

Internet of Things

Carlos E. Palau · Giancarlo Fortino ·
Miguel Montesinos · George Exarchakos ·
Pablo Giménez · Garik Markarian · Valérie Castay ·
Flavio Fuart · Wiesław Pawłowski · Marina Mortara ·
Alessandro Bassi · Frans Gevers ·
Gema Ibáñez-Sánchez · Ignacio Huet *Editors*

Interoperability of Heterogeneous IoT Platforms

A Layered Approach

 Springer

Internet of Things

Technology, Communications and Computing

Series Editors

Giancarlo Fortino, Rende (CS), Italy

Antonio Liotta, Edinburgh Napier University, School of Computing, Edinburgh,
UK

The series Internet of Things - Technologies, Communications and Computing publishes new developments and advances in the various areas of the different facets of the Internet of Things.

The intent is to cover technology (smart devices, wireless sensors, systems), communications (networks and protocols) and computing (theory, middleware and applications) of the Internet of Things, as embedded in the fields of engineering, computer science, life sciences, as well as the methodologies behind them. The series contains monographs, lecture notes and edited volumes in the Internet of Things research and development area, spanning the areas of wireless sensor networks, autonomic networking, network protocol, agent-based computing, artificial intelligence, self organizing systems, multi-sensor data fusion, smart objects, and hybrid intelligent systems.

** Indexing: *Internet of Things* is covered by Scopus and Ei-Compendex **

More information about this series at <https://link.springer.com/bookseries/11636>

Carlos E. Palau · Giancarlo Fortino ·
Miguel Montesinos · George Exarchakos ·
Pablo Giménez · Garik Markarian · Valérie Castay ·
Flavio Fuart · Wiesław Pawłowski ·
Marina Mortara · Alessandro Bassi · Frans Gevers ·
Gema Ibáñez-Sánchez · Ignacio Huet
Editors

Interoperability of Heterogeneous IoT Platforms

A Layered Approach

 Springer

Editors

Carlos E. Palau
School of Telecommunications Engineering
Universitat Politècnica de València
Valencia, Spain

Miguel Montesinos
Prodevelop, S.L.
Valencia, Spain

Pablo Giménez
Sede APV
Valencia, Spain

Valérie Castay
AFT
Paris, France

Wiesław Pawłowski
Faculty of Mathematics, Physics
and Informatics
University of Gdańsk
Gdańsk, Poland

Alessandro Bassi
ABC
Prague, Czech Republic

Gema Ibáñez-Sánchez
Instituto ITACA
Universitat Politècnica de Valencia
Valencia, Spain

Giancarlo Fortino
DIMES, University of Calabria
Rende (CS), Italy

George Exarchakos
Department of Electrical Engineering
Eindhoven University of Technology
Eindhoven, The Netherlands

Garik Markarian
Riverway House
Rinicom Ltd.
Lancaster, UK

Flavio Fuart
XLAB doo
Ljubljana, Slovenia

Marina Mortara
Dipartimento di Prevenzione, ASL TO5
Struttura Complessa Igiene degli Alimenti
Nichelino, Torino, Italy

Frans Gevers
Neways Technologies B.V.
EP Son, The Netherlands

Ignacio Huet
CSP Spain
Valencia, Spain

ISSN 2199-1073

Internet of Things

ISBN 978-3-030-82445-7

<https://doi.org/10.1007/978-3-030-82446-4>

ISSN 2199-1081 (electronic)

ISBN 978-3-030-82446-4 (eBook)

© Springer Nature Switzerland AG 2021

This work is subject to copyright. All rights are reserved by the Publisher, whether the whole or part of the material is concerned, specifically the rights of translation, reprinting, reuse of illustrations, recitation, broadcasting, reproduction on microfilms or in any other physical way, and transmission or information storage and retrieval, electronic adaptation, computer software, or by similar or dissimilar methodology now known or hereafter developed.

The use of general descriptive names, registered names, trademarks, service marks, etc. in this publication does not imply, even in the absence of a specific statement, that such names are exempt from the relevant protective laws and regulations and therefore free for general use.

