the Domain Model, currently, some additional entities have to be considered in case platforms need to be interconnected. Hence, Middleware to Middleware interactions have changed (see D4.1, Section 3.5.3.3). The components that interact with underlying IoT platforms are the entities represented by *IoT Platform* and *Platform Ontology*, which is now extended with the *Generic Ontology of IoT Platforms* module – see Section 3.2. This module, is a new concept that supports semantic translation between IoT Platforms while the *IoT Platform* entity represents knowledge about a specific IoT middleware deployment. Thus, these two entities appear in all Middleware to Middleware communication scenarios. In specific cases, they are aided by two other additions: newly added *Platform Interoperability Service* and *Platform Service*. There are also additional entities, *Interoperability Service* with extended entities, each with different interoperability feature among Services. Platform *Interoperability Service* is an abstract concept which main purpose is to interoperable different Services from Platforms. The implementation of the Domain Model in the MW2MW and DS2DS architecture, described in D3.2, is partitioned through various segments: (i) The *IoT Platform* is implemented in Bridges segment; (ii) *Interoperability Service* is implemented in Services segment; (iii) *Generic Ontology of IoT Platforms* is utilized by IPSM (via alignments with platform ontologies), Services & Communication, and Control segments; (iv) *Platform Ontology* is in Services; and (v)*IoT Service* is implemented in specific IoT Platform.

Examples of Middleware to Middleware interactions, with included additional entities, are:

Example: User accessing an IoT Platform through INTER-MW

The interaction between user and IoT Platform has changed. The *Platform Ontology* used in previous interaction (see D4.1, Section 3.5.3.3) is replaced with the *Generic Ontology of IoT Platforms* which should extend, and also cover, all the features of different ontologies. Through the *Interoperability Services*, users have unified access to IoT Platform issues. The *IoT Platform* entity is used in the interaction between user and IoT Platform (marked with red circle, see picture 3.7) and thus, with their assistance, the interaction with specific segments of underlying IoT Platforms (marked with yellow circle) is now possible (*Virtual Entity, Augmented Entity, Physical Entity*).

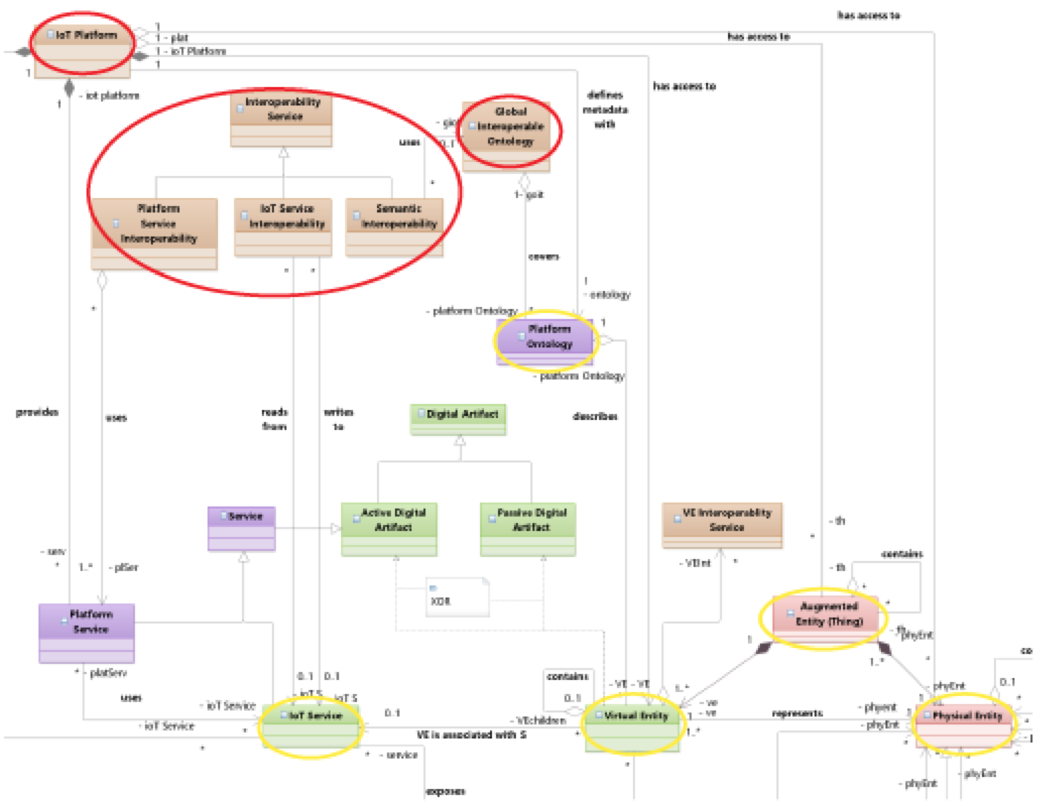

Figure 3.7: Domain Model entities involved in Middleware-to-Middleware communication when Human User accesses an IoT Platform through INTER-MW.

Example: User configuring an IoT Platform through INTER-MW

When the user attempts to configure an IoT Platform through the INTER-MW, the entity *Generic Ontology of IoT Platforms*, which represents core ontology, is needed and, in case a new Platform is added, the *Generic Ontology of IoT Platforms* is then extended to cover all the features of the different ontologies stored in *Platform Ontology*. The entities like *IoT Platform* (implementation of the Platform itself), *Platform Service* (provides the support configuration of a specific platform) and additional entity *Interoperability Service*, which supports the configuration of different Services at different platforms (see Figure 3.8), are also needed.

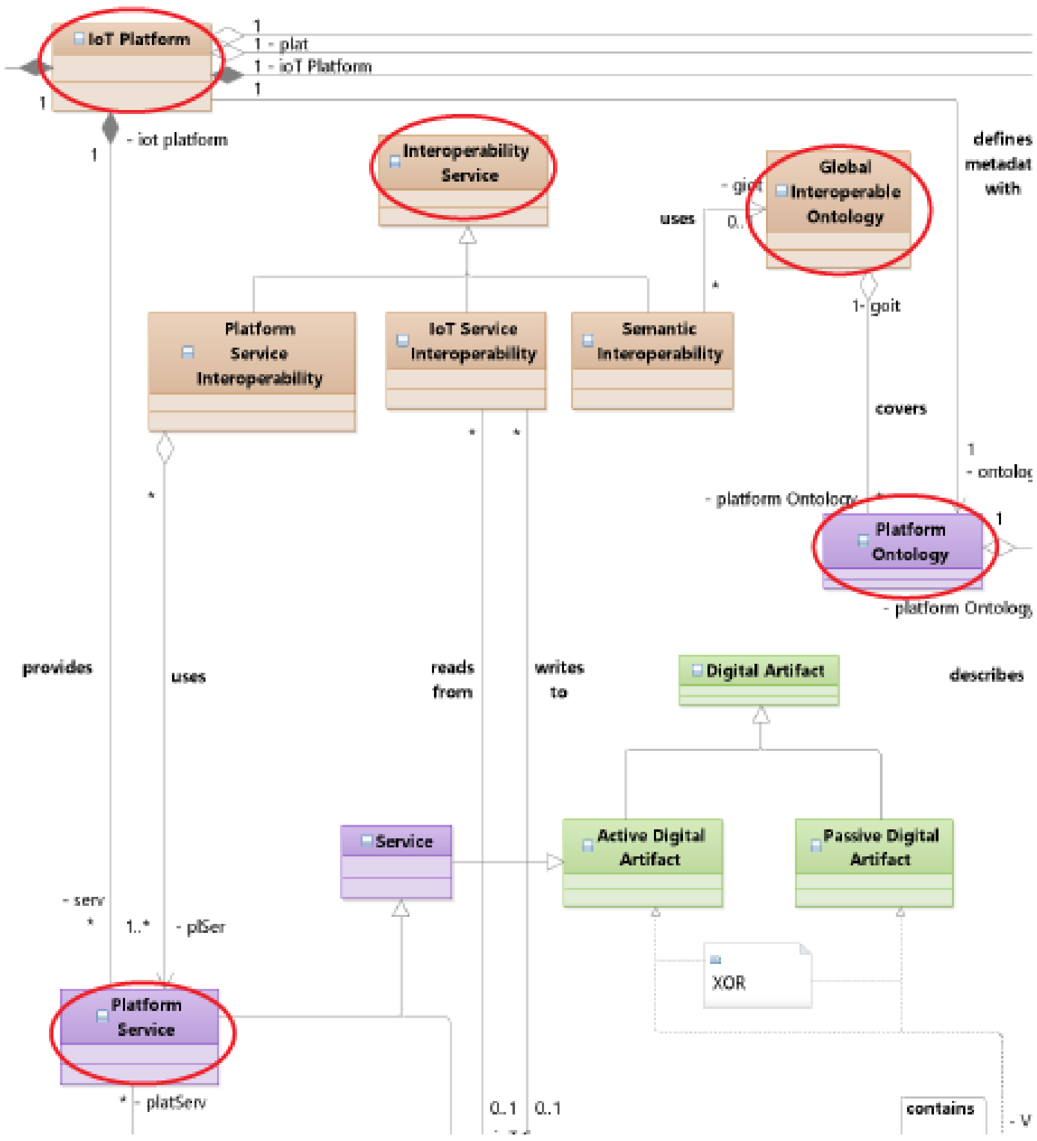

Figure 3.8: Domain Model entities involved in Middleware-to-Middleware communication when User tries to configure an IoT Platform

Example: Direct Middleware-Middleware communication between IoT Platforms through INTER-MW

Communication between IoT Platforms through INTER-MW creates the alignments generated by each *Platform ontology* separately, which is used by *Generic Ontology of IoT Platforms* entity. The latter enables communication without a mutual understanding of ontologies. The *IoT Service* entity of one Platform initiates communication/interaction with different devices and is part of *IoT Platform* entity, which provides the explicit ontologies used within platform (see Figure 3.9).

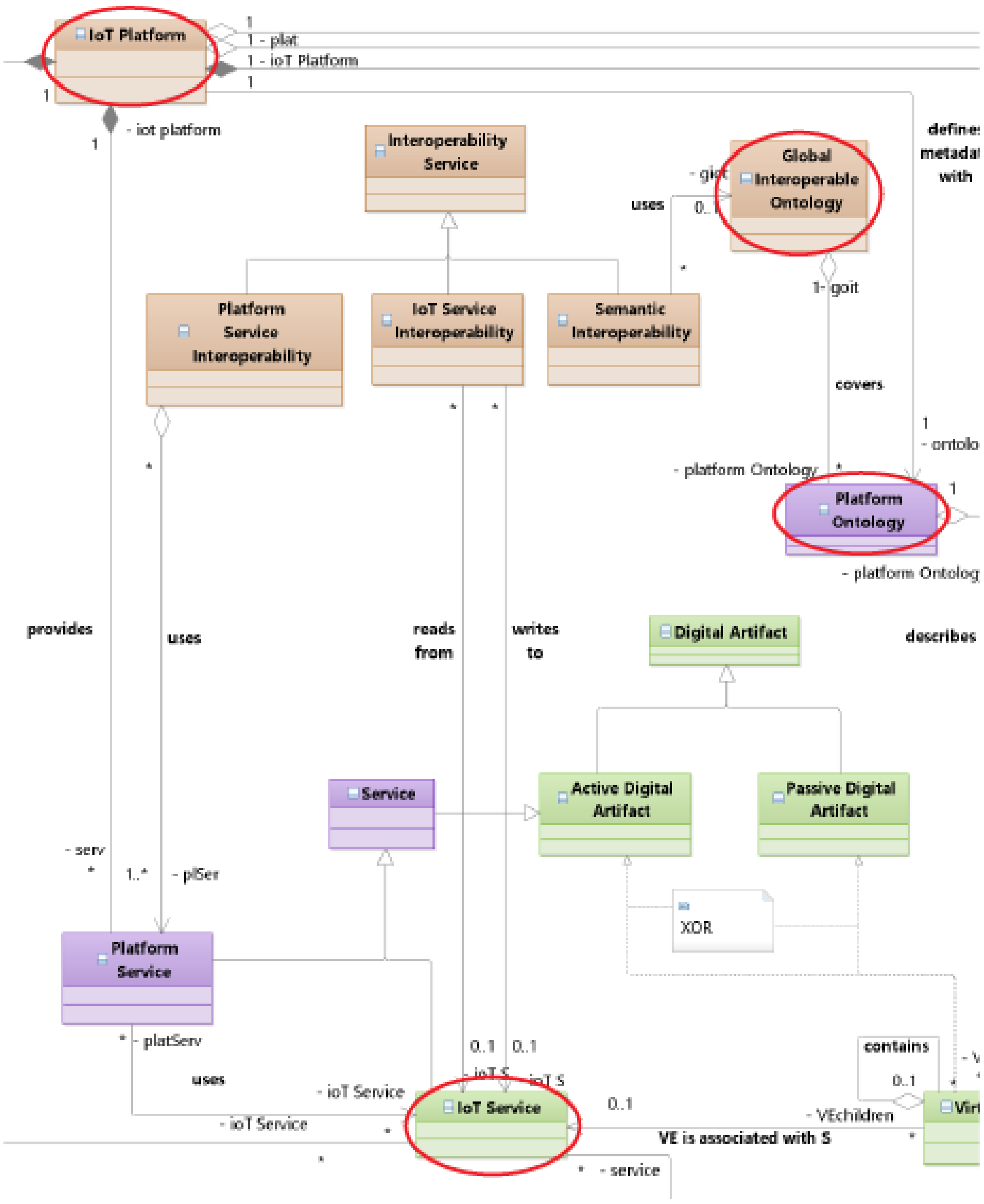

Figure 3.9: Domain Model entities involved in Middleware-to-Middleware communication between IoT Platforms.

3.4.2.4 Application & Services to Application & Services Interactions

The components of the Domain Model involved in the communication at Application and Services level are almost the same as presented in D4.1 with exception of a new entity named Interoperability Service. This entity defined in the previous subsection is extended by Platform Service Interoperability, IoT Service Interoperability and Semantic Interoperability entities, being the super class from where the latter entities are inherited. In the AS2AS interoperability case, attention is focused on the Platform Service Interoperability entity, directly related with the services offered by the IoT platform that we have been analyzed. The entities related with semantics will be used in those cases that Semantic Mediator will be needed to communicate two services.

Example: User creating a Composed Service

In this particular use case, just the Interoperability Service entity that includes Platform Service Interoperability entity as its origin, has been included.

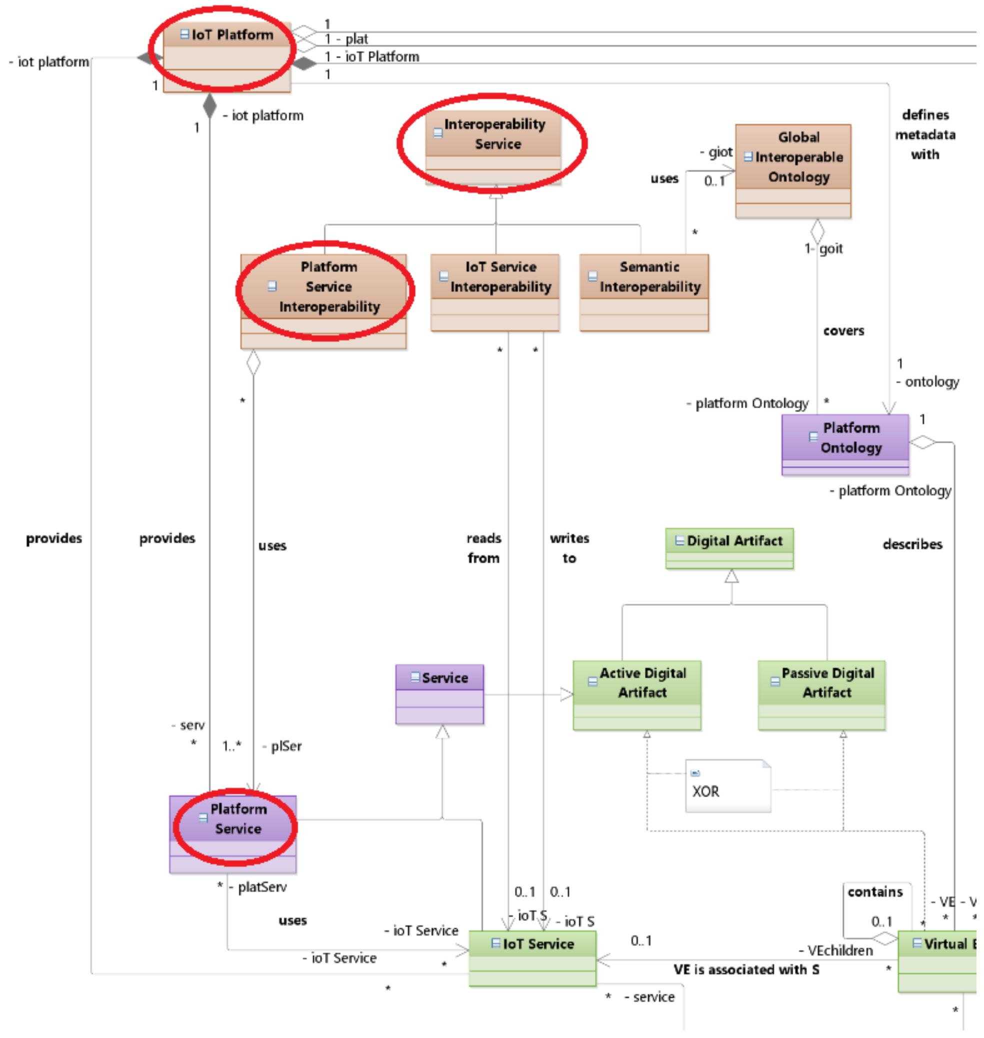

Figure 3.10: Domain Model entities involved in Application and Services-to-Application and Services communication creating a composed service.

Example: Service of an IoT Platform communicates with a service from another IoT Platform
In the second use case it is indicated that, in some cases, it is necessary utilize the Semantic Interoperability entity to communicate two services that are structured with different format. Here, the Interoperability Service entities allow to send messages to any instance of IPSM (publish them to input topics of semantic translation channels) for translation, and receive translated messages from IPSM (consume them from output topics of semantic translation channels). The assumption is that input and output messages have to be in RDF, specifically in JSON-LD message format.

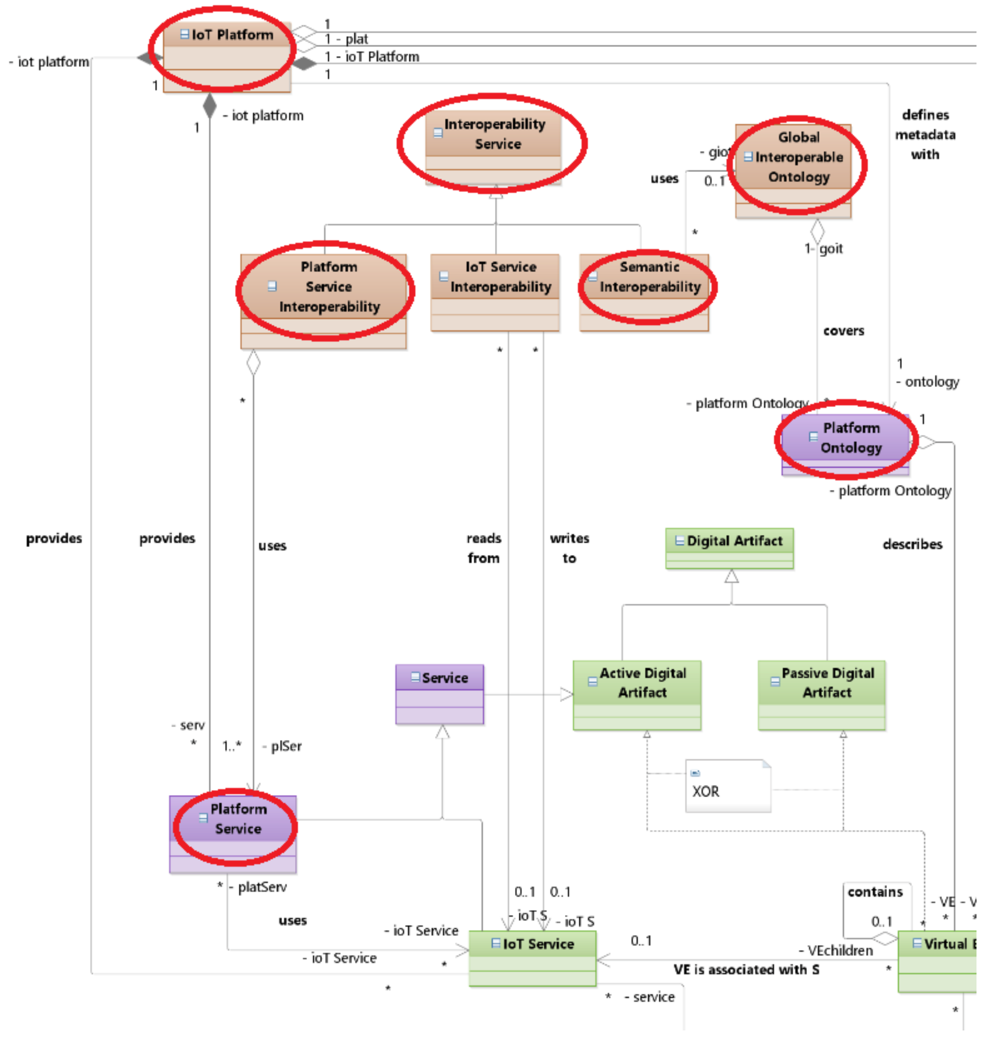

Figure 3.11: Domain Model entities involved in Application and Services-to-Application and Services communication between Platform's services.

3.4.2.5 Data & Semantics to Data & Semantics Interactions

The domain model interactions in DS2DS do not include any new elements, and the relation between elements interacting before (as described in D4.1, Section 3.5.3) did not alter.

3.4.3 INTER-IoT Channel Model for Interoperability

3.4.3.1 Device to Device Interactions

The updated Communication Model defining device to device interactions considers two cases. The first one is shown in Figure 3.12 considering device to device communication through the same physical/virtual gateway (using the rules engine explained in the previous version of this deliverable, D4.1, in section 3.5.4.2). In this case, interoperability is achieved since a sensor and actuator with different network and protocol stacks are able to interact.