The publisher, the authors and the editors are safe to assume that the advice and information in this book are believed to be true and accurate at the date of publication. Neither the publisher nor the authors or the editors give a warranty, expressed or implied, with respect to the material contained herein or for any errors or omissions that may have been made. The publisher remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

This Springer imprint is published by the registered company Springer Nature Switzerland AG
The registered company address is: Gewerbestrasse 11, 6330 Cham, Switzerland

Preface

In recent years, due to a great interest of both Industry and Academy in researching and developing IoT technology, many solutions at different levels (from the IoT device-level to full-fledged IoT platforms) have been implemented. However, there is no reference standard for IoT platform technology, and we do not foresee one in the near future. Hence, IoT scenarios will be characterized by a high degree of heterogeneity at all levels (device, networking, middleware, application service, data/semantics), preventing interoperability of IoT solutions. Although many research actions and projects have dealt and/or are dealing with developing IoT architectures in diversified application domains, not many of them have addressed interoperability/integration issues. Furthermore, no proposals (to date) have been put forward to deliver a general, application domain agnostic, fully reusable and systematic approach to solve multiple interoperability problems existing in the IoT platforms technology.

Lack of interoperability causes major technological and business issues such as impossibility to plug non-interoperable IoT devices into heterogeneous IoT platforms, impossibility to develop IoT applications exploiting multiple platforms in homogeneous and/or cross domains, slowness of IoT technology introduction at a large-scale, discouragement in adopting IoT technology, increase of costs, scarce reusability of technical solutions, user dissatisfaction. In contrast, interoperability among platforms will provide numerous benefits such as new market opportunities, the disappearance of vertical silos, and vertically oriented closed systems, architectures and application areas, to move towards open systems and platforms. Comprehensively addressing lack of interoperability in the IoT realm by proposing a full-fledged approach facilitating “voluntary interoperability” at any level of IoT platforms and across any IoT application domain, thus guaranteeing a seamless integration of heterogeneous IoT technology.

Most current existing sensor networks and IoT device deployments work as independent entities of homogeneous elements that serve a specific purpose, and are isolated from “the rest of the world”. In a few cases where heterogeneous elements are integrated, this is done either at device or network level, and focused mostly on unidirectional gathering of information. A multi-layered approach to integrating

heterogeneous IoT devices, networks, platforms, services and applications will allow heterogeneous elements to cooperate seamlessly to share data, infrastructures and services as in a homogeneous scenario.

INTER-IoT is a solution proposed to the above problem, and has aimed at the design, implementation and experimentation of an open cross-layer framework, an associated methodology and tools to enable voluntary interoperability among heterogeneous Internet of Things (IoT) platforms. INTER-IoT is the supporting environment for this book. It has been conceived and created among other potential solutions in the framework of IoT-EPI (IoT European Platforms Initiative), and has allowed effective and efficient development of adaptive, smart IoT applications and services, atop different heterogeneous IoT platforms, spanning single and/or multiple application domains, creating also its own ecosystem.

This book investigates on the multi-layered approach to achieve semantic interoperability, presenting innovative solutions for the architecture and the individual layers so as management and the methodological approach. Readers are offered with new issues and challenges in a continuously moving environment like IoT platform interoperability. In particular, the book spans the following scenarios: (1) port transportation and logistics; (2) mobile healthcare and (3) different application domains related with the INTER-IoT ecosystem. All the areas covered by the interoperability solution and its application correspond to ten authored chapters briefly introduced below.

Chapter “[Introduction to Interoperability for Heterogeneous IoT Platforms](#)” by Carlos E. Palau, et al., presents an overview of the needs, potential solutions and advances regarding IoT platforms interoperability. The chapter in particular starts reviewing the existing solutions and state of the art associated with platform interoperability and discusses the benefits of a multi-layered approach solution analysing the layers selected for the INTER-IoT solution and the potential application to two selected use cases, identifying the uniqueness of the provided approach.

Chapter “[INTER-IoT Requirements](#)” by Pablo Giménez, Miguel Llop, Regal Gonzalez-Usach, and Miguel A. Llorente proposes the main requirements and the process to gather them to achieve interoperability between IoT platforms. The chapter considers the Volere methodology as the mechanism to define, gather, select and prioritize the functional and non-functional requirements to develop an interoperable solution. Requirements are analysed following different approaches and can be used as support for potential developers that may need to perform a similar analysis for the same or different application domains.

Chapter “[INTER-IoT Architecture for Platform Interoperability](#)” by Alessandro Bassi, Miguel A. Llorente, Miguel Montesinos, and Raffaele Gravina analyses the need of a meta-architecture with a specific domain model to define the different building blocks for an interoperable solution. The chapter illustrates the contribution of INTER-IoT for the definition of a reference architecture, using IoT-A as a starting point and the link with further developments related with the IoT community. The chapter includes the software vision of the architecture and supports the multi-layered approach which is the current approach required from the market.

Chapter “[INTER-Layer: A Layered Approach for IoT Platform Interoperability](#)” by Andreu Belsa et al., describes the different solutions for each of the layers that compose the architecture. The chapter starts with a detailed state-of-the-art analysis and describes the different technologies that can be used to provide interoperability at each layer of the architecture. The technical aspects of the developments and integration are based on the requirements gathered and described in Chapter “[INTER-IoT Requirements](#)”. The cross-layer needs of the architecture are also analysed in the chapter with a specific focus on security, privacy, reliability and management.

Chapter “[Semantic Interoperability](#)” by Maria Ganzha, et al., concerns modern trends in IoT semantic interoperability, in particular the different methods to be used, with a specific focus on semantic alignment. After an overview of current literature, the chapter defines the different semantic interoperability patterns, a global ontology (GOIoTP) for platform interoperability, that agnostically address any domain. The chapter describes a key component of the architecture as the IoT Platform Semantic Mediator (IPSM) that support semantic interoperability functions at any layer but mainly at middleware and application and service layers.

Chapter “[INTER-Framework: An Interoperability Framework to Support IoT Platform Interoperability](#)” by Clara I. Valero et al., considers the different tools required to manage, secure and provide global access to the APIs of the architecture. After a brief discussion on related work the chapter describes INTER-API and the global API of the INTER-IoT architecture as a relevant contribution and innovation in the area of IoT interoperability; the management functions that allow fast configuration of the interoperability parameters at any layer and the security measures in order to achieve and guarantee interoperability highlighting scalability, flexibility and sustainability.

Chapter “[INTER-Meth: A Methodological Approach for the Integration of Heterogeneous IoT Systems](#)” by Giancarlo Fortino et al., provides support for the integration of heterogeneous IoT platforms from the analysis to the maintenance phase, something that is required due to the lack of proper interoperability standards. The chapter provides a description using software engineering of a methodological approach to achieve and manage interoperability among IoT platforms avoiding dependency on the application domain. The proposed methodology is supported by software tools that help the different actors, following their different profile in configuring every required enabler and component.

Chapter “[Interoperability Application in e-Health](#)” by Gema Ibáñez-Sánchez, Alvaro Fides-Valero, Jose-Luis Bayo-Monton, Margherita Gulino, and Pasquale Pace is a chapter where the different proposals of the previous chapters, from requirements elicitation till interoperability achievement, are analysed and deployed in the application domain of mobile health. INTER-HEALTH provides a solution that allows health experts to prevent and reduce obesity, which is one of the main causes of chronic diseases. Through INTER-IoT, two different platforms (i.e. UniversAAL and BodyCloud) are able to interoperate to exchange information and so, to provide aggregated information to health experts. The results show a clear improvement in the health of the participants compared to those not using it.

Chapter “[INTER-LogP: INTER-IoT for Smart Port Transportation](#)” by Pablo Giménez, Miguel Llop, Joan Meseguer, Fernando Martin, and Antonio Broseta explores the application of the methodology, including requirements gathering and implementation of the different components in a complex environment like a port. The chapter considers the interaction of several IoT platforms with the goal of sharing data and services, provided by the port authority of Valencia, the NOATUM container terminal and other IoT platforms provided by third parties. With the data provided, three different scenarios were defined, and showed the benefits of sharing data in the port and the logistic sector: access control and traffic, dynamic lighting and wind gusts detection.