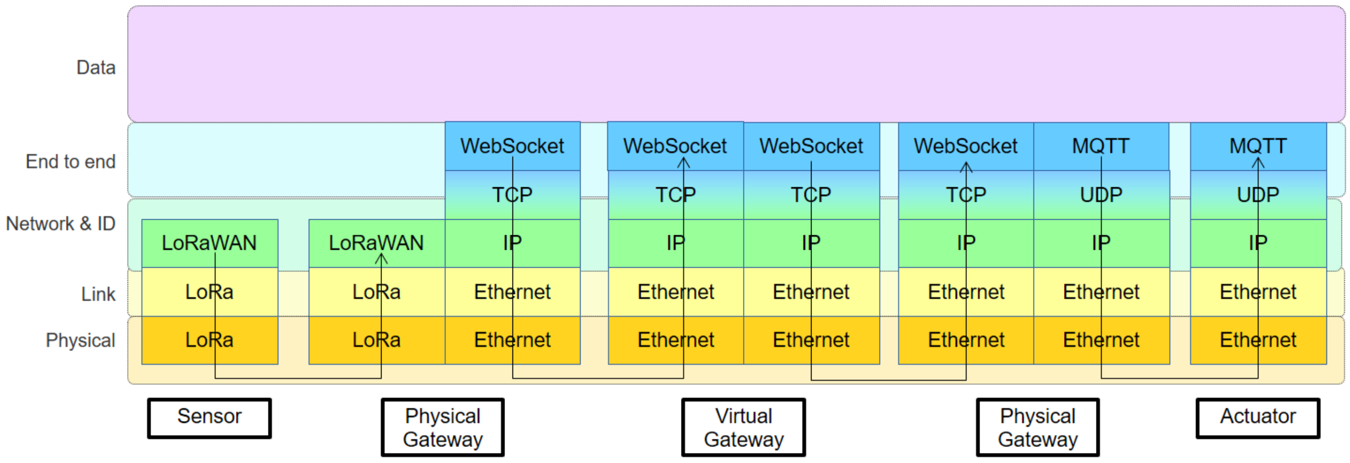

Figure 3.12: Device to Device channel interactions.

The second case is shown in Figure 3.13 and Figure 3.14, it considers device to middleware and middleware to device interactions through the gateway. In this case, device to device that are linked to different gateways is achieved through the middleware platform.

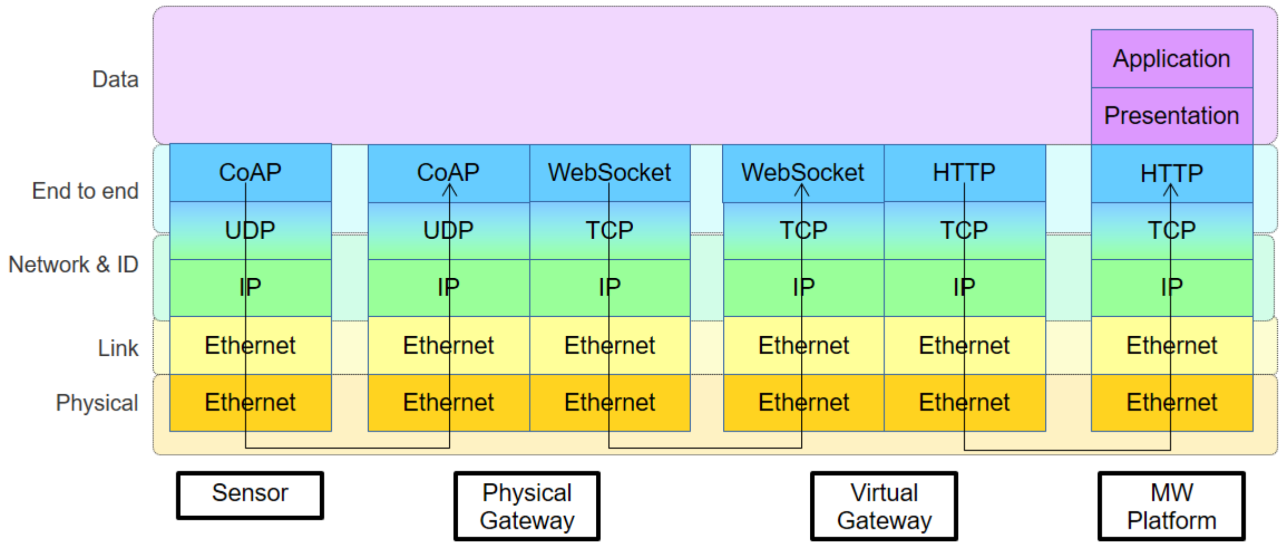

Figure 3.13: Device to Middleware channel interactions.

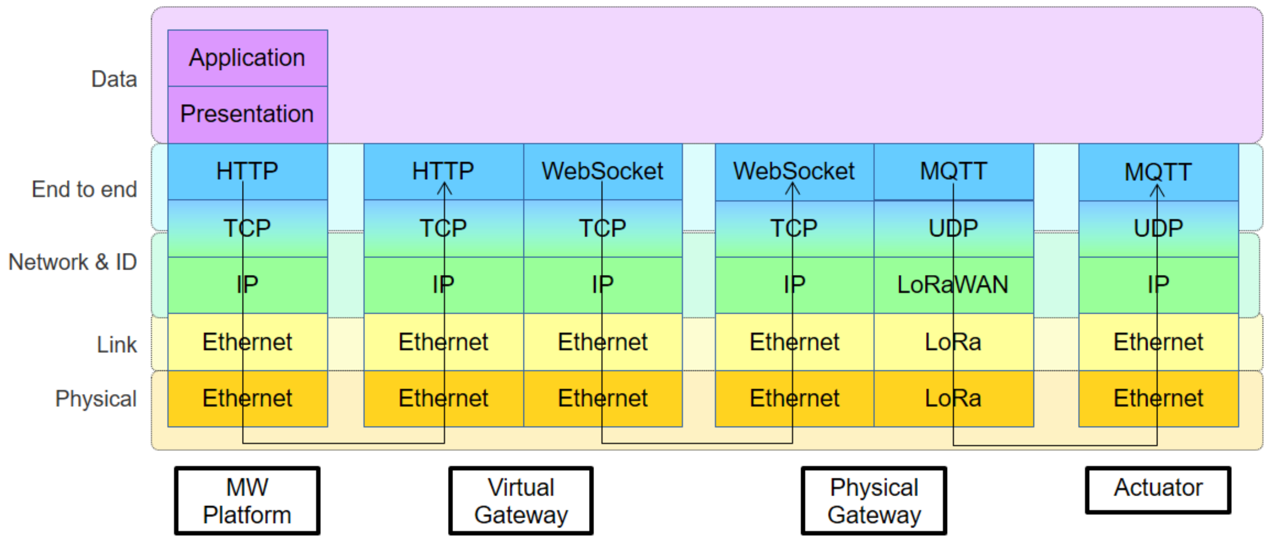

Figure 3.14: Middleware to Device channel interactions.

3.4.3.2 Network to Network Interactions

In the Network layer we also find protocols and communication paradigms that have to be implemented to allow different nodes that compose the network connect with each other and exchange data and control information about the network. In this case, in addition to the data coming from the devices and the platforms, in the network layer we have extra information or control information about the status of the network. Additionally, these extra data is used for configuration purposes.

This information flows from the virtual switches to the controller using specific protocols for network management as is OpenFlow, as it can be observed in 3.16.

In Figure 3.15 it is shown the updated communication interaction between a gateway and a platform through the SDN network. Hence, the information travels through the SDN network, implemented by virtual switches, and in some cases control information also is transported among the network from the switches to the controller.

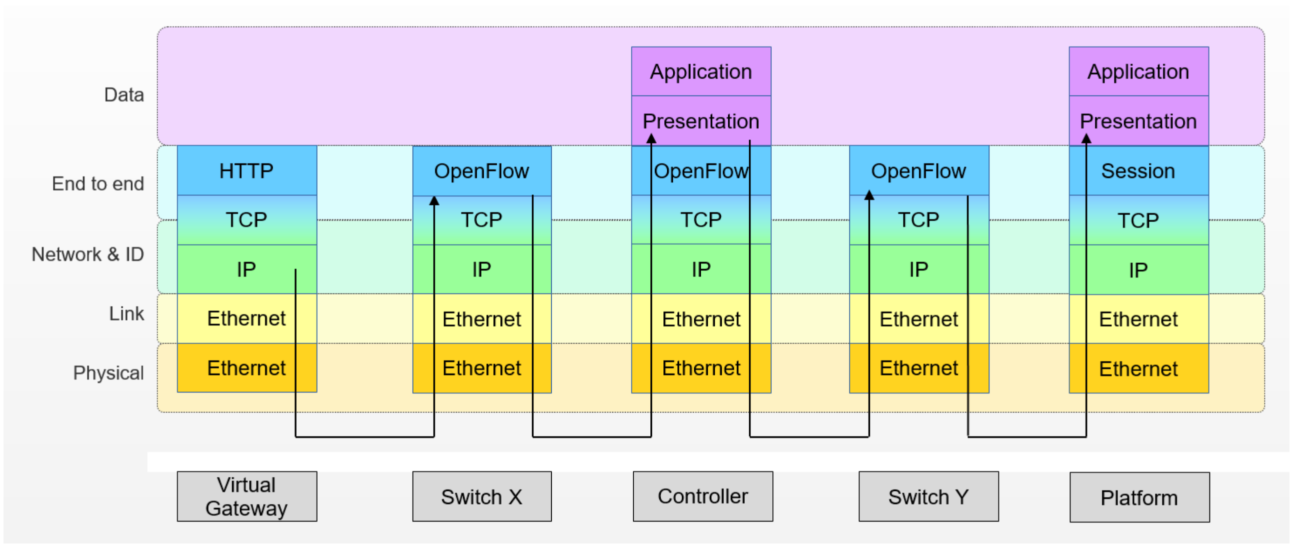

Figure 3.15: Network to network channel interactions

Additionally, an example of communication of the control and management information appears in 3.16 where, using protocols as OpenFlow and OVSDB, the new management rules can be added into the virtual switches. In the example, a switch with no rule associated to a determined packet communicates with the controller to take a decision about the packet. The controller then performs an action, based on the network parameters, to decide the next hop of the packet. Later on, the controller communicates with the switch to insert this new rule within its flow tables.

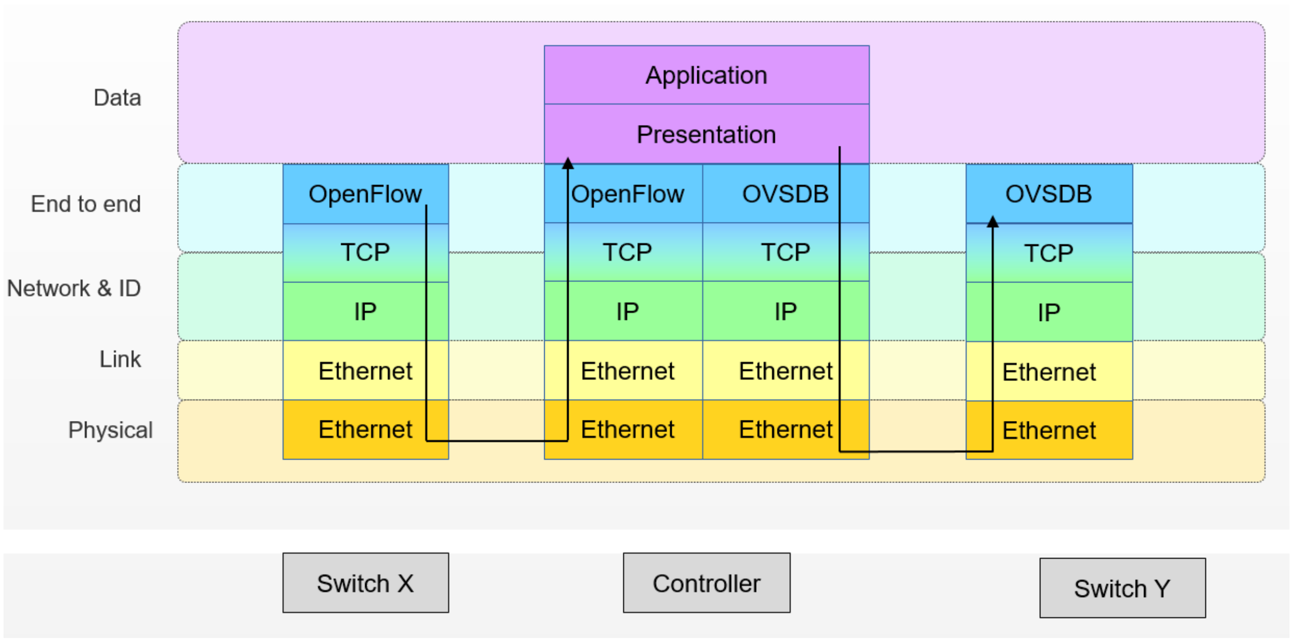

Figure 3.16: Network element to Controller channel interactions

3.4.3.3 Middleware to Middleware Interactions

An example of middleware to middleware communication is indicated in the Figure 3.17. Application of Platform A on the left side of the image (a) wants to send a message through HTTP protocol to another application of another IoT Platform B, which is shown on the right side of the image. The message must first cross The Bridge A (a)-(c) from the Platform A into INTER-IoT middleware. The Bridge A (marked as 1A in the picture) performs syntactic transformation of the message from JSON to JSON-LD, which will be later semantically translated by IPSM. By Rabbit MQ the message is sent through the AMQP protocol to the IPSM Request Manager component (2) (c)-(d) that orchestrates the communication (e)-(f) between Bridges and the IPSM component. The semantic translation of the incoming message (f)-(g) is performed in the IPSM. The message is then sent back to the IPSM Request Manager (g)-(h) which forwards the message to the Services section (3) (i)-(l), where additional processing of message may occur. The (processed) message through the IPSM Request Manager (l)-(m) reaches the IPSM (n)-(o), where again, the semantic translation occurs. Through the IPSM RM (p)-(q)-(r) the message reaches the Bridge B (1B) which is associated with the receiving Platform B (s)-(t). The Bridge B routes the message to the target platform through WiFi instead of Ethernet, demonstrating the technical agnosticism of the INTER-IoT middleware.

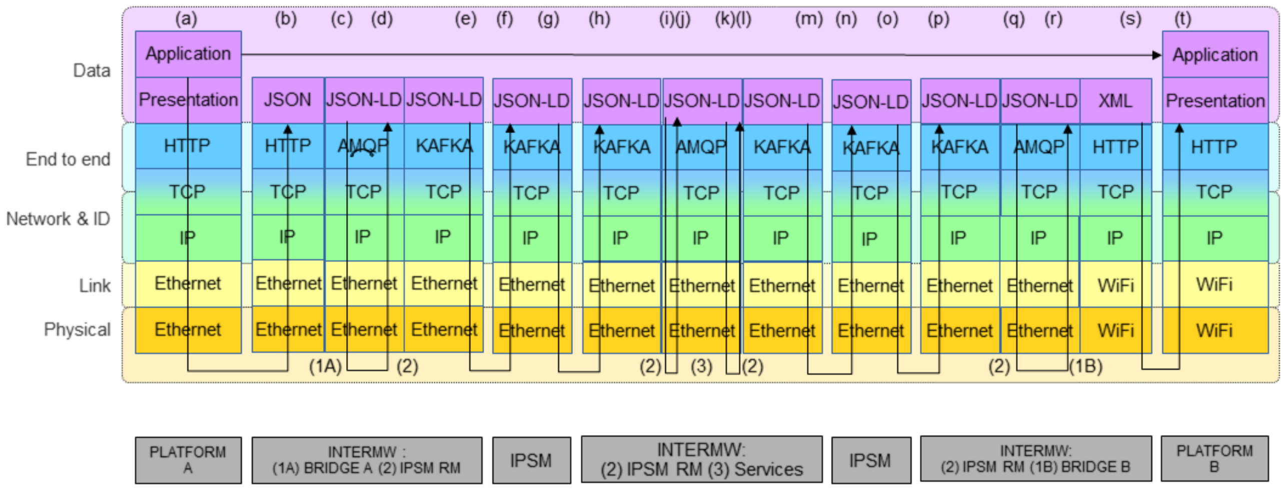
(a) (b) (c) (d) (e) (f) (g) (h) (i)(j) (k)(l) (m) (n) (o) (p) (q) (r) (s) (t)
Data
End to end
Network & ID
Link
Physical
Application
Presentation
JSON
JSON-LD
XML
HTTP
AMQP
KAFKA
TCP
IP
Ethernet
WiFi
(1A)
(2)
(3)
(1B)
PLATFORM A
INTERMW : (1A) BRIDGE A (2) IPSM RM
IPSM
INTERMW: (2) IPSM RM (3) Services
IPSM
INTERMW: (2) IPSM RM (1B) BRIDGE B
PLATFORM B

Figure 3.17: Communication diagram in middleware-to-middleware interoperability.

3.4.3.4 Application & Services to Application & Services Interactions

To define the communication and interaction between elements from the Application and Service layer, Figure 3.18 shows the IoT protocol stacks of the elements that participate in this communication.

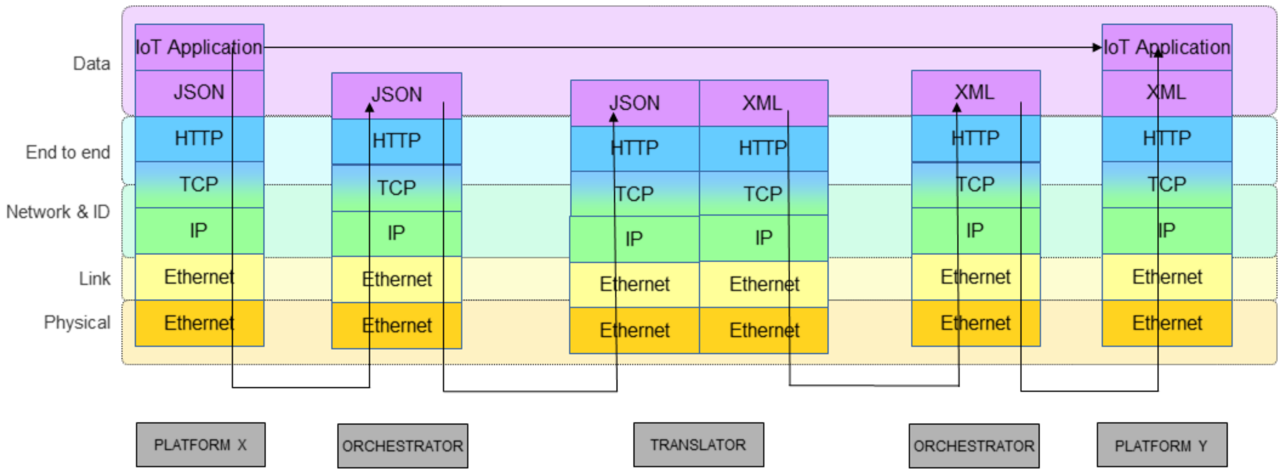

Figure 3.18: AS to AS Element Channel Interactions.

Services from the same or different IoT systems are communicating with the Orchestrator in the INTER-IoT AS2AS solution using HTTP or WebSocket protocol. Hence, the communication between IoT services is performed using these high level communication protocols. To access these IoT services, primarily RESTful Web service is used. Thus, requests made to a resource's URI will elicit a response that may be in XML, HTML, JSON or some other defined format. The response may confirm that some alteration has been made to the stored resource, and it may provide hypertext

links to other related resources or collections of resources. Using HTTP, as is most common, the kind of operations available include those predefined by the HTTP methods – GET, POST, etc.
Another commonly used communication protocol at this level is SOAP that specifies what information is exchanged between web services available in the deployment. It uses XML, with data information related with the service, for its message format, and relies normally in HTTP or, in exceptional cases, in SMTP, for message negotiation and transmission.
So that, as Figure 3.18 shows, the application layer is in charge of translation of communication protocols such as REST and SOAP form one to another and, moreover, the transformation of the message formats, if needed, to achieve the interconnection among these IoT services.

3.4.3.5 Data & Semantics to Data & Semantics Interactions

The communication model and channel interactions in DS2DS did not change since publication of the previous version of the deliverable (see D4.1, Section 3.5.4).

3.5 Security Model

In general, IoT systems are composed of a number of devices, actuators and computing resources that handle a number of physical data and information. Even if this information is public, it is important to guarantee that the information is correct, comes from the correct source, and in certain cases, that it is not readable by non-authorized actors. In the context of INTER-IoT, where different IoT systems and platforms connect, our role is not to be the ”weakest link” of the security chain: this means that INTER-IoT should be able to provide – at the very least – the same security as the systems it connects.
Therefore, while we need to implement a very strong security in our tools, some of them might not be used, as it might be unnecessary (the systems may have a weaker security) or maybe even counter-productive (performance-wise, for instance). For instance, a weaker security key may be used for encrypting a message between two systems, if one of the two pairs does not handle very long keys or, within a specific application, decided that for whatever reason a lighter security scheme would be sufficient.
We will now analyze the main properties, also listed in IoT-A Security Model: Trust, Security, Privacy, and Reliability.

3.5.1 Trust

Trust is very important in IoT systems, as IoT terminals are likely to generate a large quantity of data which will then be used by different services. Trusting the source of data is then of paramount importance, as data sent by malicious sources may compromise entire systems.
IoT-A lists different Trust-models. For INTER-IoT, as it connects different IoT systems, it is important to understand all different models as the systems may use them, in a way or another, and we must be able to carry the “trustiness” between different systems.
Hereafter we have a short list of the most important concepts in this domain:

- **Trust-Model domains:** Complex systems encompass several entities; therefore, a one-size-fits-all model able to be applied to completely different domains, objects, use cases, . . . , is not feasible. Any IoT system should then be split into coherent blocks, each one of them following a specific trust model.

- **Trust-evaluation mechanisms:** These are used in order to define a unique methodology for determining the “trustworthiness degree” of a subject within a specific system. Usually, these systems follow a deductive pattern, building up knowledge on a subject by the information collected, using any means (from direct experience to reputation from other community members).
- **Behavior policies:** They regulate how two subjects within the same Trust Model domain interact according to their trustworthiness value. In other words, a subject may decide, when exchanging information with another subject, to accept or not the value coming from another subject, according to the trust he has towards the subject.
- **Trust anchor:** A subject trusted by default by all subjects using a given Trust Model, and used to compute the trustworthiness of unknown subjects in absence of data.
- **Federation of trust:** It delineates the rules under which trust relationships among systems with different Trust Models can be defined.This is fundamental for INTER-IoT, as it’s necessary to have a federation of trust to provide interoperability between different systems, platforms, and consequently, trust models.

3.5.2 Security

Similarly to IoT-A, the security model has three layers: Service, Communication and Application. However, while IoT-A is focused on an IoT system, INTER-IoT extends the domain to the integration of different systems. Therefore, while in the communication field IoT-A divides the space in two (namely, Unconstrained Devices and Constrained Devices, INTER-IoT adds the platforms communication, which is similar to the Unconstrained node, although it may have different security characteristics. If both can bear the most complete security protocols, an IoT platform can include both constrained and unconstrained nodes, making his overall security more complex.
The domain of constrained nodes is very heterogeneous, with a large number of protocols, most of which focus on small bandwidth and low power use. Bluetooth Low Energy (BLE), for instance, is a typical example of a very insecure communication protocol often used in IoT devices. It is quite easy, and has been demonstrated in numerous occasions [7] [8], that is possible to perform eavesdropping on BLE communication, as well as man-in-the-middle attacks and identity tracking.
As outlined in IoT-A, the role of gateways is fundamental for keeping the constrained domain secure. Gateways, as they are unconstrained device, need to “sandbox” constrained nodes so that, if they are compromised, the effect is just limited to the dropping of the information sent by them, and if possible to deactivate them until the attacker has been found and isolated.
In the implementation of the layers, security is a first-class citizen, specially in the lower interoperability layers. Thus, at gateway level, security in source and destination is supported, via SSL. Device-level security is a responsibility of the manufacturer, since it needs an embedded implementation. To handle this, the INTER-IoT Gateway reports about the security implementation of the connected devices, letting the data consumer rely or not in the data according their security constraints, or even implement further mechanisms.
Moreover, all exposed interfaces in INTER-IoT (not only at device level) are protected with authentication and authorization mechanisms in a centralized Identity Manager. This is specially relevant for the use of INTER-API, which can be accessed publicly.

[7]http://www.cs.tufts.edu/comp/116/archive/fall2015/hosullivan.pdf
[8]https://www.forbes.com/sites/lconstantin/2017/09/12/critical-bluetooth-flaws-put-over-5-billion-devices-at-risk-of-hacking

3.5.3 Privacy

Privacy is a mandatory characteristics of INTER-IoT; as well, one of our use-cases deals with health data, which have clearly a very important privacy requirement.
Following the IoT-A work, we will also use the functional components Authentication component, Trust and Reputation component in order to guarantee the privacy of the data whenever required. These will be explained in detail in the Section 5
The table 3.2 illustrates the way INTER-IoT will handle privacy within these functional components.

Table 3.2: Privacy in functional components

Threat	Result	Mitigation
Identity spoofing	User's identity is spoofed	Robust user authentication procedure preventing man-in-the-middle attacks, with proper credentials-management policy provided by an Authentication component.
	User is involved in transactions with a malicious peer	Trustworthy discovery/resolution/lookup system. Trustworthiness of the entire system is guaranteed through its security components (especially Authentication and Trust and Reputation) as well as its global robustness (security by design).
Information Disclosure	Attacker gains knowledge of user's private parameters	The Identity Management component enforces a robust pseudonymity scheme that ensures anonymity and unlinkability.
	Attacker gains knowledge of user's location	User location can be hidden through reliance on pseudonyms provided by Identity Management.

4 INTER-IoT Metadata Model

As described in D4.1 the INTER-IoT reference meta-data model is proposed in the form of an OWL ontology. The INTER-IoT ontology named GOIoTP (Generic Ontology for IoT Platforms) was engineered to fulfill the requirements put forth by INTER-IoT, but also to offer a core ontology for any artifact in the IoT domain. While reusing existing standardized and established ontologies used in IoT, GOIoTP additionally proposes its own extensions, in order to provide a complete, modular ontology, useful in a wide range of IoT use cases.

Because GOIoTP is a core ontology, it defines some stub classes – i.e. classes that do not have a robust definition, or subclasses. This is a design decision intended to enable different extensions, for those stub concepts, depending on particular needs of an implementation, and follows the usual design patterns for top-level ontologies. Other than offering less restrictions on implementers, stub classes are also easier to align to and from other core IoT ontologies.

In order to offer a more complete interoperability solution, we have also defined an extension to GOIoTP that expands the definitions and subclass structure for the stub classes. GOIoTPex (GOIoTP extended) is the multi-module ontology that extends the reference model of GOIoTP with concrete entities. It completes GOIoTP and is suitable for scenarios, in which INTER-IoT user does not have any preference, when it comes to the data model used internally by INTER-IoT components, i.e. when the user is looking for a complete modeling solution.

The details of GOIoTP and GOIoTPex are presented in what follows. The information in this deliverable is not meant to duplicate the documentation available online[1], as well as in the ontology files themselves (also available online), but rather to reinforce and add to it. What is unique to this document, is that in select places it explicitly connects to D4.1 and explains the INTER-IoT ontologies from perspective of T4.1 and T4.2.

4.1 Reuse of IoT-related ontologies

The most significant change going from the INTER-IoT ontology described in D4.1 to GOIoTP (outside of defining new classes, properties, and instances) is the update concerning SSNX[2] (the legacy version of SSN). SSNX was first chosen as the most significant ontological basis, and was since replaced by the new version – SSN/SOSA[3].

[1] http://docs.inter-iot.eu/ontology/

[2] https://www.w3.org/2005/Incubator/ssn/ssnx/ssn

[3] https://www.w3.org/TR/vocab-ssn/

Table 4.1: IoT ontologies comparison

Category (Subdomain)	SSN	SAREF	oneM2M BO	IoT-Lite†	OpenIoT†
Thing	✓	✓	✓	✓	✓
Device	✓	✓	✓	✓	✓
Device Deployment	✓$^{\alpha}$	✓	✓$^{\oslash\alpha}$	✓	✓
Device Properties & Capabilities	✓				✓
Device Energy	✓	X$^{\epsilon}$			✓
Function & Service	✓^{Fun}	✓	✓	✓^{S}	
Sensing & Sensor properties	✓	✓$^{\beta}$		X$^{\oslash}$	✓
Observation	✓$^{\alpha}$	✓	✓		✓
Actuating & Actuator Properties	✓^{SOSA}	✓$^{\beta}$		✓$^{\oslash}$	
Conditionals	✓				

†	Extends modules of SSNX
SOSA	Only in SSN/SOSA
Fun	Defines functions in DUL alignment
α	No time or location
β	Implicit, implied by device functions
ϵ	Rich energy model
S	Service only
$\oslash$	Only small or provisional description, or a stub

In table 4.1 we present the information about comparison of key IoT ontologies, collected from D4.1 and [5], with a small update for the latest version of SSN/SOSA.

The change (i.e. update) to the newest version of SSN/SOSA had major consequences. The addition of actuation in SSN/SOSA meant that we could remove some ontology imports, such as iot-lite, that were used to provide that functionality before. The iot-lite geolocation and services classes were substituted with our own modules. The Service module was custom built after analysis of existing ontologies, and GeoSPARQL was introduced to the geolocation module. The new observation module of SSN/SOSA fit much better into INTER-IoT meta-data model requirements, so it was fully adopted, with only a minor extension. Detailed information about GOIoTP and GOIoTPex modules can be found in further sections of this document. The difference between SSN/SOSA and the old version of SSN, called SSNX, is well documented online[4].

To sum up the introduced changes resulted in a great simplification of import structure for GOIoTP, and it currently only imports (directly) SSN/SOSA, and (indirectly) its imports (e.g. the time ontology[5]). It also uses entities from GeoSPARQL[6] and SWEET Units[7] ontologies. Required modules, that were previously offered by removed imports, were substituted with our own custom modules.

[4]https://www.w3.org/TR/vocab-ssn/#&Developments
[5]https://www.w3.org/TR/owl-time/
[6]http://www.opengis.net/ont/geosparql
[7]http://sweet.jpl.nasa.gov/2.3/reprSciUnits.owl

4.2 Meta-data model scope

Definition of the scope of the Inter-IoT reference meta-data model is the output of the process defined in D4.1, section 2.4. The scope itself was discussed in D4.1, section 3.3.2, following the process of information gathering from sources, such as INTER-IoT grant agreement and requirements, IoT platform market analysis, and existing ontological standards for IoT. Here, we present only a short summary. For more details, we refer the readers to D4.1.

A number of general meta-data terms and categories were identified in D4.1. The categories included *Device*, *Middleware*, *Service*, *User*, *Application*, *Network*, *Data* and *Provenance*. Each category contained multiple terms, as detailed in D4.1. Since then the categories and their terms were mapped into ontological entities, and grouped into ontological modules. The modular structure of GOIoTP and GOIoTPex is presented in the following section.

4.3 Generic Ontology for IoT Platforms

Generic Ontology for IoT Platforms (GOIoTP) is at the same time the INTER-IoT reference meta-data model and a core ontology for IoT. It is extended by GOIoTPex that augments and “annexes” GOIoTP with concrete entities. The two ontologies together offer a meta-data model for any IoT application, with GOIoTP being the extensible core ontology, and GOIoTPex offering useful extensions, while still preserving a level of generality. Both ontologies are designed to be modular and, although it is formally correct to support, for instance, GOIoTP without the location module, we recommend that the ontology is treated as a whole, and to view the modules as conceptual division, rather than a formal one. The GOIoTPex data model is implemented in INTER-MW, and also, to some extent, used by other INTER-IoT components.
In principle, the entities defined in GOIoTPex extend and instantiate those defined in GOIoTP. For instance, while GOIoTP defines a class for data formats, namely *iiot:DataFormat*, it does not define any specific formats. GOIoTP extends this concept by defining both subclasses like *iiotex:JSON*, and instances of those subclasses, e.g. *iiotex:NGSIv2JSON*. In this way the general classes of GOIoTP can be used in a wider area, that the more specific GOIoTP entities. On the other hand, GOIoTPex offers convenient collection of classes, instances and properties for general use, and as a reference of extension of GOIoTP.
Note that because GOIoTPex extends GOIoTP, the compliance, inclusion or implementation of GOIoTPex necessarily includes GOIoTP. In other words, in this text we assume that using GOIoTPex implies also using GOIoTP. In this way, GOIoTP is a vertical module of GOIoTPex.
In the following sections the contents of INTER-IoT ontologies is described, and relation between GOIoTP and GOIoTPex clarified. For brevity, full URIs are not used in the text, or in the images in this section. The prefixes for all used ontologies can be found in section 4.3.1.10. The usage of GOIoTP and GOIoTPex in INTER-IoT is described in section 2 of this document, and in D3.2.

4.3.1 Ontology structure

The modular structure of GOIoTP is mirrored in GOIoTPex. The defined modules are as follows:

- **Device** – (optionally virtualized) hardware connected to IoT
- **Platform** – software platforms that host IoT devices

- **Observation** (and actuation) – information gathered by IoT entities from the world, or the intention to act upon it
- **Units & measurements** – systems of units and values
- **User** – human or software that uses, or is a client of IoT entities
- **Geolocation** – position within the world, defined by coordinates
- **Service** – web services offered by software
- **Provenance** – origin, ownership and history of IoT entities

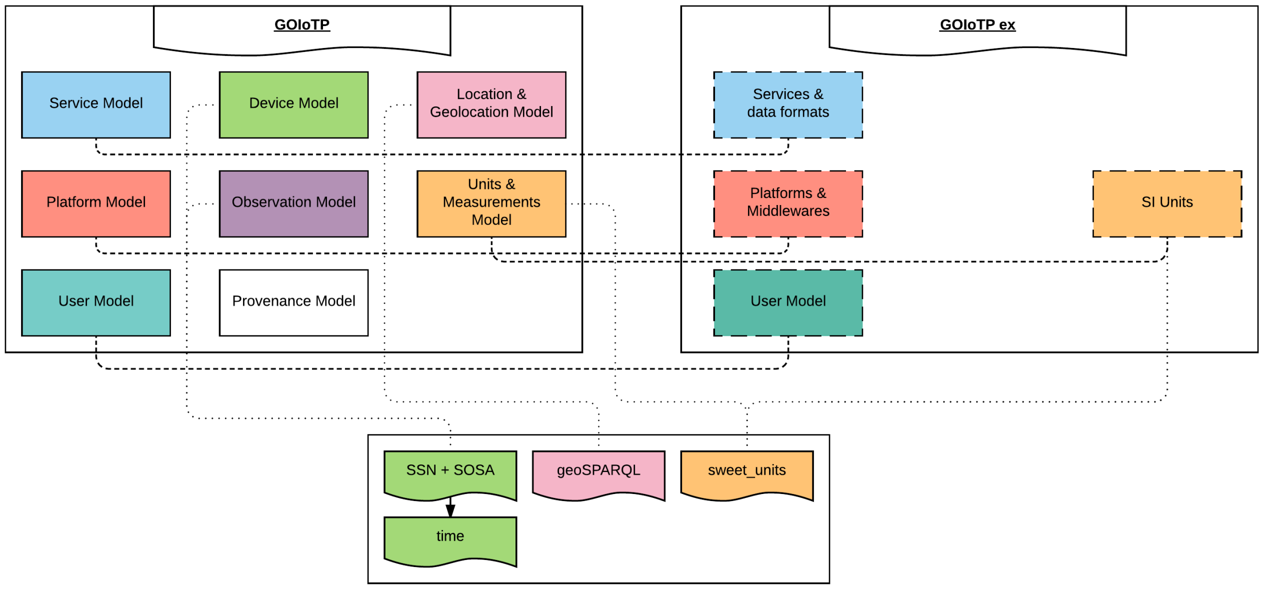

Figure 4.1: An outline of GOIoTP and GOIoTP modules.

The overview of GOIoTP modular structure is available in Figure 4.1. GOIoTPex extends only select modules of its base ontology. In what follows, each module has a subsection with a description and a diagram. The key for the module diagrams is in Figure 4.2. Note that the diagrams depict only the elements that are most important for a given module. Refer to the ontology files themselves for a full information. In particular, GOIoTP defines a couple of properties that are not specific to any module, such as *iiot:hasName* and *iiot:hasDescription* meant to contain human-readable text, with which any GOIoTP entity can be annotated. To improve clarity, such properties, including some SSN/SOSA properties, are not included in the diagrams, unless they have a particular importance for a given module.

4.3.1.1 Imported ontologies

The only direct imports of GOIoTP are SSN and SOSA. Additionally, GOIoTPex imports GOIoTP. Entities from other ontologies are also used, but, because those are cherry-picked, we do not import the whole ontologies. This is also done to avoid long import chains that would slow down any ontology explorer, editor, or triplestore that loads all ontology imports in a chain. Those properties are

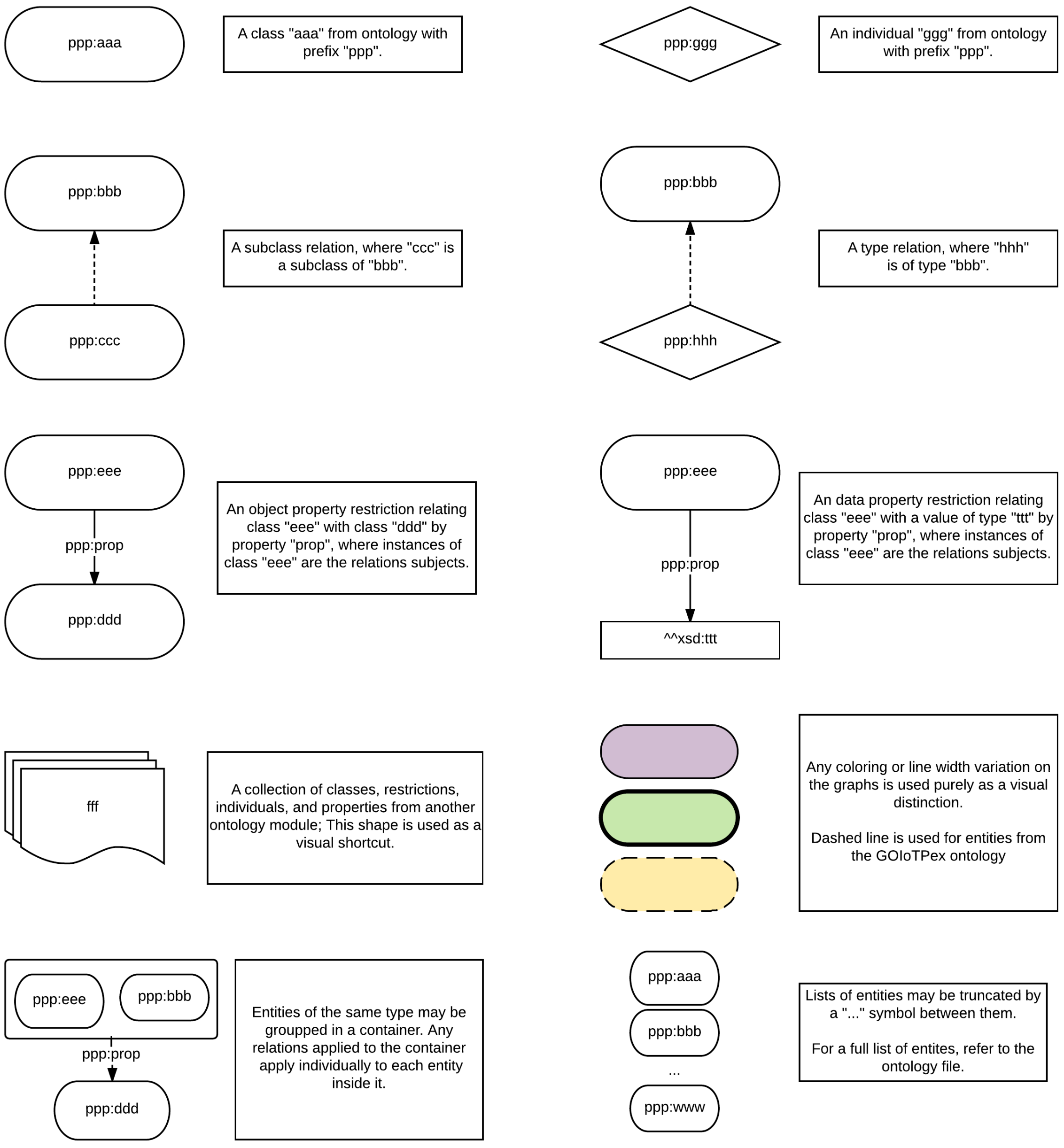
ppp:aaa
A class "aaa" from ontology with prefix "ppp".
ppp:ggg
An individual "ggg" from ontology with prefix "ppp".
ppp:bbb
ppp:ccc
A subclass relation, where "ccc" is a subclass of "bbb".
ppp:bbb
ppp:hhh
A type relation, where "hhh" is of type "bbb".
ppp:eee
ppp:prop
ppp:ddd
An object property restriction relating class "eee" with class "ddd" by property "prop", where instances of class "eee" are the relations subjects.
ppp:eee
ppp:prop
^^xsd:ttt
An data property restriction relating class "eee" with a value of type "ttt" by property "prop", where instances of class "eee" are the relations subjects.
fff
A collection of classes, restrictions, individuals, and properties from another ontology module; This shape is used as a visual shortcut.
Any coloring or line width variation on the graphs is used purely as a visual distinction.
Dashed line is used for entities from the GOIoTPex ontology
ppp:eee
ppp:bbb
ppp:prop
ppp:ddd
Entities of the same type may be groupped in a container. Any relations applied to the container apply individually to each entity inside it.
ppp:aaa
ppp:bbb
...
ppp:www
Lists of entities may be truncated by a "..." symbol between them.
For a full list of entites, refer to the ontology file.

Figure 4.2: GOIoTP module diagrams key.

specifically from GeoSPARQL[8] and SWEET Units[9] (Semantic Web for Earth and Environment Technology[10]) ontologies, and other ontologies or namespaces accompanying those two (for instance geographical functions[11], or mathematical relations[12]). Entities used in similar fashion by SSN/SOSA come from the Time[13] ontology.

4.3.1.2 Device module

The device module (Figure 4.3) can be considered to be the core part of GOIoTP and, although other ontology modules can be used independently, this one was the starting point of this ontologies design. It models the core part of IoT, which is the device. It is centered around 4 classes – *iiot:IoTDevice*, *sosa:Sensor*, *sosa:Actuator* and *sosa:Sampler*.

The most significant, from the modeling perspective, is the *iiot:IoTDevice* class. As a subclass of *sosa:Platform* it hosts entities of type *sosa:System*. It is a class that represents a collection that includes any kind of smart device (physical or virtual) connected to IoT, for instance mobile phones, tablets, wireless keyboards, light fixtures, traffic gates, parking meters, thermometers, smart medical equipment, and so on.

The most informative classes are *sosa:Sensor* and *sosa:Actuator*, which signify the purpose of a *sosa:System* in an IoT ecosystem. Both are subclasses of *sosa:System*, and can therefore be hosted on a *iiot:IoTDevice* (or any other *sosa:Platform*). Any *sosa:System* can have subsystems. Informally speaking, sensors and actuators *usually* serve a smaller purpose than a whole IoT device – e.g. detection of change in a single observable property (e.g. air temperature).

Finally, GOIoTP does not place any special importance on *sosa:Sampler*, which is a new introduction to SSN since SSNX. It is, as a principle, supported by GOIoTP and INTER-IoT, but we have not found any specific use case during our work that would include it.