Chapter “[IoT Ecosystem Building](#)” by Regel Gonzalez-Usach, Carlos E. Palau, Miguel A. Llorente, Roel Vossen, Rafael Vaño, and Joao Pita analyses the mechanism to extend the developments of an IoT Open Source project. The chapter describes the methodology and new actors associated with the interoperability framework defined in the previous chapters. A detailed description of different projects is associated with INTER-IoT that validated different enablers, components and methods of the proposed interoperability approach with the aim of sustainability and extendibility.

Our gratitude is for all chapter contributors, the reviewers, and for the Editorial Board from Springer for their support and work during the publication process.

Valencia, Spain	Carlos E. Palau
Rende (CS), Italy	Giancarlo Fortino
Valencia, Spain	Miguel Montesinos
Eindhoven, The Netherlands	George Exarchakos
Valencia, Spain	Pablo Giménez
Lancaster, UK	Garik Markarian
Paris, France	Valérie Castay
Ljubljana, Slovenia	Flavio Fuat
Gdańsk, Poland	Wiesław Pawłowski
Nichelino, Italy	Marina Mortara
Prague, Czech Republic	Alessandro Bassi
EP Son, The Netherlands	Frans Gevers
Valencia, Spain	Gema Ibáñez-Sánchez
Valencia, Spain	Ignacio Huet

Contents

Introduction to Interoperability for Heterogeneous IoT Platforms	1
Carlos E. Palau, Giancarlo Fortino, Miguel Montesinos, Pablo Giménez, Garik Markarian, Valérie Castay, Flavio Fuart, Wiesław Pawłowski, Marina Mortara, Alessandro Bassi, Frans Gevers, Gema Ibáñez-Sánchez, Ignacio Huet, and George Exarchakos	
INTER-IoT Requirements	27
Pablo Giménez, Miguel Llop, Regel Gonzalez-Usach, and Miguel A. Llorente	
INTER-IoT Architecture for Platform Interoperability	49
Alessandro Bassi, Miguel A. Llorente, Miguel Montesinos, and Raffaele Gravina	
INTER-Layer: A Layered Approach for IoT Platform Interoperability	95
Andreu Belsa, Alejandro Fornes-Leal, Clara I. Valero, Eneko Olivares, Jara Suárez de Puga, Fernando Boronat, and Flavio Fuart	
Semantic Interoperability	133
Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Bartłomiej Solarz-Niesłuchowski, Paweł Szmeja, and Katarzyna Wasielewska	
INTER-Framework: An Interoperability Framework to Support IoT Platform Interoperability	167
Clara I. Valero, Andreu Belsa, Alejandro Fornes-Leal, Fernando Boronat, Miguel A. Llorente, and Miguel Montesinos	
INTER-Meth: A Methodological Approach for the Integration of Heterogeneous IoT Systems	195
Giancarlo Fortino, Raffaele Gravina, Wilma Russo, Claudio Savaglio, Katarzyna Wasielewska, Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, and Rafał Tkaczyk	

Interoperability Application in e-Health 231
Gema Ibáñez-Sánchez, Alvaro Fides-Valero, Jose-Luis Bayo-Monton,
Margherita Gulino, and Pasquale Pace

INTER-LogP: INTER-IoT for Smart Port Transportation 257
Pablo Giménez, Miguel Llop, Joan Meseguer, Fernando Martin,
and Antonio Broseta

IoT Ecosystem Building 279
Regel Gonzalez-Usach, Carlos E. Palau, Miguel A. Llorente,
Roel Vossen, Rafael Vaño, and Joao Pita

Introduction to Interoperability for Heterogeneous IoT Platforms



Carlos E. Palau, Giancarlo Fortino, Miguel Montesinos, Pablo Giménez, Garik Markarian, Valérie Castay, Flavio Fuart, Wiesław Pawłowski, Marina Mortara, Alessandro Bassi, Frans Gevers, Gema Ibáñez-Sánchez, Ignacio Huet, and George Exarchakos

C. E. Palau (✉)

DCOM, Universitat Politècnica de València, Camino de Vera, 46022 Valencia, Spain
e-mail: cpalau@com.upv.es

G. Fortino

DIMES, Università della Calabria, Via Pietro Bucci, 87036 Arcavacata, Rende CS, Italy