The device model of GOIoTP offers two ways to think of IoT devices. First one, closer to the original vision of SSNX, is that an *iiot:IoTDevice* does not offer any sensing or actuating capabilities itself, instead being a platform that groups together (i.e. hosts) systems of sensors and actuators. Those systems offer sensing, or actuating, and are smaller and more granular than the whole device, and may have subsystems of their own. The second way to look at an IoT device, is to treat an IoT device as a single-purpose machine, i.e. an instance of both *iiot:IoTDevice* and a *sosa:Sensor* (or *sosa:Actuator*). This approach is suitable for specialized devices, such as a single purpose sensor measuring air quality. Because such sensor does not offer any other capability, we treat the whole device, as if it was a sensor itself.

To summarize, an instance of an *iiot:IoTDevice* can be a host for multiple systems, but it can also be a single purpose *sosa:Sensor* or *sosa:Actuator* itself. We have found that in our initial implementations of INTER-IoT software, the latter case appeared much more often. Formally, not every *sosa:Sensor* or *sosa:Actuator* is an *iiot:IoTDevice*. An *iiot:IoTDevice usually* either is a *sosa:Sensor* (or *sosa:Actuator*), or hosts at least one *sosa:Sensor* or *sosa:Actuator*.

Technically, an instance of *iiot:IoTDevice* and a *sosa:Sensor* is at the same time a *ssn:System* and a *sosa:Platform* (which is inferred from SSN/SOSA taxonomy). This is a callback to the device model in SSNX, where *oldssn:Device* was a subclass of *ssnx:System* and *ssnx:Platform*. Unfortunately, a

[8]http://www.opengis.net/ont/geosparql
[9]http://sweet.jpl.nasa.gov/2.3/reprSciUnits.owl
[10]https://sweet.jpl.nasa.gov/
[11]http://www.opengis.net/def/function/geosparql/
[12]http://sweet.jpl.nasa.gov/2.3/relaMath.owl
[13]http://www.w3.org/2006/time

device class, as such, does not exist in SSN/SOSA, and introduction of *iiot:IoTDevice* is a rectification of that fact.

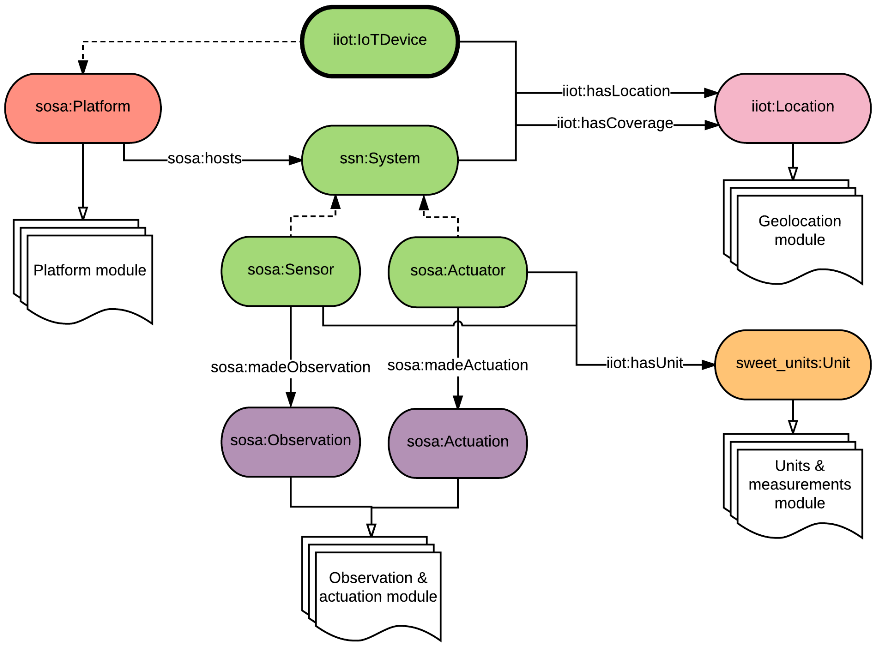

Figure 4.3: GOIoTP device module summary diagram.

4.3.1.3 Platform module

The platform module (Figure 4.4) extends the SOSA *sosa:Platform* description and defines a new class of *iiot:SoftwarePlatform*. Platforms in GOIoTP can have explicitly defined semantics, asserted via the *iiot:hasSemantics* property with the value being an instance of *iiot:Semantics*, or its subclass *iiot:Ontology*. This property is useful to INTER-IoT and other interoperability systems that interoperate (on the semantic level) artifacts with different semantics.
A *iiot:SoftwarePlatform* has components (either instances of *iiot:PlatformComponent*, or its subclass *iiot:Middleware*), akin to subsystems of *ssn:system*. GOIoTPex defines a collection of subclasses of *iiot:Middleware* that represent software used in INTER-IoT demo and pilot implementations, e.g. *iiotex:FIWARE*, *iiotex:WSO2*, or *iiotex:UniversAAL*.
The connection of the platofrm module and the device module is realized through the fact that *iiot:IoTDevice* is a subclass of *sosa:Platform*, and platform components are subclasses of *ssn:System*.

4.3.1.4 Observation and actuation module

Observation and actuation module (Figure 4.5) defines data structures that represent information which is either gathered by a sensor, or produced by an actuator, with the intention to act on the world. Instances of both *sosa:Actuation* and *sosa:Observation* can be connected with an instance of a *sosa:Result*, which is the entity that holds the information most relevant to sensing or actuation. GOIoTP extends the SSN/SOSA model of results by defining the *iiot:hasResultValue* property that holds a literal value of the observation or actuation.

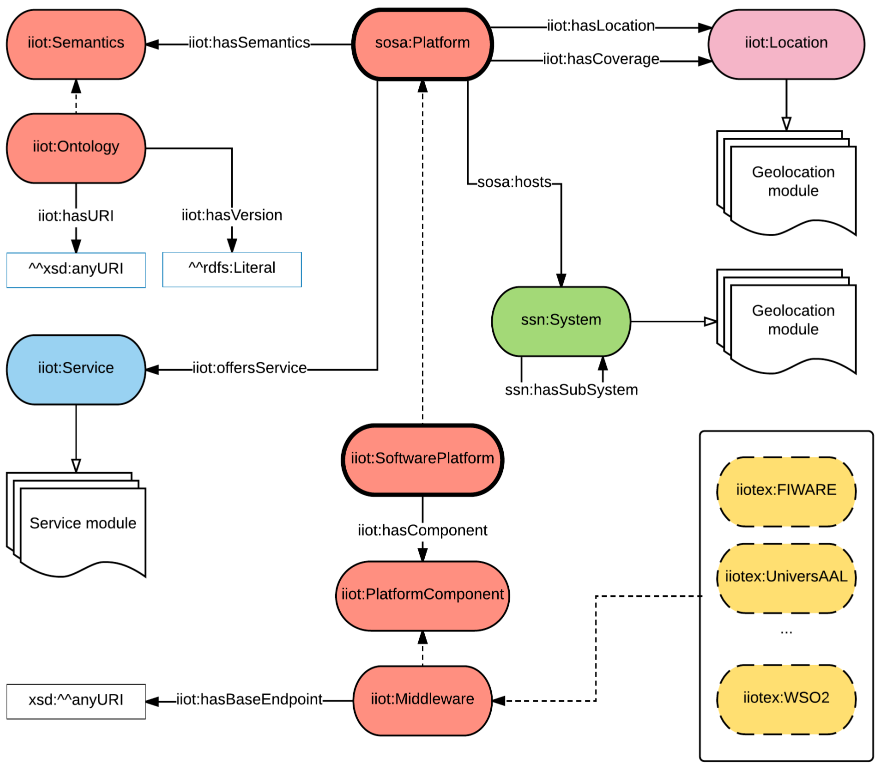

Figure 4.4: GOIoTP platform module summary diagram.

In GOIoTP a single observation (or actuation) may have more than one result. This is done to account for the fact that some results of observations may be connected and may make sense only in the context of each other. For instance a blood pressure measurement that produces information about systolic and diastolic blood pressure of a patient would be modeled in GOIoTP as a single observation with multiple results.
The SSN/SOSA model of observations, actuations and results is itself quite verbose, as it includes information about time, observed (or acted upon) properties and features of interest, and is well connected to the device module of SSN/SOSA. Because of that the observation and actuation module of GOIoTP does not introduce much new information or changes into it. Note, however, that other GOIoTP modules i.e. location, user, and units offer additional properties that enhance observations, actuations and results (see further subsections in this chapter for details). As stated before, in our opinion GOIoTP modules work best when all of them are utilized, and the ontology itself should be used as a whole, as opposed to selecting single modules.
The observation and actuation module is attached to the device module by the *sosa:madeActuation* and *sosa:madeObservation* for *sosa:Actuator*s and *sosa:Sensor*s respectively.

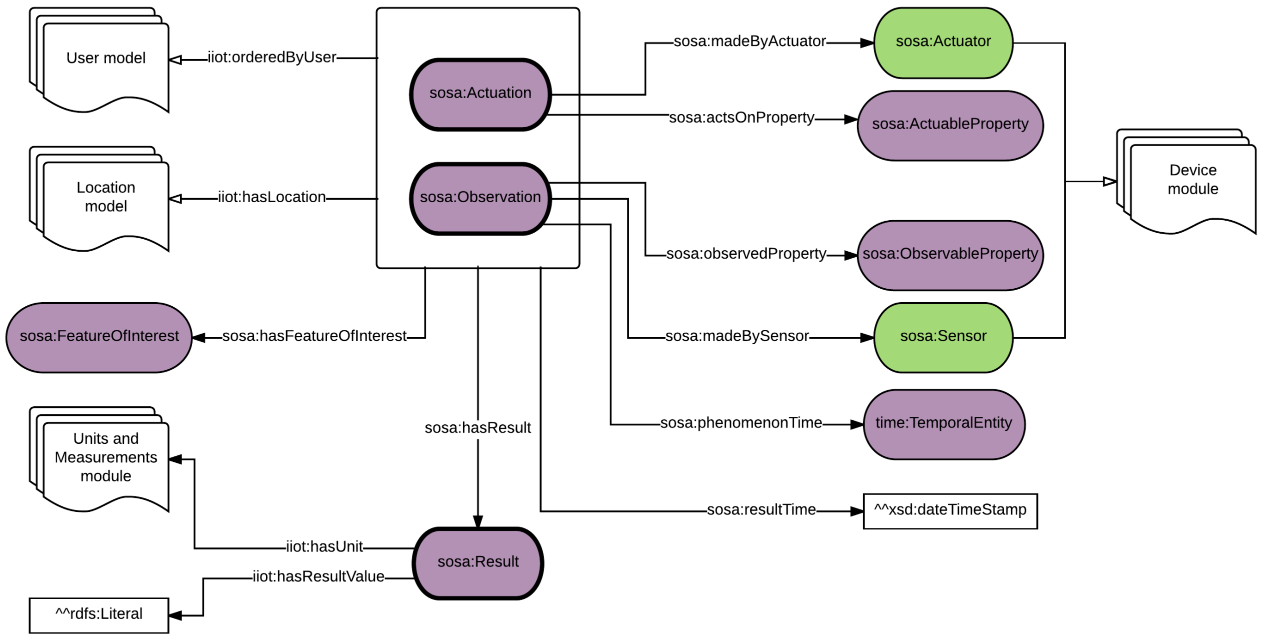

Figure 4.5: GOIoTP observation & actuation module summary diagram.

4.3.1.5 Units and measurements module

The units and measurements module, as shown in Figure 4.6, in GOIoTP makes use of the class of *sweet_units:Unit* to attach information about unit of measurement to instances of *sosa:Result* (from the observations module). This is a very simple, but also much needed information that puts a valuable context on the value of any *sosa:Result*. The class of *iiot:MeasurementKind* subclasses *ssn:Property* and its instances are intended to store information about the kind, or type, of measurement or unit (e.g. mass, length, etc).
The generic model of GOIoTP allows for any system of units to be used under *sweet_units:Unit*

and *iiot:MeasurementKind*. GOIoTPex implements this idea by utilizing a full suite of units and measurement kinds from both the SI system in general, and the SI units defined in SWEET Units ontology. Subclasses of *iiot:MeasurementKind* are defined, such as *iiotex:Volume*, *iiotex:Mass*, *iiotex:Luminance* and many others; and organized under a basic SI taxonomy (e.g. *iiotex:SIBaseMK* for base SI units, *iiotex:SIDerivedMK* for derived units and so on). Instances that represent concrete SI units, such as *sweet_units:kilogram*, or *sweet_units:ampere* are also included in GOIoTPex under the taxonomy defined in SWEET Units ontology.

4.3.1.6 User module

The user module (Figure 4.7) is centered around the *iiot:User* class. This class is a subclass of the *sosa:FeatureOfInterest* class. It describes human or software that uses, or is a client in the context of an IoT system.
The user module connects to the device, observation, services and platform modules via *iiot:hasUser* and *iiot:orderedByUser* properties. The *iiot:hasUser* is a generic connection between a user and an entity. When a device has a user, it means that its capabilities or functionalities are being used by a particular user. This property is very general in meaning and GOIoTP does not put many constraints on the way it is meant to be used. The *iiot:orderedByUser* property, on the other hand, describes that an observation or actuation was ordered to be done by a user. It can be used, for example, to inform that a particular doctor ordered a measurement of patients blood pressure, or that a software agent decided that an actuation needs to be performed by a smart gate sensor, in order to close the gate.
Because of the connection to SSN/SOSA *iiot:User* can be used like any other *sosa:FeatureOfInterest* to connect users to observations, or actuations (e.g. to say that an observation was made about a users *ssn:Property*).
GOIoTPex uses a subclass of *ssn:Property*, namely *iiotex:AuthenticationData*, to extend the user module. Instances of this class are meant to store information known to the user and required to access some system or platform, for instance, a password and login pair. GOIoTPex does not place any restrictions on *iiotex:AuthenticationData* (e.g. passwords can be plain-text, encrypted, or not required/present at all), so it needs to be extended before usage.
Note that although *iiot:User*s can be human, GOIoTP does not define properties that would usually be attached to a human user, such as first and last name, age, nationality, address of residence, and so on. If such properties are required, we recommend using well-known and standardized vocabularies, such as schema.org[14].

4.3.1.7 Geolocation module

The geolocation module (Figure 4.8) proposes a model centered around the class of *iiot:Location*, which describes any physical location, such as a building, a specific room, or a city. The geographical coordinates are attached to a *iiot:Location* with the *geosparql:asWKT* property, whose value is a *geosparql:wtkLiteral* (WKT stands for “well known text”). WKT is a string format that allows declaration of geographical points (e.g. “POINT[31.2543139, -24.2584805]”), lines and areas bounded by many different shapes. Although WKT also allows for textual descriptions, like “AREA["Netherlands offshore."]”, they are optional, and we recommend to put any textual information relevant to ontological entities in annotation properties, as opposed to the value of the *geosparql:asWKT* property. WKT also allows for definition of (optional) “vertical extent” i.e. a height range, and even a temporal extent i.e. a time range over which the geoposition is relevant. Because of how complicated a WKT

[14] http://schema.org

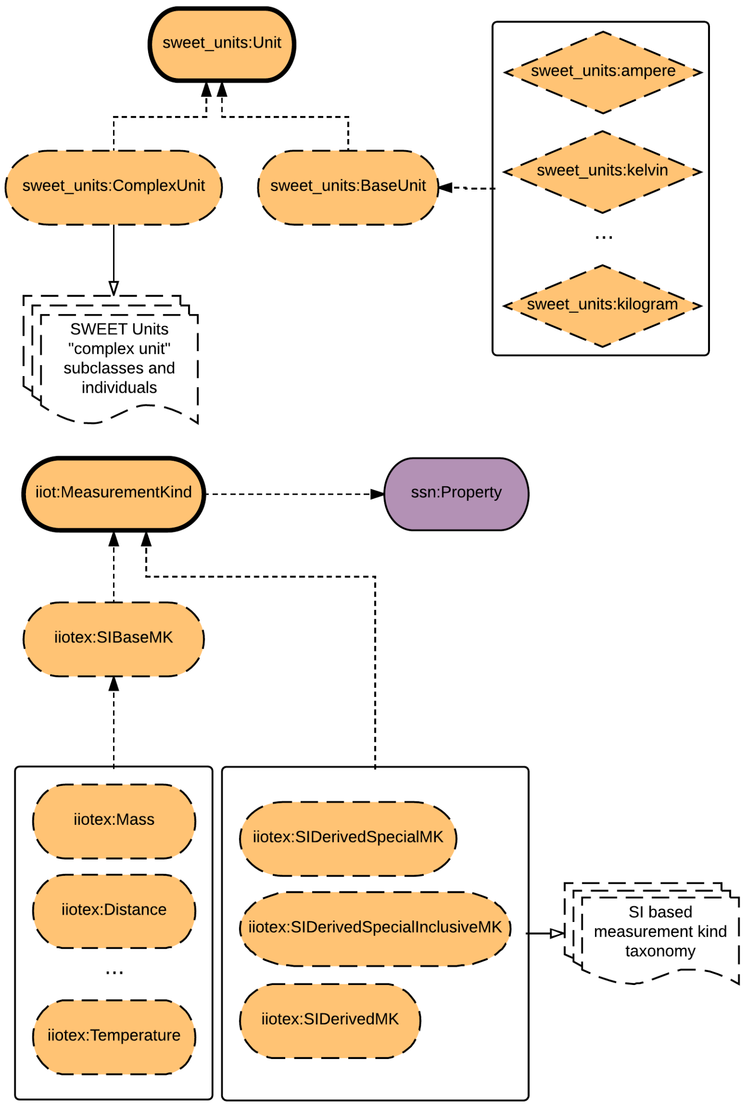

Figure 4.6: GOIoTP units & measurements module summary diagram.

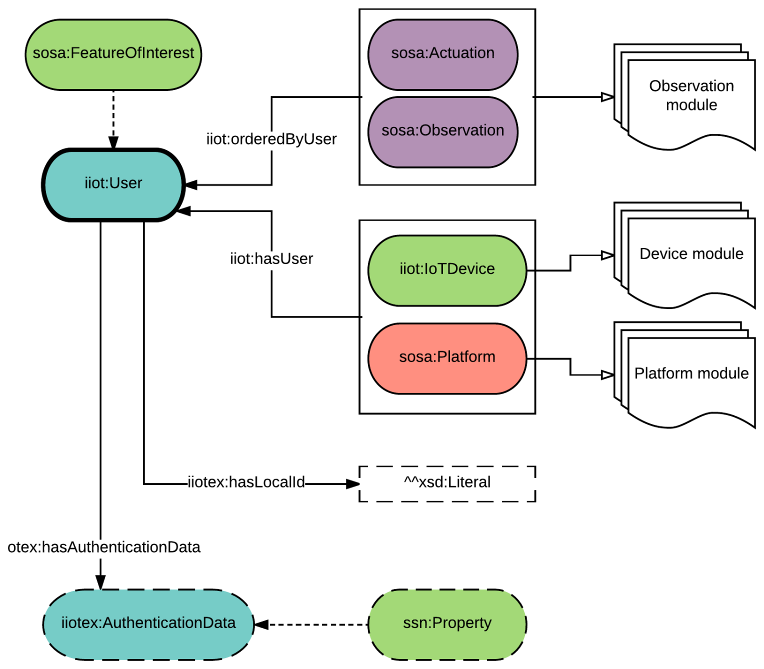

Figure 4.7: GOIoTP user module summary diagram.

description can become, we advise that the way in which it is actually used is explicitly documented, when using GOIoTP. For instance, if a software system only parses geopositional information, without any additional information (i.e. the optional textual, temporal and vertical descriptions), it should be made clear in the documentation. More information about GeoSPARQL[15] and WKT[16] is available in their respective online documentations.
GOIoTP does not define any other location-specific properties, such as address, or country code information. The geolocation module connects to device, observation and platform modules through *iiot:hasLocation* and *iiot:hasCoverage* properties. These properties are applicable to *sosa:Result*, *ssn:System* and *sosa:Platform*. The first property informs about a physical location of an entity, while the latter defines a point, or an area inside which the functionalities or capabilities of an entity are offered. For instance, a temperature sensor that covers a whole room (*iiot:hasCoverage*) can be placed in one specific corner of that room (*iiot:hasLocation*). Note that the value of both properties is a *iiot:Location*, so the *iiot:hasLocation* property value is not limited to a description of a point, and may be use any allowed *geosparql:asWKT* value, including an area. In such cases the property informs about an approximate location (e.g. an entity in a building), as opposed to a precise point.

4.3.1.8 Service module

The service module (Figure 4.9) offers a model for services offered by platforms, or smart devices. An instance of a *iiot:Service* (which can be offered by any *sosa:Platform*, including *iiot:IoTDevice* and *iiot:SoftwarePlatform*) has an interface (i.e. *iiot:ServiceInterface*). The interface, is a description of a way of accessing the service, with a formal definition via *iiot:hasServiceDefinition* (e.g. a WSDL file),

[15]http://www.opengeospatial.org/standards/geosparql
[16]http://www.opengeospatial.org/standards/wkt-crs

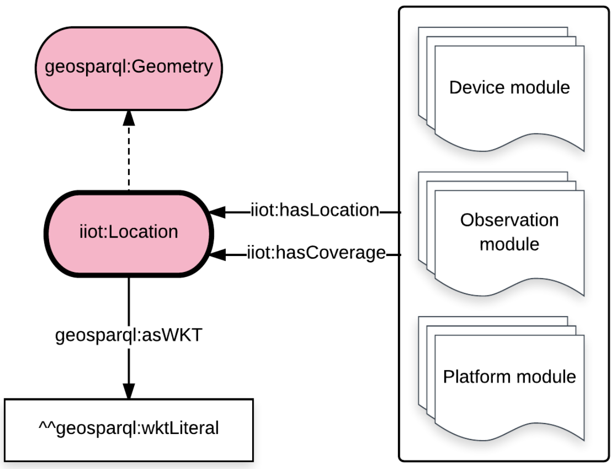

Figure 4.8: GOIoTP geolocation module summary diagram.

an input (*iiot:ServiceInput*) with an URL endpoint (at which the service is accessed), and an output (*iiot:ServiceOutput*). Both *iiot:ServiceInput* and *iiot:serviceInput* can specify a *iiot:DataFormat* – for instance JSON.
GOIoTPex defines subclasses and instances for the service module. It offers specific subclasses of *iiot:Service*, such as *iiotex:CEPWebService* for complex event processing webservices; subclasses of *iiot:ServiceInterface* such as *iiotex:REST*, and subclasses of *iiot:DataFormat*, e.g. *iiot:JSON*. Very specific data formats, such as FIWARE NGSIv2 JSON are included in GOIoTPex as instances (*iiotex:NGSIv2JSON*).
The service module connects to the platform module through the capability of platforms to offer services via the *iiot:offersService* property.

4.3.1.9 Provenance module

GOIoTP does not define its own dedicated provenance classes or properties, instead reusing select few from existing ontologies. *Dublin Core*[17] properties are used to annotate the ontology itself (e.g. *dc:creator*). We recommend Dublin Core for annotating the basic provenance information. If a more complex provenance description is needed, we recommend the PROV-O[18] ontology. It defines a very robust provenance model, and has an official alignment to SSN/SOSA available at `https://www.w3.org/TR/vocab-ssn/#PROV_Alignment`, so it can be easily used with GOIoTP. In our experience, the PROV-O model offers a model that is simply too verbose for a general IoT use-case, so we did not explicitly include it in our ontology.
GOIoTPex defines only one provenance-like property *iiotex:hasLocalId*. Applicable to any *iiot:User* or *iiot:IoTDevice* this property stores information about identifiers that are valid within some local system, but not necessarily globally unique. This property is of particular significance to INTER-IoT, and possibly other interoperability systems. INTER-IoT on the middleware level, and higher levels of interoperability, manages entities (devices in particular) that come from different systems (middlewares), and already have an identity within those systems. In order to preserve uniqueness

[17] `http://dublincore.org/documents/dces/`
[18] `https://www.w3.org/TR/prov-o/`

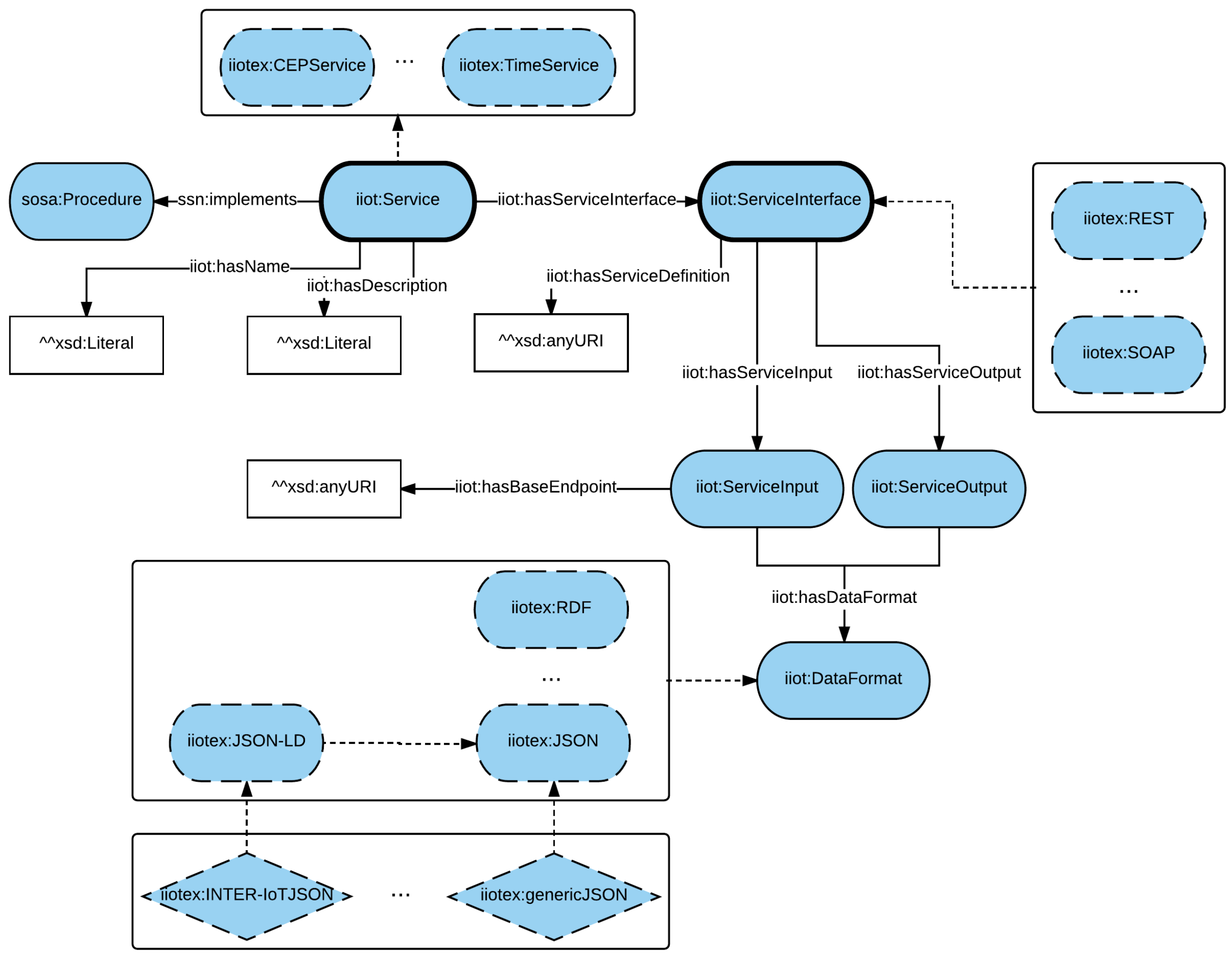

Figure 4.9: GOIoTP service module summary diagram.

of identifiers INTER-IoT uses its own RDF URI-based identification system. The *iiotex:hasLocalId* property lets platforms that use INTER-MW attach information about their local identifier, without the need to change its own internal (i.e. local) identification scheme to suit INTER-IoT. Instead, this information is treated as something relevant only in the scope of a particular identifier system, and not as a globally unique identifier.

4.3.1.10 Prefixes

Name	prefix	URI
GOIoTP	iiot	`http://inter-iot.eu/GOIoTP#`
GOIoTPex	iiotex	`http://inter-iot.eu/GOIoTPex#`
SSN	ssn	`http://www.w3.org/ns/ssn/`
SOSA	sosa	`http://www.w3.org/ns/sosa/`
SSNX	oldssn	`http://purl.oclc.org/NET/ssnx/ssn#`
Open Geospatial URI base	ogc	`http://www.opengis.net/`
OGC simple feature geometries	sf	`http://www.opengis.net/ont/sf#`
OGC geometries	gml	`http://www.opengis.net/ont/gml#`
GeoSPARQL	geosparql	`http://www.opengis.net/ont/geosparql#`
GeoSPARQL functions	geosparqlf	`http://www.opengis.net/def/function/geosparql/`
GeoSPARQL rules	geosparqlr	`http://www.opengis.net/def/rule/geosparql/`
NASA SWEET units	sweet_units	`http://sweet.jpl.nasa.gov/2.3/reprSciUnits.owl#`
NASA SWEET mathematical operations	sweet_oper	`http://sweet.jpl.nasa.gov/2.3/reprMathOperation.owl#`
NASA SWEET representation	sweet_repr	`http://sweet.jpl.nasa.gov/2.3/repr.owl#`
NASA SWEET mathematical relations	sweet_mrela	`http://sweet.jpl.nasa.gov/2.3/relaMath.owl#`
NASA SWEET scientific relations	sweet_screla	`http://sweet.jpl.nasa.gov/2.3/relaSci.owl#`
W3C time	time	`http://www.w3.org/2006/time#`
Vocabluary annotation	vann	`http://purl.org/vocab/vann/`
Vocabluary of a Friend	voaf	`http://purl.org/vocommons/voaf#`
Friend of a friend	foaf	`http://xmlns.com/foaf/0.1/`
Dublin Core elements	dc	`http://purl.org/dc/elements/1.1/`
Dublin Core	dcterms	`http://purl.org/dc/terms/`
Semantic Web Status	vs	`http://www.w3.org/2003/06/sw-vocab-status/ns#`
OWL	owl	`http://www.w3.org/2002/07/owl#`
RDF	rdf	`http://www.w3.org/1999/02/22-rdf-syntax-ns#`
RDF schema	rdfs	`http://www.w3.org/2000/01/rdf-schema#`
XML	xml	`http://www.w3.org/XML/1998/namespace`
XML datatypes	xsd	`http://www.w3.org/2001/XMLSchema#`

Table 4.2: Ontology prefixes for GOIoTP

5 INTER-IoT Reference Architectures

The Reference Architecture of INTER-IoT is derived from the IoT-A Reference Architecture and is a refinement of the previous work in Deliverable D4.1. The main target has been to simplify and to make it more usable by everyone. While the concepts remain largely the same, the way that they are laid out should help final users to get acquainted sooner with the different concepts and to use the RA as a tool for their daily design.
As a reminder, a Reference Architecture is a blueprint for developing Concrete Architectures. Its main scope is to apply different views to functionality areas, in order to identify the relationships between different modules and the need of different functions.

5.1 Functional View

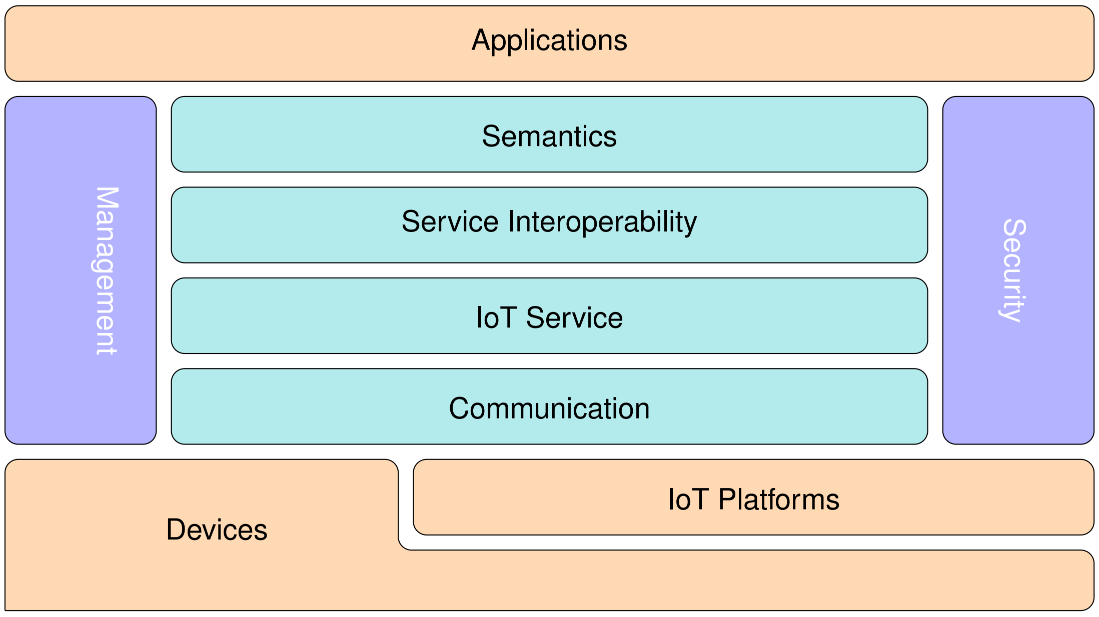

Figure 5.1: Overall Functional Architecture

Table 5.1 shows the approach we followed for defining the final INTER-IoT Functional Architecture. There are 9 different functional groups: Devices, IoT Platforms, Management, Security, Applications, Communication, IoT Services, Service Interoperability and Semantics.
Devices and IoT Platforms are kept “separate” in order to illustrate the different nodes that can be part of an INTER-IoT architecture. However, both of these groups fall outside the scope of the analysis,

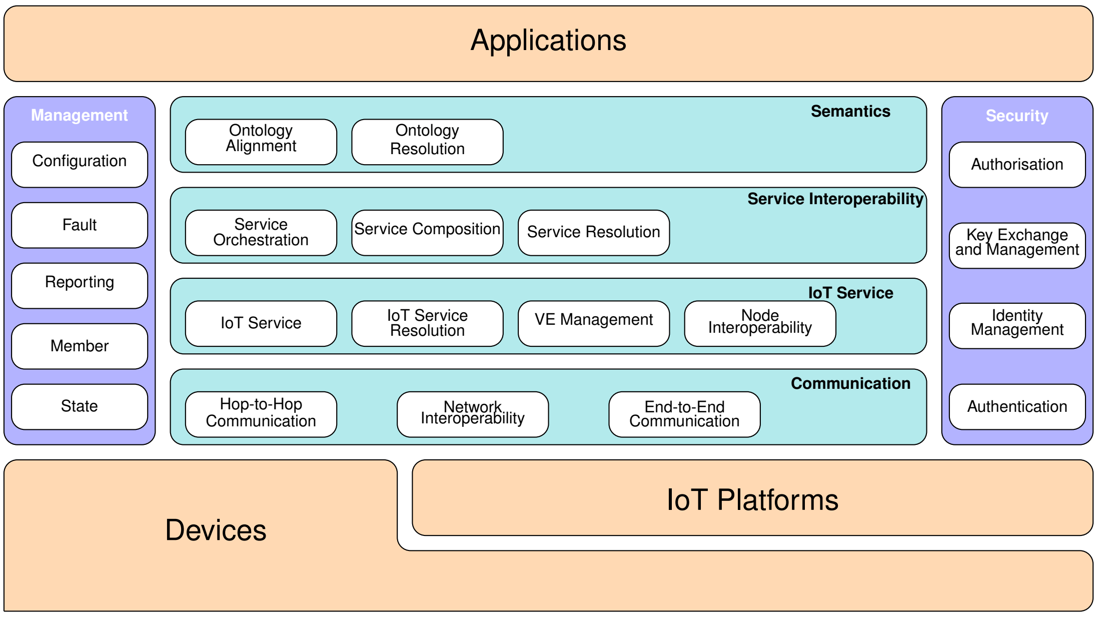

Figure 5.2: Mapping of the modules in the FA groups

together with the Application group, and therefore will be ignored.
For what concerns the other 6 functional groups, the explanations are in the following subsections.

5.1.1 Semantics

The Semantics FG is the central Functional Group that addresses the challenges related to semantic interoperability of IoT Platforms. This group has been unchanged in relation to D4.1, so we will just cite its main characteristic for completeness. The main purpose of this FG is to provide support for the Service Interoperability FG.

The Semantics FG consists of two Functional Components (see fig 5.3)

- Ontology Resolution;
- Ontology Alignment.

Semantics
Ontology Alignment
Ontology Resolution

Figure 5.3: Semantics FG

The **Ontology Resolution** FC is responsible for managing the different ontologies used by various IoT Platforms connected through INTER-IoT. These ontologies have a double approach:

- Syntactic knowledge;
- Semantic knowledge.

The syntactic knowledge is about being aware of the syntax that the IoT Platforms uses for interchanging data, what usually is related to the communication protocol being used or the type of the API: JSON, XML, etc. The semantic knowledge is about being aware of the structure and meaning of the data, usually through OWL or similar definitions (JSON-schema, XSD, etc.). The Ontology

Resolution FC is the component that stores these data descriptions and offers access to them for the Ontology Alignment FC.
The **Ontology Alignment** FC is responsible for performing the alignment from a source data with an ontology to a target data with its own ontology. It makes the data translation between two ontologies, using the ontology definitions resolved by the Ontology Resolution FC.

5.1.2 Service Interoperability FG

The role of the Service Interoperability FG is to support the Application & Service to Application & Service (AS2AS) interoperability through the definition and execution of new compound services that make use of already existing services in the underlying IoT Platforms. Therefore, its goal is to use services from different IoT systems and create new services based on them.
The Service Interoperability FG consists of three Functional Components (see figure 5.4):

- Service Resolution;
- Service Composition;
- Service Orchestration.

Service Interoperability
Service Orchestration
Service Composition
Service Resolution
Figure 5.4: Service Interoperability FG

The **Service Resolution** FC is responsible for the storage of what we call flows. A flow is a logical definition of a sequence of steps, each of which can be a service existing in an IoT Platform. The functions of the Service Resolution FC are three: (1) to resolve the access to IoT Platform services that can be used in a flow, (2) to store the definition of services and atomic components so that they can be used by the Service Composition FC and instantiated by the Service Orchestration FC and (3) to provide storage and access to the logical definition of flows. The flows that are defined for service interoperability have to be stored by the Service Resolution FC, also enabling the semantic cataloguing of services and their discovery.
The main role of the **Service Composition** FC is to design new compound services based on services that IoT Platforms exposes. These services have been previously defined and catalogued by the Service Resolution FC. The new services are designed like flows which will be later executed. The flows that are designed by the Service composition FC are stored by the Service Resolution FC.
Finally, the **Service Orchestration** FC is responsible for the execution of the flows that are stored in the catalogue managed by the Service Resolution FC. The execution of these flows are initiated by triggers (user request, IoT Platform event or alert, data received, etc.) which have been defined for each flow.

5.1.3 IoT Service

The IoT Service Functional Group is responsible of linking the existing services, devices and platforms. From the organisation point of view, it's the Functional Group that bears the biggest difference from the previous version of the Functional View. In order to simplify, the previous functions and concepts that were listed in the Device Interoperability FG and Platform Interoperability FG are now belonging to this FG. As well, the communication FC is taken out of this FG, and, as it was done in the original IoT-A Functional Architecture, is it a Functional Group that allow communication between the different nodes (Devices and Platforms).

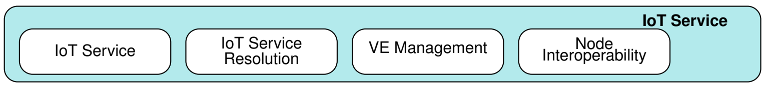

Figure 5.5: IoT Service FG

The **IoT Service** FC is responsible for managing all Services linked to the IoT nodes. These include functionalities for discovery, look-up, and name resolution of IoT Services, as well as IoT Platforms catalogs with their characteristics. This includes, for instance, all functions needed for connecting to an IoT Platform and accessing their resources (specific discovery, lookup, data query, data subscription, device registry, etc.), using appropriate protocols and APIs that each platform exposes. These functions were isolated in the Platform Access FC in the previous version of the FV; however, as they can be considered as IoT services, we tried to simplify the layout putting them within the IoT Service FC.
In general, these services expose resources of devices to the rest of the components. They may allow to gather information about a sensor in a continuous asynchronous way – after a subscription, for instance – or it may allow to submit requests to an actuator. A specific IoT Service could be to provide access to recent history of sensor observations. In the final design, the IoT Service FC is also responsible for performing device and platform interactions, like querying data from different devices and platforms in a common way, mapping sensor data flows from a source to a destination, offering subscriptions to sensor data; in the previous FV design, this function was belonging to the Platform Service FC.
The typical functions of the IoT Service FC are two:

- To access resources, interacting in three different ways: (1) to query information about a resource of a device, e.g. get current temperature of thermometer X, (2) to subscribe to observations about a resource of a device and receive notifications asynchronous for each new observation, e.g. receive all temperature measurement below 0^oC for thermometer X, (3) to submit a request to a resource of an actuator, e.g. switch light actuator Y on.
- To provide the necessary functions for finding the appropriate IoT Services, which may include: discovery, lookup, service locators, service management, etc. The IoT Service runs in the virtual plane, decoupling the interaction with the resources of devices from their usage.

The **IoT Service Resolution** FC allows any system to be able to contact IoT Services. As in IoT-A, it will also gives the different services the capability to manage their descriptions, in order to be discoverable by a class of users.
The **Virtual Entity Management** FC allows the interaction with an IoT Platform on the basis of Virtual Entities rather than IoT Services. It contains the functions to associate the Virtual Entities with the IoT Services and with the physical things they represent. The typical functions of the Virtual Entity FC are:

- Discovery and lookup functions to find VEs and their resources and register of new ones.
- Handling VEs, which includes getting the values of the entities' attributes, updating this data, and accessing its recent history.

The **Node Interoperability** FC is responsible for implementing the primitives for accessing the different nodes (both devices and IoT Platforms).
In case of Devices, interoperability means that rules are defined for each set of devices. Devices may be either semantically or syntactically interoperable, meaning that they may be able to communicate using a common communication protocol or their actions may be linked following a a logical set of rules. Therefore, this FC implements the directives for allowing two devices to interact.

In case of Platforms, there can also be substantial differences for what concern interoperability, particularly at semantic level. in this case, this FC will implement the Ontology properties as specified in the Semantics FG.

5.1.4 Communication

As in the original IoT-A Functional view, we thought that for simplicity matters having a FG dedicated to handle communication within different entities would greatly help the understanding and the usage of the Functional View in general.

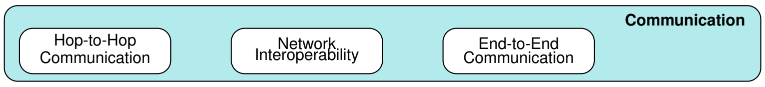

Figure 5.6: Communication FG

However, we introduced the new concept of **Network Interoperability** FC. This FC extends the original Network Communication FC, which took care of enabling communication between networks through Locators (addressing) and ID Resolution. As INTER-IoT is focusing on interoperability between different systems, this FC is responsible for managing the interoperability between networks or parts of the network that belong to an IoT deployment. We understand the network level of an IoT deployment as the protocols, systems, and devices that work on the layer 2 and 3 of the OSI stack of protocol. The particularity of the network on the IoT is the treatment of many different types of data flows as well as protocols to support this communication. The Network Interoperability FC addresses the mobility of objects through different access networks or secure seamless mobility and the backing of real time data among the network. The operation in highly constrained environment is also an important issue. The interoperability solution is based on software defined paradigms but mainly on two approaches: SDR for interoperability on access network and SDN/NFV for the core network. IoT-A
Similarly to IoT-A model, we have the **Hop To Hop Communication** FC, which provides the first layer of abstraction from the device's physical communication technology, enabling the usage and the configuration of any different link layer technology from any kind of devices. This FC is therefore focusing on Device communications. Its main functions are to transmit a frame from two different devices belonging to the same network. The Hop To Hop Communication FC is also responsible for routing a frame.
Furthermore, the **End To End Communication** FC takes care of the whole end-to-end communication abstraction. This includes reliable transfer, transport and, translation functionalities, proxies/gateways support and of tuning configuration parameters when the communication crosses different networking environments. in INTER-IoT, this means both between devices belonging to different networks, both constrained and unconstrained, and different IoT platforms.
This FC is responsible to set up a channel between different actors, so that they are able to send and receive a message. It also takes care of protocol translation (for instance, COAP to HTTP or IPv6 to 6LowPAN).

5.1.5 Security

The Security FG is responsible for ensuring the security and privacy during the interaction of all systems.
It consists of four functional components:

- Authorisation;

- Key Exchange & Management;
- Identity Management;
- Authentication.

The **Authorization** FC manages the different policies and perform access control decisions. This access control decision can be called whenever access to a restricted resource (whether a device or a specific service from an IoT platform) is requested. For example, this function can be called from the IoT Service Resolution FC, to check if a user is allowed to perform a lookup on a specific resource. Needless to say, this FC plays an important role to protect privacy of users.
The two basic functionalities offered by the Authorization FC are 1. to determine whether an action is authorized or not -the decision is made based on the information provided from the assertion, service description and action type- and 2. to manage policies. This refers to adding, updating or deleting an access policy.

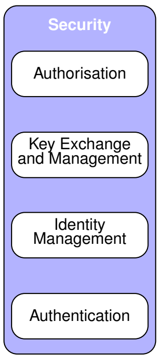

Figure 5.7: Security FG

The **Authentication** FC, as the name states, provides user and service authentication mechanisms. The basic functionality it provides is checking the credentials provided by a user, and returning an assertion (yes / no). If the requester has provided the right credentials, it establishes secured contexts between this node and various entities in its local environment. Therefore, the two functionalities provided by the Authentication FC are 1. to authenticate a user based on provided credential and 2. to verify whether an assertion provided by a user is valid or invalid. The **Identity Management** FC addresses privacy questions by issuing and managing pseudonyms and accessory information to trusted subjects so that they can operate (use or provide services) anonymously. Creating a ephemeral identity is fundamental in preserving privacy between data exchange between two parties that may not trust each other: therefore, this FC is in charge of creating such an identity and the necessary security credentials during the authentication process, in order to comply with privacy directives and at the same time provide assurance about the legitimacy of the data and identity.
The **Key Exchange and Management** (KEM) FC is involved to enable secure communications between two or more peers that do not have initial knowledge of each other or whose interoperability is not guaranteed, ensuring integrity and confidentiality. Two functions are attributed to this FC:

- Distribute keys in a secure way: upon request, this function finds out a common security framework supported by the issuing node and a remote target, creates a key (or key pair) in this framework and then distributes it (them) securely. Security parameters, including the type of secure communications enablement, are provided;
- Register security capabilities: nodes and gateways that want to benefit from the mediation of the KEM in the process of establishing secure connections can make use of the register security capabilities function. In this way the KEM registers their capabilities and then can provide keys in the right framework.

5.1.6 Management

In a similar way than IoT-A did, we followed FCAPS for the Management FG. We didn't think ti was worth to change it as it incorporates already established standard recommendations from ITU-T, and

already used in IoT applications. Therefore, our Management FG is the same as the one used in IoT-A.
FCAPS stand for:

- Fault;
- Configuration;
- Accounting (Administration);
- Performance;
- Security.

The importance of Security in the IoT context was the basis of the decision to keep it completely separate from this WG, and to have a dedicated FG instead. As well, we consider Accounting as an IoT Service, therefore will be covered within that FG.
In details, the **Configuration** FC is responsible for initialising the system configuration such as gathering and storing configuration from different Devices and IoT platforms. It is also responsible for tracking configuration changes if any and planning for future extension of the system. As such, the main functions of the Configuration FC are to retrieve a configuration and to set the configuration:

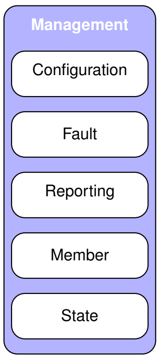

Figure 5.8: Management FG

- The retIoT-A configuration function allows to retrieve the configuration of a given system, either from history (latest known configuration) or directly from the system (current configuration, including retrieval of the configuration of one or a group of Devices), enabling tracking of configuration changes. The function can also generate a configuration log including descriptions of Devices and IoT Platforms. A filter can be applied to the query;
- The set configuration function is mainly used to initialise or change the system configuration.

The **Fault** FC handles all different failures that can happen within a given action. As such, it identifies, isolates, corrects and logs any behaviour that is not following the correct procedures. As in normal error handling, this FC is supposed to intervene whenever an unexpected behaviour occurs in any FC. This FC is then notified and can then ask for additional data or context. Fault logs are one input used for compiling error statistics. Such statistics can be used for identifying fragile functional components and/or devices.
The **Member** FC is responsible for the management of the membership and associated information of any relevant entity (FG, FC, IoT Service, Device, Communication, IoT Platform, Ontology, Interoperability, Application) to an IoT system. It is typically a database storing information about entities belonging to the system, including their ownership, capabilities, rules, and rights. This FC works in tight cooperation with FCs of the Security FG, namely the Authorisation and Identity Management FCs. The Member FC allows to update and retrieve members belonging to a specific domain.
The **Reporting** FC can be seen as an overlay for the other Management FCs. allowing to retrieve reports on specific set of topics.

The **State** FC monitors and predicts state of the IoT system. For a ready diagnostic of the system, as required by Fault FC, the past, current and predicted (future) state of the system are provided. This functionality can also support billing. The rationale is that Functions/Services such as Reporting need to know the current and future state of the system. For a ready diagnostic of the system one also needs to know its current performance. This FC also encompasses a behaviour functionality, which forces the system into a particular state or series of states. An example for an action for which such functionality is needed is an emergency override and the related kill of run- time processes throughout the system. Since such functionality easily can disrupt the system in an unforeseen manner this FC also offers a consistency checks of the commands issued by the change State functionality in the State FC. The functions of the State FC are to change or enforce a particular state on the system. This function generates sequence of commands to be sent to other FCs. This function also offers the opportunity to check the consistency of the commands provided to this function, as well as to check predictable outcomes (through the predict State function). A second function is to monitor the state. This function is mainly used in subscription mode, where it monitors the state of the system and notifies subscribers of relevant changes in state. Other functions of the FC are to predict the state for a given time, to retrieve the state of the system through access to the state history and to update the state by changing or creating a state entry.

5.1.7 Using the Functional View

The Functional View illustrated above is supposed to be used in a "live" way. We will show in section 6 how to use in a practical way the different Functionality Groups and Components shown above, so that it will be easier for developers to get acquainted with the different schemas.
The use cases are taken from the Deliverable D2.4 (INTER-IoT Requirements and Business Analysis), D2.3 (Use cases manual).

5.2 Information view

The main reason about bringing connectivity to objects is to gather information about the environment and be able to modify it. This information exchange can happen directly between objects, or more commonly between an object and a back system. Therefore, the way to define, structure, store, process, manage and exchange information is fundamental in the connected objects domain.
Following the IoT-A created a specific view (the information view) in order to specify a static information structure and a dynamic information flow.
Based on the IoT Information Model, the Information View gives more details about how the relevant information is represented in an IoT system. As the Information View belongs to the reference architecture space, and not a specific system architecture, concrete representation alternatives are not part of this view. The information view also describes the components that handle the information, the flow of information through the system and the life cycle of information in the system. As described earlier, the Virtual Entity is a key concept of any IoT system as it models the Physical Entity that is the real element of interest. As specified in the IoT IM, Virtual Entities have an identifier (ID), an EntityType and a number of attributes that provide information about the entity or can be used for changing the state of the Virtual Entity, triggering an actuation on the modelled Physical Entity. The modelling of the EntityType is of special importance, as it can be used to determine what attributes a Virtual Entity instance can have, defining its semantics. The EntityType can be modelled in two different ways: either based on a flat type system or as a type hierarchy, enabling sub-type matching. EntityTypes are similar to classes in object-oriented programming, so UML class diagrams are

suitable for modelling EntityTypes. Similarly, the generalization relation can be used for modelling sub-classes of EntiyTypes, creating a hierarchy of several EntityTypes inheriting attributes from its super-classes. Services provide access to functions for retrieving information or executing actuation tasks on IoT Devices. Service Descriptions contain information about Service interfaces, both on a syntactic as well as a semantic level. Furthermore, the Service Description may include information regarding the functionality of the resources, or information regarding the device on which the resource is running. The association between Virtual Entities and Services captures the information on what kind of actuation or data is possible to obtain by which Virtual Entity. The association includes the attribute of the Virtual Entity for which the Service provides the information or enables the actuation as a result of a change in its value. Information in the system is handled by IoT Services. IoT Services may provide access to On-Device Resources, that provide real-time information about the physical world accessible to the system. Other IoT Services may further process and aggregate the information provided by IoT Services/Resources, deriving additional higher-level information. Furthermore, information that has been gathered by the mentioned IoT Services or has been added directly by a user of the IoT system can be stored by a special class of IoT Service, the history storage. A history storage may exist on the level of data values directly gathered from sensor resources as a resource history storage or as a history storage providing information about a Virtual Entity as a Virtual Entity history storage.

6 Guidelines

This chapter is a guide for the application of the reference architecture in real environments. For that, real scenarios for the two application domains are described, in which it is analyzed how the reference architecture can be applied.

6.1 INTER-LogP

6.1.1 IoT support for transport planning and execution

Scene This scene refers to the IoT support for transport planning and execution scenario, that facilitates the provision of connected services to plan and execute transport services in a port environment through interoperable IoT platforms. Both the vehicle (physical entity) and the driver (user) send and receive data from the road hauler's IoT cloud platform. The data sent from the vehicle, comes from multiple sensors and devices placed on it and is then processed through a virtual entity that will provide management decisions in the form of data or explicit orders (e.g. designation of the port terminal to go to, loading order etc.) aiming at improved transport services. Parallel to this, the driver interacts with sensors and devices to send data to the road hauler that also relies on the same virtual entity that will produce management decisions. At the same time, vehicles in the port area always send data to the road hauler's IoT cloud platform as well as to that of the port authority, informing them both of the arrival of the position vehicle. In case of need of extra vehicles not available within the road hauler's fleet, an external vehicle can be used. This vehicle provides data to the virtual entity to which it has been granted access by the road hauler. Vehicles in the port area, when dealing with containerized shipments, also notify the container terminal IoT platform of their whereabouts.
Scene summary Location: port area
Covered topics:

- Transport planning and execution
- Remote transport services management
- Smart Objects

Key benefits: implementation of seamless management processes that allow for efficient, simplified and optimized decision-making while providing partially shared information to involved stakeholders (hauler, port authority, driver, container terminal).
Description from the user's point of view Jack is a professional road transport professional driver who has arrived at the port upon receiving an order to pick up a container. Upon arrival, the sensors and devices on board the vehicle notify both the port authority and the container terminal's IoT platforms of his presence. In turn, upon processing of the received data and subsequent orders sent by

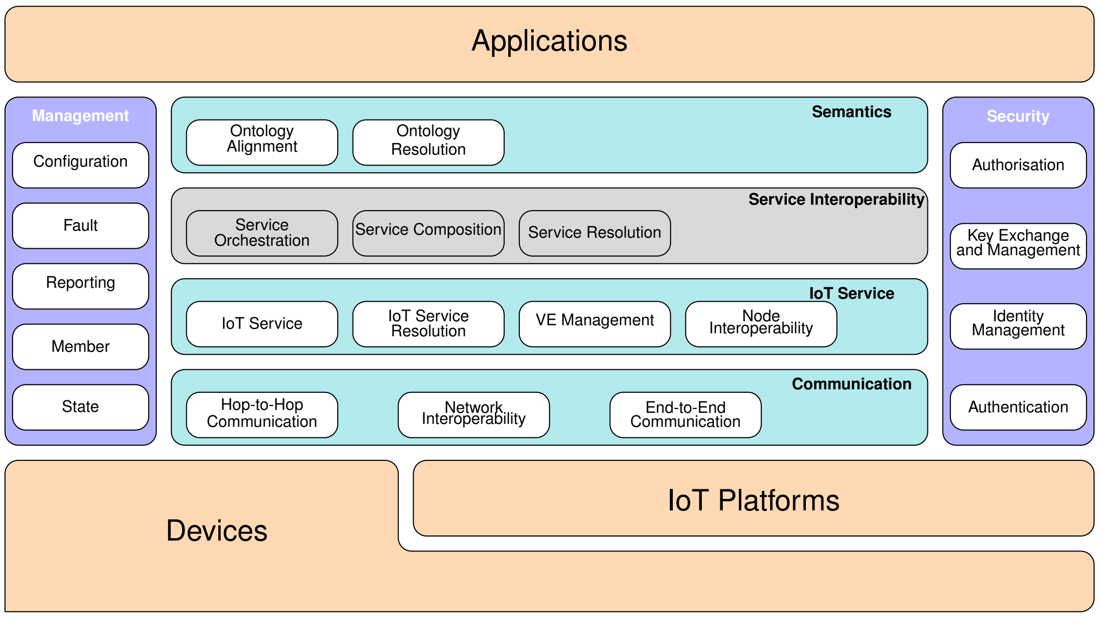

Figure 6.1: RA Functional view in scenario 1

the hauler , the driver knows exactly at which designated area of the container terminal , at what time he should present himself to load the container. Upon completion of this task, the hauler receives data from the sensors and devices in the vehicle, informing him of the departure of the vehicle, thus allowing him to update in real-time the transport planning for the day.

Relationship with the Reference Architecture In this scenario there are two kind of interoperability levels. On one hand, the truck needs a gateway to connect al the sensors inside the truck and send the data to the hauler company IoT platform. Therefore, Device Access and Device Interoperability functional groups are required. On the other hand, there is an exchange of information between platforms, when the truck access to the port. In this case, the platform interoperability functional group is necessary to send the information. The syntactic and semantic translation of the data is also needed, so the semantic functional group will be used.

6.1.2 IoT weighbridges

Scene This scene shows how transport weighbridge operations can be facilitated through IoT-based coordination of weighbridging operations. Chronologically, when a vehicle arrives near the weighing area, the devices and sensors lodged on it send arrival data to both the weighbridge's the road hauler's company IoT platforms. The weighbridge platform accesses the data through temporary access it is granted by the road hauler to the virtual entity, allowing the weighbridge platform to send, in turn, data and orders to the driver so that he knows when to approach the bridge to start weighing operations. The weighbridge IoT platform knows vehicle to which the data and orders has to be sent through an automatic identification of the vehicle (e.g. license plates recognition). The weight information is then automatically sent to the destination (customer) once received by the hauler from the weighbridge IoT platform.

Scene summary Location: port container terminal

Covered topics:

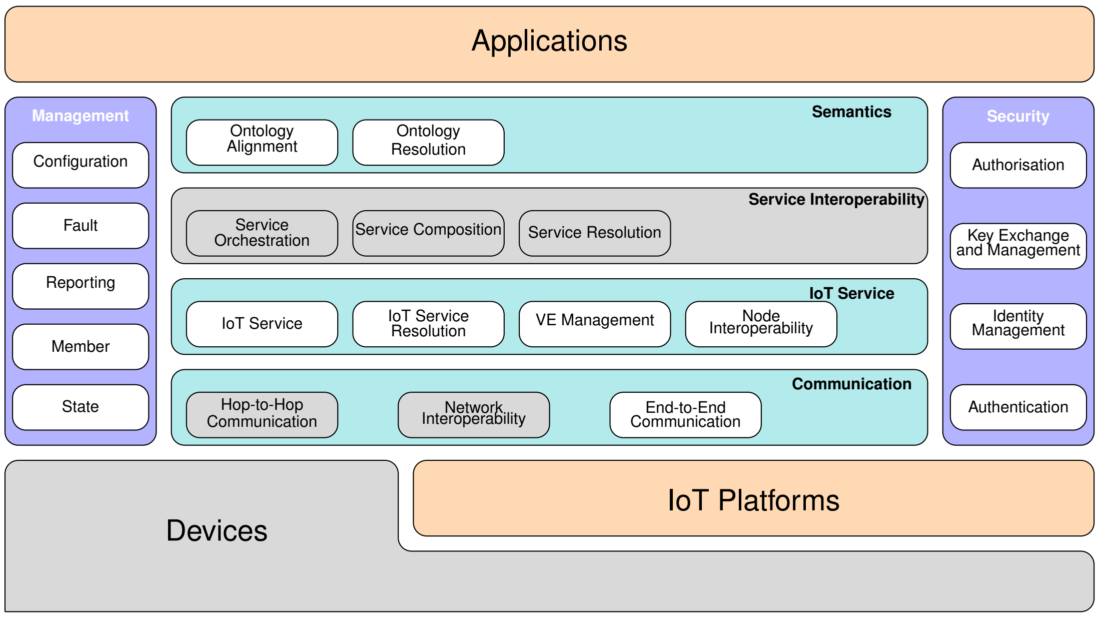

Figure 6.2: RA Functional view in scenario 2

- Weighbridge operations
- Transport planning and execution
- Customer Relations Management
- Smart Objects

Key benefits: an optimized and time-saving organization of weighing activities allowing for facilitated customer relations features that can help value transport services.

Description from the user's point of view Sam is an experienced driver who has just picked-up a container and is proceeding to the weighbridge area of a port container terminal. Upon approach, based on the data received from the vehicle, the road hauler grants access to a virtual entity to the weighbridge authority. At the right time, weighbridge authority sends an order to proceed to weighing the vehicle. The driver arrives at the indicated time. As soon as the weighing operation is completed, the information is transmitted to the road hauler IoT platform. At the same time, the haulier sends this weight information to the client as a valuable customer relationship service. Thus, all valuable outcomes of this weighing operation are seemingly fulfilled without having to resort to multiple and time-consuming human operations (e.g. identification through physical observation, administrative transmission of weight information to the client, indirect orders given by weighbridge authorities to driver etc.).

Relationship with the Reference Architecture In this scenario exist an exchange of information between two platforms. Therefore, the Platform Interoperability functional group is essential. And, since the two platforms do not have to use the same ontology, the semantic functional group would also be used.

6.2 INTER-Health

6.2.1 Chronic disease prevention

Scene The core scenario is about fostering healthy lifestyle to prevent chronic diseases by means of a decentralized monitoring at patients' homes taking advantage of INTER-Health system. Healthcare professionals will follow the progress of the patients based on the data taken at their homes with health devices: it will be a scale, a sphygmomanometer and a bracelet. Weight will be taken weekly, blood pressure will be taken daily but only if the patient is hypertensive, and physical activity every day in terms of minutes and steps. Patients will also have to fill in, twice a month, questionnaires about eating and physical activity habits. Patients will be motivated with positive messages on the phone. This scenario will increase cost-effectiveness of the applied resources.
The patient who decides to participate in the experimental study is the Physical Entity. The Virtual Entity are the *sensitive measurements detected* by health devices and a smartphone. The connection between the Virtual and Physical Entity are these resources: *INTER-Health platform*, the *user interface on PC/tablet/phone* and the *Computerized Nutritional Folder*. The patient's Virtual Entity is associated with the INTER-Health platform through these devices: *bluetooth scale, sphygmomanometer, activity wristband* and *computer questionnaire*. The users are the *patient*, *Family doctors* and *health operators*. The chronic diseases prevention services are the *Health Status Notification for patient, family doctors and health operators* and the *Alert System by phone*.
Scene summary Location: Patient's house
Covered topics:

- Home Care
- Remote health monitoring
- Smart Objects

Description from the user's point of view Simona is a 54 years old housewife who is in a situation of overweight and decide to participate in an experimental study for monitoring her lifestyle at the Prevention Department of her healthcare center. After being recruited, Simona receives the experimental kit (Bluetooth scale and a quantitative bracelet). After the first visit at the healthcare center, where the health professional records Simona's weight, height, body mass index, eating habits and physical activity, Simona will take next measurements at her home, so in this way she has to come back to the healthcare center at 6th and 12th month only. The weight and steps number will be collected by health devices, while eating and physical activity habits will be detected filling online questionnaires. All this information will be gathered through a mobile application. The healthcare professionals can consult all Simona data through the Professional Web Tool, which is available in INTER-Health system, being able to detect any change and provide motivational messages that can improve their habits in pro of a better life. Now, Simona doesn't need to go to the hospital and spend time there. INTER-Health system provides tools enough to the health professionals to follow her status and apply any medical action when needed.
Relationship with the Reference Architecture The patient's devices and the healthcare center devices work on different separate platforms, and must be accessed by the same application regardless. So there is need for the Platform Interoperability functional group. Since both platforms utilize different models and formats, the Semantic functional group also comes into place.

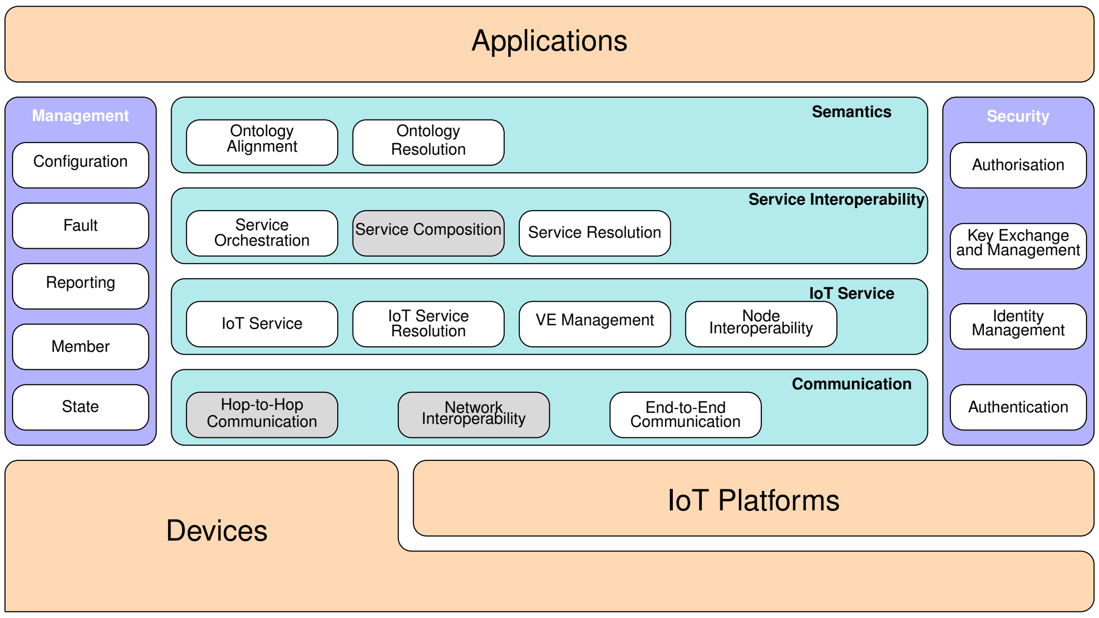

Figure 6.3: RA Functional view in scenario 3

6.2.2 Surveillance systems for prevention programs

Scene This scene shows how important the wide use of computers and other devices in the adolescent segment of the population for health promotion in children, the school is in fact considered one of the most favorable setting to prevent overweight and obesity being the environment school, the place where the majority of children and adolescents spend most of the day. In this case, the instrument used is a multifunctional device healthy kiosk detecting in school a certain kind of objective measures (anthropometric data) of boys with IoT strategy allowing to monitor and control the health status of children (telemedicine objectives).
The Physical Entity is the student involved in the prevention monitoring program. The Virtual Entity are the *sensitive measurements detected* by the multifunctional device healthy kiosk The connection between the Virtual and Physical Entity are these resources: *Multifunctional device healthy kiosk* and the *Users Interface on PC* consulted by the parents. The Virtual Entity is associated with the Multifunctional device health Kiosk through the *bluetooth scale, sphygmomanometer* and *altimeter* devices. Users are the *parents, family doctors* and *health operators*. Prevention program services are the *Health status notification for parents, family doctors and health operators* and the *Nutritional education program by health operators* for preventive health scenarios.
Scene summary Location: School
Covered topics:

- Multifunctional devices healthy kiosk
- Remote health monitoring
- Smart objects

Description from the user's point of view James is a 16 years old student, after his parents signed informed consent, he has the right to choice to take part in the monitoring of measurements for the multifunctional device healthy kiosk, present in the gymnasium of his school. There are health operators who, in collaboration with teachers, help guys in the collection of measurements. James is a normal-weight guy (BMI 24) but he has a waist circumference of 98 cm and the blood pressure level is borderline. Parents through telemedicine can consult in a timely punctual way the data relating to measures of their son and decide to consult their family doctor. The family doctor recommends James to follow a proper nutrition and physical activity being a subject with risk for cardiovascular disease; a borderline condition in adolescence can become pathological condition in adulthood. Even health operator can search James's data, as part of telemedicine they can program targeted interventions for nutritional education as a preventive measure.

Relationship with the Reference Architecture Device access and Device interoperability functional groups are required since they facilitate the connection to the Kiosk. To obtain a certain level of independence at application level, the Semantic functional group can be used to abstract the information modeling.

7 Ethics

7.1 Introduction

Ethics is a central consideration to all INTER-IoT planning and development. As requested at the interim review, an ethical advisory board has been established. This board, within INTER-IoT, continuously reviews ethical issues. The aim of the committee is to ensure that ethical considerations and issues are addressed in the conduct of the research and development work undertaken within the project. The committee seeks to support and encourage the process of ethically conducted research to maintain the safety and well-being of participants and researchers to promote ethical values.
This deliverable focus its objective in the improvement of Reference Model, Meta-data and Reference Architecture for the INTER-IoT project. Some of the ethical objectives to take in account are the related with ethical issues of security, privacy and confidentiality, increased expectations, and withdrawal of the service. Moreover, in specific scenarios or use cases like the ones related with INTER-Health we need to extend our ethical focus in key issues as the ethical concepts of autonomy, independence, quality of life, beneficence, non-maleficence and justice. Additionally, the data protection and enclosure is guaranteed as have seen before in D3.2 ethical section.

7.2 Data Protection

As data protection is main objective into the project, and moreover, within this deliverable the ethical review of data management has been specified to focus on personal data, meta-data and sensitive data. Personal data means data which relate to a living individual who can be identified:

(a) from those data

(b) from those data and other information which is in the possession of, or is likely to come into the possession of, the data controller, and includes any expression of opinion about the individual.

Additionally, any indication of the intentions of the data controller or any other person in respect of the individual.
Sensitive data means personal data consisting of information as to

(a) the racial or ethnic origin of the data subject,

(b) his political opinions,

(c) his religious beliefs or other beliefs of a similar nature,

(d) whether he is a member of a trade union,

(e) his physical or mental health or condition,

(f) his sexual life,

(g) the commission or alleged commission by him of any offence, or

(h) any proceedings for any offence committed or alleged to have been committed by him, the disposal of such proceedings or the sentence of any court in such proceedings.

It is possible that INTER-IoT will be used during processing of these types of data, so appropriate controls have to be built into the layer components to enable systems to do this ethically by conforming to the data protection act. This is achieved by assuring the proper security of the communications and data handling within the components of each layer and across layers, as per the security considerations described in previous sections, encompassing the common aspects of integrity, privacy, authentication, authorization of information systems, and the other, more "IoTspecific" ones of pseudonimity, autonomous communication, and semantic querying. Ethics in INTER-IoT concerns issues related with General Data Protection Regulation (GDPR). It is EU regulation containing provisions on the protection of individuals with regard to the processing of personal data and free flow of personal data. The purpose of the regulation is to achieve full harmonization of substantive law within the EU and the free flow of personal data. The main idea of GDPR is to protect personal data at every phase of data processing.

- Lawfulness, fairness, and transparency: personal data should be processed lawfully, fairly, and in a transparent manner.
- Limited purpose: personal data should be collected for specified, explicit, and legitimate purposes and not further processed in a manner that is incompatible with those purposes
- Data minimisation: personal data should be adequate, relevant, and limited to which it is necessary in relation to the purposes for which they are collected.
- Accuracy: personal data stored and managed should be accurate and, where necessary, kept up to date.
- Storage limitation: personal data should be kept in a form which permits the identification of data subjects for no longer than is necessary for the purposes for which the personal data is processed.
- Confidentiality and integrity: personal data should be processed in a manner that ensures appropriate security of the personal data, including protection against unauthorized or unlawful processing and against accidental loss, destruction or damage, using appropriate technical or organisational measures.

Because the components of each layer do not interpret the payload data being transferred, handledand/or stored within them, they do not hinder the above principles per se. It is however necessary that during the process of implementation of each component the developers are aware of the principles themselves, so as to not introduce unwanted features, workarounds, hot fixes or "hacks" that interfere with them, even if in a temporary fashion, as long as there is the chance that these are utilized in the pilots real-life deployments. It therefore becomes an official recommendation for developers to read the above principles and take them into consideration during the development process.

8 Conclusions

This deliverables completes the work done in tasks 4.1 and 4.2. The effort here was devoted to develop a sustainable abstract model, that could be used in practice in order to develop a concrete INTER-IoT architecture and to drive the development in WP4, as well as influence other WPs (namely, 3 and 5).

The problem of these exercise, in general, is that they can became too abstract and practically useless, or perceived as such, by the users or the ones developing a concrete architecture. With tis in mind, we tried to simplify the work already started in D4.1 and, while fixing some issues and converging on more widely accepted solutions, we decided to have simplicity as a principle.

The problem of being too abstract and separated from the real developments is also present within the project. Therefore, we also tried to explain in details how this deliverable fits within the global INTER-IoT picture. While in the previous iteration we already put all the images on how the Functional Architecture was mapping into the different activities, here we devoted a chapter to fully and clearly explain how this work fits with the other developments in the project.

For what concerns the INTER-IoT Reference Model, we followed an evolutionary pattern and discussed in depth the changes, in order to clarify the different new inclusions. It has to be remembered that the main purpose of our project is the Interoperability between different platform, therefore the "traditional" IoT-A Domain Model had to be modified so that the specificities of our approach were taken into account. While it's possible using a standard IoT-A model to model two different systems, it's not so straightforward, so that an update was needed to make the whole process smoother.

Furthermore, the main differences may be noticed in the Functional Model. A great work was done in simplifying it, so that it became much more "user-friendly", as one of its main limitation in the previous iteration was that users were almost scared to use it, as they were not sure which box was fitting exactly their example and how to combine them. We hope that, with the layered approach we followed, it will be more immediate how to map any different example into the model and to have a representation in a smooth way.

A similar evolutionary approach was followed in the Metadata section. As we thought it was an important part of the work, it has now a chapter "on his own" rather than belong to the general area of Reference Model.

The functional view of the Reference Architecture had to be updated, following the redesign of the Functional Model in the RM. Again, we hope that simplifying the idea and the graphics will make the whole tool more usable and the lower the much feared "steep learning curve". As the other views and perspectives of the Functional Architecture were similar to the Deliverable D4.1, we decided not to repeat he same information twice, to keep this document as lean as possibile.

While the work in the modeling section is never finished, and can be refined in an infinite loop, we are very satisfied with this iteration, that we consider not only a starting point for further developments, but also a very good synthesis of the internal discussions we had and the different experiences and ideas we had.

A Approach

A.1 Introduction

There is currently a plethora of organizations trying to develop the ultimate reference models for IoT systems. It's possible to enumerate several efforts such as IoT-A [10], IEEE P2413 [6], ITU-T [7], IIC [8], oneM2M [9], just to name a few.
It must be noticed, though, that despite a very diverse vocabulary, most concepts are more or less the same. For instance, in figure A.1 it is possible to notice the fact that IoT-A functional model, ITU-T reference architecture and oneM2M functional architecture are almost equivalent.
The IoT-A Architectural Reference Model [10] has been chosen as a reference for the work performed in INTER-IoT. Reasons for this choice are:

1. IoT-A is a complete and mature solution that allows to go from a use case and a number of requirements to a concrete architecture, taking into considerations different factors such as communication between devices, security, or information flow.

2. As all architectural approaches are somehow similar, it's not time-consuming or complex to map different efforts into the IoT-A concepts.

3. IoT-A is used as a base by EU projects, providing a common ground with other EPI cluster actors.

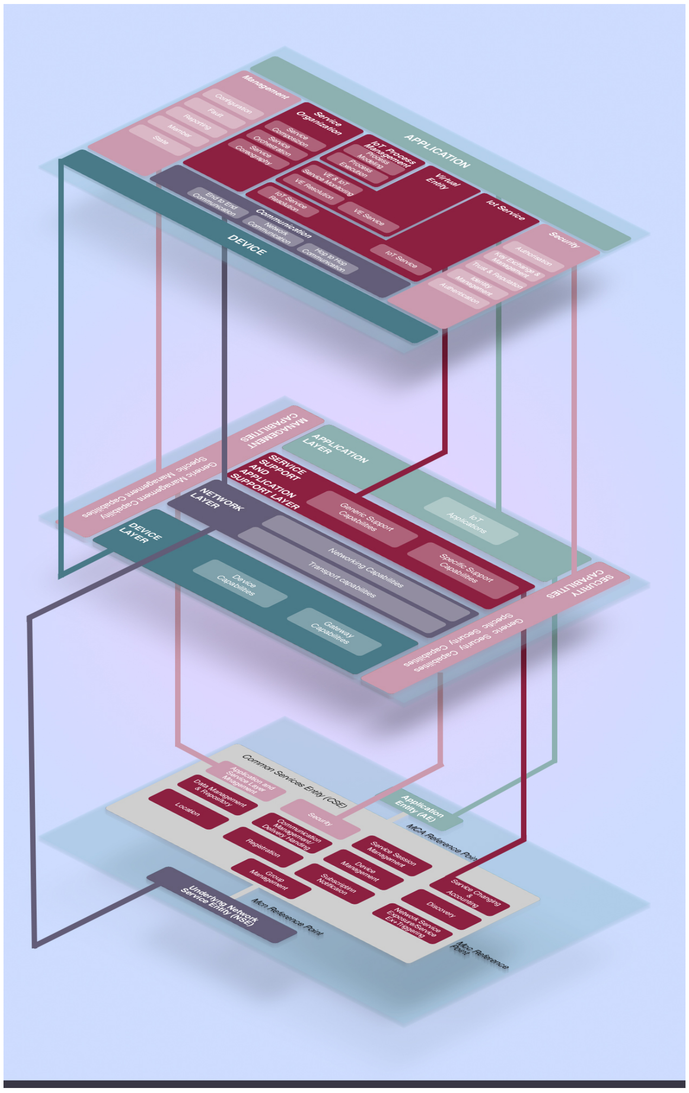

Figure A.1: Similarities between IoT architectures

The IoT-A provides a complete methodology for creating IoT platforms based on a reference architecture and using the characteristics of the use cases and requirements of a determined application. INTER-IoT, however, is intended to provide tools and mechanisms to achieve interoperability among existing IoT platforms. The INTER-IoT interoperability approach must be as generic as possible to provide global solutions regardless of the technologies or instances (platform independence), and the field of application (domain independence). Then, while the problem domain (IoT) and a lot of concepts of IoT-A are valid and applicable to INTER-IoT, the approach is different. Accordingly, this document follows the terminology and the general methodology approach described in IoT-A, to define an architecture reference model and a reference architecture to create interoperability mechanisms among IoT platforms. INTER-IoT will use this framework to instantiate several mechanisms (so-called Layer Interoperability Infrastructures, LIIs), and thus validate the Reference Architecture developed. However, as for its generic approach, the RA and/or the Architectural Reference Model (ARM) could be used to generate new solutions for different interoperability problems in the future. In the subsequent sections, the approach used to generate each of the models and views that eventually define the INTER-IoT ARM and RA is detailed.

A.2 The IoT-A Architectural Reference Model

A.2.1 IoT-A Background

IoT-A was a lighthouse EU-funded project that established an Architectural Reference Model for the Internet of Things domain. The project ran from 2010 until 2013, and can be considered the foundation for all the EU efforts done in this area since then.

IoT-A main goal was to promote a high level interoperability between different IoT systems. This interoperability had to be developed from the communication level as well as at the service and knowledge levels across different platforms established on a common grounding. The IoT-A project developed common tools and methodologies to achieve this. While existing literature provides methodologies for dealing with system architectures (hereafter called concrete architectures) based on Views and Perspectives, establishing a reference architecture is quite different.

An Architectural Reference Model (ARM) can be therefore visualized as the matrix that eventually derives into a large set of concrete IoT architectures. For establishing such a matrix, based on a strong and exhaustive analysis of the state of the art (SOTA), a super-set of all possible functionalities, mechanisms and protocols that can be used for building concrete architectures must be identified. It's possible then to select baseline technologies – such as protocols, functional components, or architectural options – according to different dimensions like distribution, security, or response time, in order to build solid INTER-LAYER/INTER-FW solutions. A usual representation of IoT-A is the IoT-A "tree".

The basic concept of the picture A.2 is that IoT-A connects several baseline technologies, such as communication protocols (6lowpan, ZigBee, IPv6, . . .) and device technologies (sensors, actuators, tags) with an almost infinite number of application and services. The trunk of the tree represents the Architectural Reference Model, composed by the Reference Model and the Reference Architectures: the set of models, guidelines, views, perspectives, and design choices that can be used for building fully interoperable concrete domain-specific IoT architectures (and therefore systems).

A.2.2 Basic Usage of the IoT-A ARM

The ARM can be used for several purposes, from more abstract ones to more concrete developments.

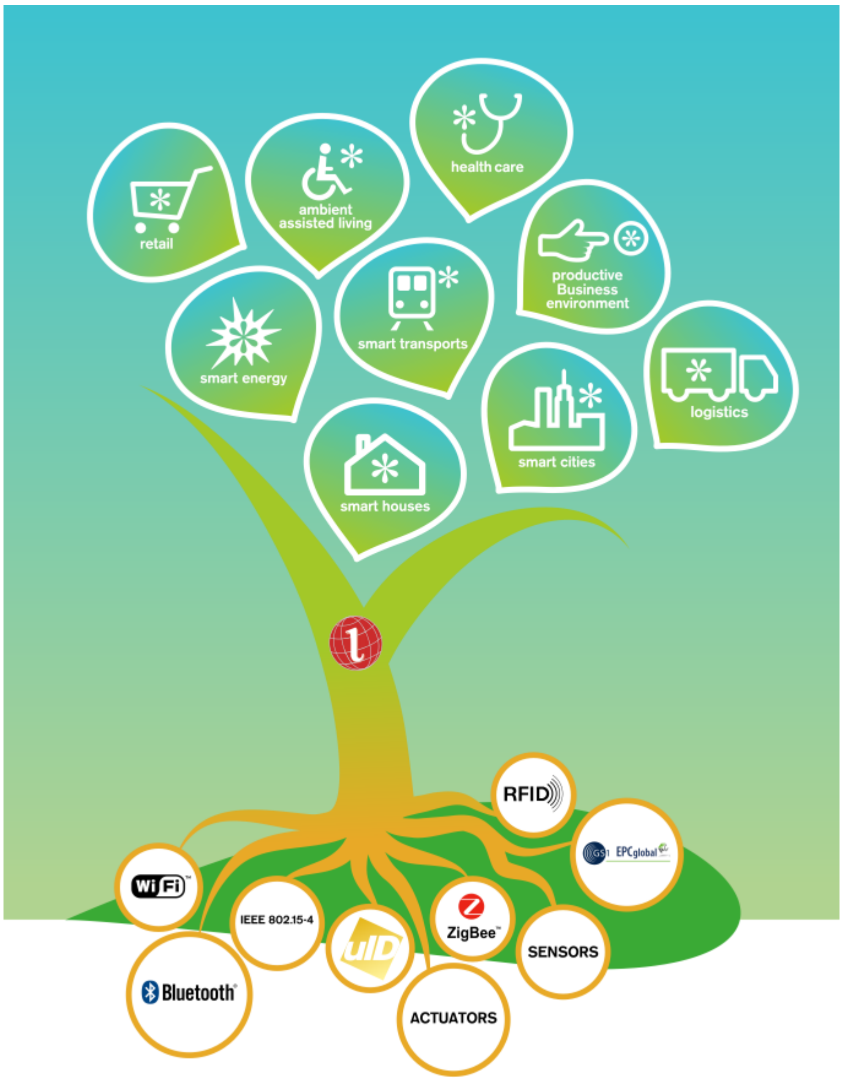

Figure A.2: The IoT-A tree

A.2.2.1 Cognitive aid

At a more abstract level, such as product conception and development, an ARM can be used for different purposes. First, it provides a roadmap for discussions, since it defines a clear language and grammar that everyone involved in the creative process can use, and which is intimately linked to the architecture, the system, and the usage domain. As well, the high-level view provided in such a model is of high educational value, since it provides a comprehensive and at the same time abstract view of the domain, helping non-technical people (or simply, people new to the IoT field) in understanding the particularities and intricacies of IoT. Furthermore, the ARM can assist IoT project leaders in planning the work and organizing the teams needed. For instance, the Functionality Groups identified in the Functional View of the IoT system can also be a list of independent teams working on an IoT system implementation. The ARM also provides a clear guidance in identifying independent building blocks for IoT systems. This constitutes very valuable information when dealing with questions like system modularity, processor architectures, third-vendor options, or re-use of already developed components. All these points show that establishing a common ground for any field is not an easy task. In this field, a common ground would encompass the definition of IoT entities and the description of their basic interactions and relationships with each other, which is the main objective of the IoT-A effort.

A.2.2.2 Generation of architectures

A major benefit of the IoT-A ARM is the capability of generating architectures for specific systems. This architecture generation is done by providing best practices and guidance for helping translating the ARM into concrete architectures. The benefit of such a generation scheme for IoT architectures is not only a certain degree of automatism of this process, and thus the saved R&D efforts, but also that the decisions made follow a clear, documented pattern.

A.2.2.3 Identifying differences in derived architectures

When using the IoT-A ARM-guided architecture process any differences in the derived architectures can be attributed to the particularities of the pertinent use case and the thereto related design choices [4]. When applying the IoT-A ARM, a list of system functional blocks and data models, together with predictions of system complexity, can be derived for the generated architecture. Furthermore, the IoT-A ARM defines a set of tactics and design choices for meeting qualitative system requirements. All these facts can be used to predict whether two derived architectures will differ and where. The IoT-A ARM can also be used in a “reverse mapping” fashion. System architectures can be cast to the IoT-A ARM language; this is what we will do in this document, analyzing the different platforms and translating the different system architectures into a common language and mapping.

A.2.2.4 Achieving interoperability

While developing a concrete architecture, fulfilling a set of qualitative requirements inevitably leads to design challenges. Since there is usually more than one solution for each of the design challenges (we refer to these solutions as design choices), the IoT-A ARM cannot guarantee interoperability between any two concrete architectures a priori, even if they have been derived from the same requirement set. Nevertheless, the IoT-A ARM is an important tool in helping to achieve interoperability between IoT systems. This is facilitated by the “design choice” process itself. During this process, it’s possible to identify the design choices made; comparing two different architectures, it should be clear what architecture measures must be taken to achieve interoperability and at which point in the respective systems this can best be done. Interoperability might be achieved a posteriori by integrating one IoT system as subsystem in the other system, or by building a bridge through which key functionalities of the respective other IoT system can be used. Notice though that these workarounds often fall short of achieving full interoperability. Nevertheless, building bridges between such systems is typically much more straightforward than completely re-designing either system; usually doing so, a fair level of interoperability can be achieved.

A.2.3 Architecture concepts

Architectural views provide a standardized way for structuring architectural descriptions. Views can also be used for structuring reference architecture descriptions. Choosing architectural views for the development of a coherent IoT Reference Architecture proved to be instrumental for the success of the IoT ARM, as they provide an intuitive delineation of each addressed aspect. There is not a single, commonly accepted list of architectural views; the chosen ones for this work are:

- Context view;
- Functional view;
- Information view;
- Deployment view;

Views do address technical aspects, while stakeholder requirements are often formulated as qualitative requirements. Their solution to this issue, which we adopt in this document, is to introduce architectural perspectives. These perspectives cut across the views. In other words, they do not replace views but provide an abstraction layer above the views.

A.3 Domain Model

The Domain Model (DM) is the first step in the creation of the reference model. In the IoT realm, the creation of a Domain Model is carried out starting from the analysis of the concepts used in the Internet of Things, like devices, things, services and so on. IoT-A's DM is a good example of this. INTER-IoT, aims at building a reference model on the foundations defined by IoT-A, leveraging terminology and representation methodology to solve the problem of interoperability in existing IoT platforms. The reason is that traditional IoT models focus on designing a system around the Internet of Things concept, whereas in INTER-IoT we deal with making a set of IoT platforms interoperable. This means that, although all the IoT concepts and models are valid, we need to extend them to consider the existence of different platforms. As a matter of fact, this could be seen as a system of systems approach, with multiple platforms and an upper actor configuring an overall system, but also as a mesh of platforms that need to interchange content through a multilayer mediator. INTER-IoT allows both approaches. Taking all that into account, we have analyzed the IoT-A's Domain Model, and have checked it against the most common sensor/actuator ontologies (W3C SSN, W3C SOSA, IETF SAREF, oneM2M, OGC Sensor Things, . . .) to check its validity.
Once we have ensured that the IoT-A's Domain Model is valid for the INTER-IoT objectives, we have extended it to include new concepts necessary for the INTER-IoT, mainly related to its multi-platform approach.
For achieving this, we have performed several steps. First, we have reviewed the requirements of the project. Next, we have made an analysis of various IoT platforms. We have been collaborating in

parallel with several IoT platform analysis tasks that have been conducted in the project. The result is a Domain Model aimed at the interoperability of IoT platforms, suited to INTER-IoT goals.

A.4 Information Model and Meta Data Model

A.4.1 Introduction

IoT-A defines a generic model of information that passes through any IoT system. The central element of this model is a *Virtual Entity* IoT Domain Model) that has some Attributes with MetaData attached. An IoT-A Virtual Entity needs to have two special data elements that describe it: an identifier, and a type (entityType). Additionally, Virtual Entities may have multiple attributes, each with a name, type, and annotated values. The description of an entity, in this model, allows for multiple values of attributes, each of which may be annotated with meta-data. The annotations may go deeper, with meta-data about meta-data and so on. This description is realized in the IoT-A information model through a *ValueContainer*, an instance of which combines an attribute value and its meta-data annotations. Additionally, IoT-A defines generic classes for *Service*, *Resource* and *Device* descriptions [10].

This generic model can be used to model a wide variety of information. In particular, in the implementations of IoT systems there is a need to have a specific definition of what meta-data items, attributes and virtual entities, a given system operates on. In fact, the IoT-A methodology itself suggests that the definition of, for instance, what entity types are available, is left to the implementer. Following the IoT-A suggestion, of using specific schemas and models to describe available types of virtual entities, attributes and meta-data, INTER-IoT uses ontologies and semantic vocabularies to define/augment the information model.

Note that it is not the role of the reference meta-data model to implement any of the described mechanisms. For instance, if the model contains information about webservices, it may inform a reader about types of services and communication protocols required to access those services, as well as supported data formats (e.g. JSON, XML, csv, etc.). It does not actually implement any service.

The INTER-IoT reference meta-data model is a set of ontologies, along with documentation, that is used to define specific implementations of IoT-A information model. The INTER-IoT model, in particular, describes the types of *Virtual Entities*, *Attributes* and *MetaData* for INTER-IoT understanding, and includes descriptions of Services, Resources and Deviceswhich are included, but not expanded upon in the IoT-A model. The process of creation of the INTER-IoT reference meta-data model is described in D4.1.

The INTER-IoT reference meta-data model is comprised of two modular ontologies - GOIoTP and GOIoTPex, described in section 4. The ontologies are expressed in Web Ontology Language (OWL), in which the basic axioms follow the RDF model of “subject, predicate, object” triple. This basic model can be mapped to any meta-data item defined in IoT-A information model. IoT-A attributeName maps to OWL property name and attributeType to property type[1] (annotation, data or object). IoT-A ValueContainers map to simple property assertion values. Similarly, any IoT-A Virtual Entity maps to an OWL resource.

Additionally, GOIoTP defines specific classes, such as *iiot:Service*, *iiot:IoTDevice*, *iiot:User* and many others, that map directly to corresponding IoT-A entities (Service, Device, User, etc). Section 4.3 contains detailed information about GOIoTP, and its entities, organized into modules.

[1]https://www.w3.org/TR/owl-ref/#Property

B IoT Functional View Platform Analysis

Following the functional view proposed in IOT-A, an analysis of the platforms under study has been done. In the next subsections, commonalities and discrepancies between platforms will be analyzed for each functional group, paying attention to the relevant functional components in IoT-A. To perform the study, each platform has been studied and determined if they provide at least one feature to cover partially or completely each of the functional components of the IOT-A. Then, the number of the FC-compliant has been aggregated, giving an idea of the availability of the features, also revealing what is considered important in the industry and academia, which has a relevance in order to prioritize functionalities when a great adoption is intended. This analysis aims at two main objectives, on one hand, an analysis of the so-identified market and research relevant platforms, which (as aforementioned) helps to find overall connection between different solutions and determine which ways of interoperability will be more effective and beneficial in the long term. On the other hand, the project has a limited scope and it will offer support (at integration-ready level) to a limited number of platforms as already stated in several points of this documents. The re-elaboration of the analysis with the initially supported platforms will give also an idea of which interoperability layers' mechanisms prioritize in order to provide early results to support pilots and third parties to join.

B.1 Application

The application comprises all those features that are domain/application specific and thus, they are out of the scope of the definition of the platform interoperability, as is defined in INTER-IoT. While in the study performed some aspects of the application FG were described (such as the domains where the platform operates or offer specific solutions, the conclusions of them are not relevant for the functional view or the development of a reference architecture to build interoperability mechanism between IoT platforms. For completeness, the prevalence of the domain aimed in the platforms under study is shown in the following histogram:
The analysis reveals that the domains with more support are healthcare, transport, home, city and parking. This shows the areas where the IoT is expanding faster (such as transport) or it has a smoother implantation (as in cities).

B.2 Management

The study of the Management functional components for the set of 16 platforms shows a clear predominance of the Configuration FC (implemented in 12 out of 16 platforms analyzed) and the Reporting FC (10/16). This shows that system initialization including the attached devices and the assessment of the overall performed are considered key pieces for an IoT platform. However, in general, the implementation of the IOT-A FCs of this group is high in the analyzed set, finding only a

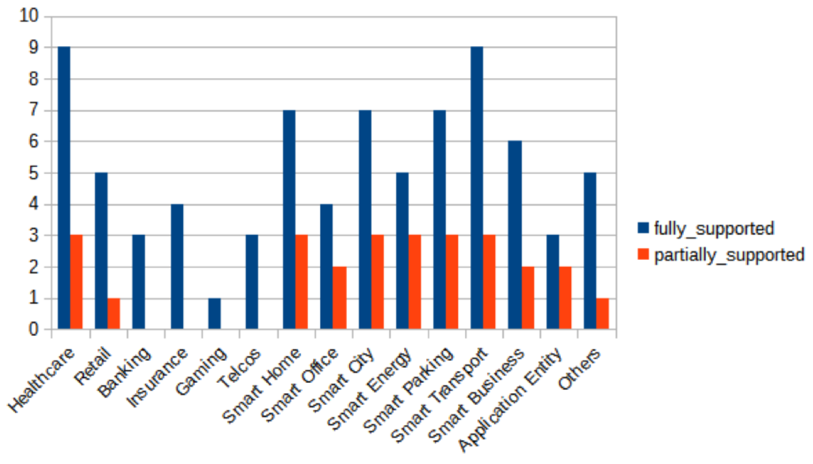

Figure B.1: Domain prevalence in studied platforms

significant lack of coverage on the State FC, which can be related to the legacy systems support that many of the platforms present. The legacy systems coming from home automation or telemetry often do not support queries or network info reporting, making virtually impossible to feature the state of the device networks. As a matter of the facts, those platforms that were born with a strong support of these communication protocol, do not care much about those features that were not available for their focus technologies. In the case of the selected platforms to be natively supported in INTER-IoT, all the FC similarly covered, being particularly remarkable, as in the case of the Configuration, which is implemented by all of them. In the rest of FCs, the 3/5 platform cover them, a prevalence similar to the results of the complete study.
For the purpose of the INTER-IoT Project, this means that, at management level, a common configuration interface could be implemented for the INTER-IoT user, as part of the framework planned. Other features would have compatibility with some but not all the initially supported platforms, so that the feasibility is limited and the decision of implementation depends strongly on the requirements of the project and the INTER-FW (see deliverables D2.3 and D2.4). In general, FIWARE, MS Azure and Open IoT are the most complete platforms in terms of system management among the selected platforms, while extending the scope to the complete study, GE Predix and Sofia2 also have full feature set.

B.3 Service Organisation

The service organization group, which evaluates the ability to manage services in the platforms is highly supported by all platforms, presenting figures of coverage about the 60% – 70% of each functional component. In this case, the degree of implementation is similar in the three components, being slightly more popular the service orchestration. Service organization is a concept highly bound to the service presence itself, so it can be observed in the detail that is very usual that platform with more upper layers implement at least two components of this group while more middleware-centric platforms (such as AllJoyn or OneM2M) do not implement any at all, since they are device focused in spite of the service focus of the former.
For the selection of platforms initially supported, the situation is different. OneM2M is a middleware

centric platform, so it does not give support to any kind of service composition, not including services in its domain model. The rest of platforms have a wider scope and support in some way operations and combinations with services. Consequently, it can be observed a coverage of 80% in orchestration and choreography. This FG is especially relevant for the AS2AS, as it gives a first idea of which platforms will be able to connect services and also provides an idea of the service interoperability mechanisms supported in the focus group.

B.4 IoT Process Management

This group has a significant lower coverage in all platforms, being present in around the 50% of platforms, regardless if it is analyzed the full set of platforms or the focus group.

The concept of IoT processes in IOT-A is related to the Business Management and how the IoT specific constraints are mapped there. As the definition of IoT Process is very specific and new, part of the platforms does not offer a particular solution for this characteristics, transferring the responsibility to the end user (using external services or custom logic). This is particularly true in the device or middleware centric platforms which provide more features in the lower layers. In the case of the focus group the situation is similar. Process modeling is supported in Azure, Open IoT and universAAL (with limitations in the last two, though) while process execution is supported only in Azure and OpenIoT. With the perspective of interoperability, this FG is not very relevant, since the business processes concerning two or more different platforms can be modeled (and executed) externally leveraging the already proposed AS2AS layer.

B.5 Virtual Entity

The Virtual Entity Functional Group has the mission of handle the relations between virtual entities and associated services, providing the needed mechanisms to discovering, updating and accessing to entity level services and features. In the global study, it has been found that the accomplishment of these features is high except for the VE & IoT Service. Monitoring component, which is probably the most complex of the three components identified in IOT-A for this FG. Therefore, it is declared to be implemented (or partially covered in less than the 50% of the analyzed platforms (6 out of 16). However, the VE Resolution and the VE service capabilities are supported in more than the 70% of platforms, which shows the relevancy of these components for the platforms.

For the focus group, the implementation of these components reaches higher levels, reaching the 80% in the case of VE Services component. In this case, the OneM2M platform makes the difference since it does not support virtual entity related components, despite the rest of the platforms of the subset. Virtual entities are a key concept in the INTER-IoT concept, architecture and framework. It is thanks to the entity virtualization that heterogeneous data can be harmonized, stored, transmitted and even translated into different ontologies. As explained in previous sections and in INTER-IoT Deliverable D3.1, released at the same time of this document, the concept is largely used at interoperability level, being a keystone for D2D and MW2MW. The high accomplishment of this functional group in the focus group and in the global study guarantees the viability of the solutions proposed in the Deliverable D3.1.

B.6 IoT Service

The IoT Service FG and its components are well covered in the IoT platforms, according to the study done. In this case, the study was one step further and analysed the prevalence of specific services, from a set of platform services typically present in sensor-related scenarios:

- Query information
- Update information
- Use resource operation/service
- Subscribe to information
- Subscription with filters
- Registration
- Historic data access
- CEP
- Big data storage
- Others
- IoT Client

While for the IoT Service Resolution, a list of features was also provided, based on the definition of IoT-A.

- Discovery
- Lookup
- Service Id. Resolution
- Service Descr. Mgmt.
- Others

In general terms, the service implementation levels are high, reaching the 100% or near in cases as the registration or the complex event processing (CEP). This also occurs in the IoT Service Resolution, which is supported in the 70% of platforms on average for the four specific features analyzed.
For the initially supported platforms, the conclusions are similar, showing again the lack of services implemented in OneM2M, much more centred in communications than in services. The IoT Service Resolution is also well covered by all the platforms with the known exception of OneM2M.
The following two histograms show the number of platforms that implement a version of the listed features for each of the FC related to IoT Services.

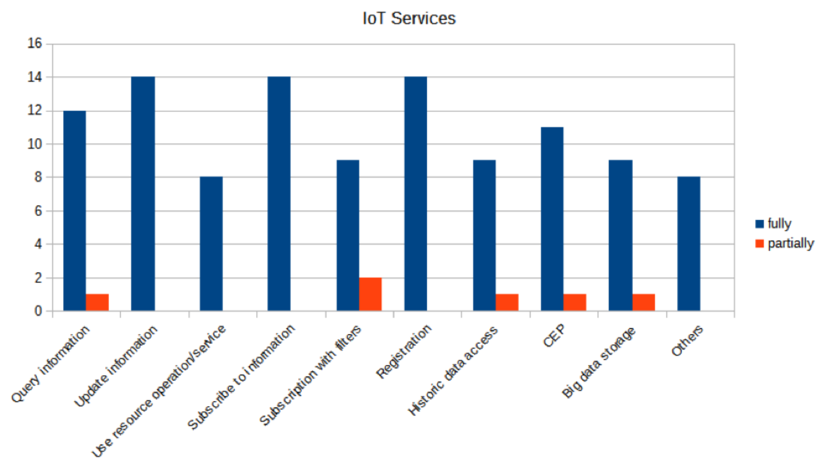

Figure B.2: IoT services implemented in the studied IoT platforms

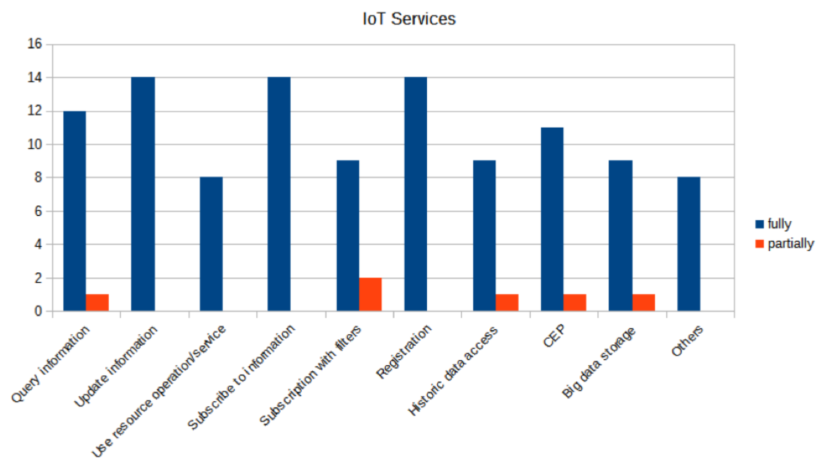

Figure B.3: IoT service resolution policies in the studied IoT platforms

B.7 Security

The Security FG is a transversal group that applies to all the rest of FGs per the IoT-A guidelines. With this in mind, a thorough analysis of the security capabilities was performed, assessing not only if each component is implemented somehow in every platform but also describing, when possible, the technologies used and more specific details related to the implementation of the components. The results show that the security is a common concern in the platforms analyzed. All the analyzed platforms implement more than one component of the identified in the Functional View. However, the strategies to accomplish the Security related operations differ between platform. While authorization and authentication are the preferred mechanisms (present in around 90% of the platforms), trust and reputation is significantly less prevalent with only the 30% of platforms implementing policies or components in this way.

For the focus group, results are similar, with a better support (practically full support) of the authorization and authentication components and a better coverage of the rest of components. The security FG is a concern of each interoperability layer and the INTER-IoT framework, which is in charge to coordinate and orchestrate all the security policies in order to maintain or improve the existing security standards in the platforms. According to the results obtained, the interoperability efforts here should go on the direction of ensuring and, when possible, centralizing the authentication and authorization in platforms.

B.8 Communication

The diversity of the device-to-device or device-to-gateway communications is one of the reasons of having so heterogeneous platforms. It is usually a starting point for creating a so-called information silo, since the lack of device interoperability with other devices or, more important, with other platforms usually ends in a domain/application specific deployment that forgets completely about interoperability due to the difficulties to achieve it. This is the closest FG to the physical level, and thus it should better be implemented in the device and middleware centric platforms. However, probably due to the reasons described previously, the implementation of the components in this group is still poor in the platforms analyzed. The most spread mechanism of communication supported is the end to end communication, while the network communication and the hop to hop communications are marginally covered.

The situation in the general study and in the case of the focus group is similar. With this results, it keeps clear that further efforts in communication standardization are needed. From the project point of view, the end to end communication is the best option to implement interoperability mechanisms, as devised in D2D layer (see Deliverable D3.1).

As mentioned, the range of different communications/standards/protocols to (mostly) physically transmit information from one device to a gateway, a dongle or another device is vast. The study considered this and analyzed the full set of possible communication in all the platforms reviewed, whose result is depicted in the figure B.4

It is obvious that this chart mixes very different communication modes, some of them compatible between them (e.g. it is possible to have Bluetooth and MQTT communications at the same time). The analysis has considered each supported communication regardless the OSI layer which is aimed at, to show on one hand the variety of possibilities and also to find the most popular device to platform communication methods that are supported in the cohort under study.

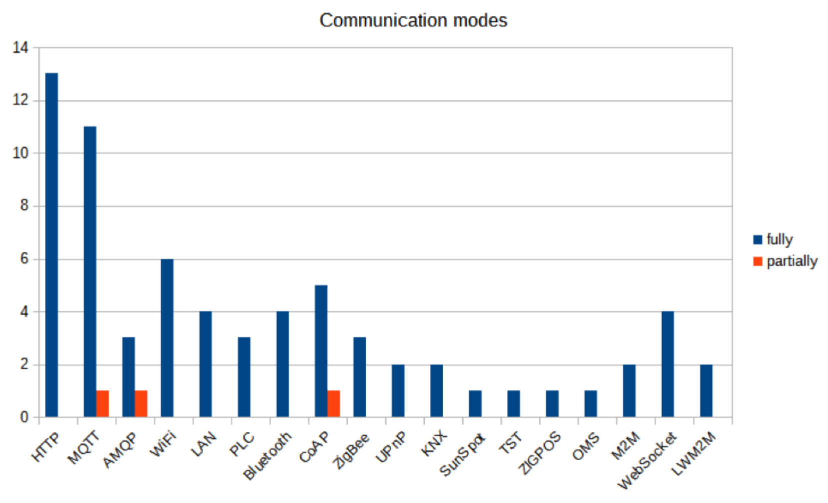

Figure B.4: Communication protocols at different layers supported by the platforms studied

B.9 Device

This group is considered out of the scope of this document and is not included in the analysis. However, since information about supported devices is offered, there has been created a histogram with the main groups of device groups and its support in the studied platforms.

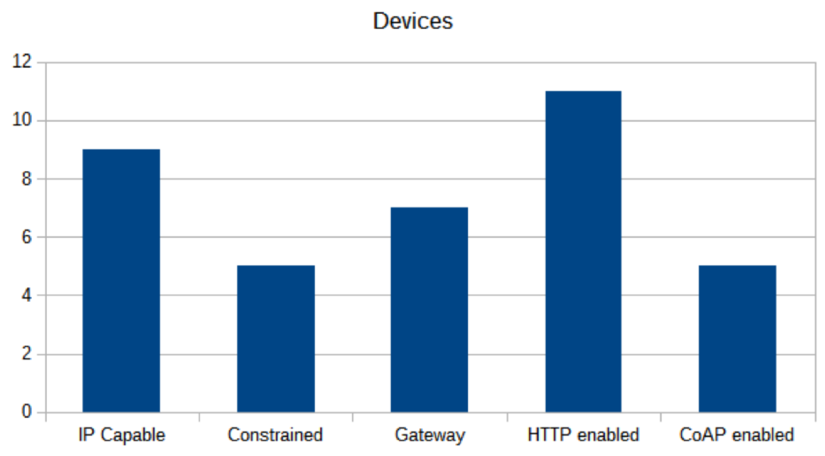

Figure B.5: Type of devices aimed by the platforms studied

Bibliography

[1] Rozanski, Nick and Wodds, Eoin, Applying Viewpoints and Views to Software Architecture http://www.viewpoints-and- perspectives.info/vpandp/wp- content/themes/secondedition/doc/VPandV_WhitePaper.pdf (accessed: 2013-05-08), 2005- 2013.

[2] Rozanski, Nick and Woods, Eoin. Software Systems Architecture Ð Working with Stakeholders Using Viewpoints and Perspectives, Addison Wesley, 2011.

[3] Shames, P. and Yamada, T. Reference architecture for space data systems. s.l. : DSpace at Jet Propulsion Laboratory [http://trs- new.jpl.nasa.gov/dspace-oai/request] (United States), 2004.

[4] IEEE Architecture Working Group, IEEE Std 1471-2000, Recommended practice for architectural description of software-intensive systems, 2000.

[5] Ganzha, Maria and Paprzycki, Marcin and Pawłowski, Wiesław and Szmeja, Paweł and Wasielewska, Katarzyna Towards Common Vocabulary for IoT Ecosystems—Preliminary Considerations Intelligent Information and Database Systems, 9th Asian Conference, ACIIDS 2017, Kanazawa, Japan, April 3-5, 2017, Proceedings, Part I

[6] Standard for an Architectural Framework for the Internet of Things (IoT) IEEE Standards https://standards.ieee.org/develop/project/2413.html

[7] Overview of the Internet of things ITU-T https://www.itu.int/rec/T-REC-Y.2060-201206-I

[8] Industrial Internet Reference Architecture v 1.8 Industrial Internet Consortium http://www.iiconsortium.org/IIRA.htm

[9] Functional Architecture Release 2 OneM2M http://www.onem2m.org/technical/published-documents

[10] Internet of Things - Architecture http://www.meet-iot.eu/iot-a-deliverables.html