M. Montesinos

PRODEVELOP S.L., Carrer del Cronista Carreres, 13, Valencia, Spain

P. Giménez

Fundación Valenciaport, Avinguda Moll del Turia, s/n, 46024 Valencia, Spain

G. Markarian

RINICOM Ltd, Morecambe Road, Lancaster LA1 2RX, UK

V. Castay

Département des Etudes et Projets, AFT-DEV, 82 rue Cardinet, 75845 Paris, France

F. Fuart

XLAB doo, Pot za Brdom 100, SI-1000 Ljubljana, Slovenia

W. Pawłowski

Faculty of Mathematics, Physics and Informatics, University of Gdańsk, Gdańsk, Poland
e-mail: wieslaw.pawlowski@ug.edu.pl

M. Mortara

Dipartimento di Prevenzione, ASL TO5, Via San Francesco d'Assisi, 35., 10042 Nichelino (TO), Italy

A. Bassi

ABC, Ruska 50, 101 00 Prague, Czech Republic

F. Gevers

Neways Technologies B.V., Science Park Eindhoven 5709, The Netherlands

G. Ibáñez-Sánchez

ITACA, Universitat Politècnica de València, Camino de Vera, 46022 Valencia, Spain

I. Huet

CSP Spain, C/ Menorca, 19 - Edificio Aqua, 46023 Valencia, Spain

G. Exarchakos

Department of Electrical Engineering, Eindhoven University of Technology, 5600 MB Eindhoven, The Netherlands

© Springer Nature Switzerland AG 2021

C. E. Palau (eds.), *Interoperability of Heterogeneous IoT Platforms*, Internet of Things, https://doi.org/10.1007/978-3-030-82446-4_1

Abstract INTER-IoT presents a novel layer-oriented solution for interoperability, to provide interoperability at any layer and across layers among different IoT systems and platforms. Contrary to a more general global approach, the INTER-IoT layered approach has a higher potential in order to provide interoperability. It facilitates a tight bidirectional integration, higher performance, complete modularity, high adaptability and flexibility, and presents increased reliability. This layer-oriented solution is achieved through INTER-LAYER, several interoperability solutions dedicated to specific layers. Each interoperability infrastructure layer has a strong coupling with adjacent layers and provides an interface. Interfaces will be controlled by a meta-level framework to provide global interoperability. Every interoperability mechanism can be accessed through an API. The interoperability infrastructure layers can communicate and interoperate through the interfaces. This cross-layering allows to achieve a deeper and more complete integration.

1 Introduction

The connection of intelligent devices, equipped with a growing number of electronic sensors and/or actuators, via the Internet, is known as the Internet of Things (IoT). With the IoT, every physical and virtual object can be connected to other objects and to the Internet, creating a fabric of connectivity between things and between humans and things [1, 2]. The IoT is now widely recognised as the next step of disruptive digital innovation.

The International Communications Union (ITU) and the European Research Cluster on the Internet of Things (IERC) provide the following definition: IoT is a dynamic global network infrastructure, with self-configuring capabilities based on standard and interoperable communication protocols, where physical and virtual things have identities, physical attributes and virtual personalities and use intelligent interfaces. All of them seamlessly integrated into the information network [3].

The design of the Internet and specifically the extension of the Internet to the IoT, rely on the convergence of the infrastructure with software and services. A common practice is required to think/design cross solutions between software and infrastructure in order to provide integrated solutions for some of the complex problems in the current and future systems. In the IoT environment this convergence is evident, and the continuous evolution generates more and more smart connected objects and platforms that are embedded with sensors and their respective associated services, in some cases considering virtualization.

IoT is the network or overlay associations between smart connected objects (physical and virtual), that are able to exchange information by using an agreed method (including protocols) and a data schema. IoT deployments are increasing, the same applies to standards, alliances and interest for homogenization. All of this is giving a strong push to the IoT domain to be considered as one of the most promising emerging technologies. As an example, Gartner (one of the world's leading information technology research and advisory company) estimates the number of web-connected

devices will reach 25 billion by 2020. In other words, more devices, appliances, cars, artefacts, and accessories will be connected and will communicate with each other, and with other objects, thus bringing amplified connectivity and better supply chain visibility. The applications of the IoT are numerous i.e. every object could be transformed into a smart object that sends several valuable information to other devices. As an example, in the port industry IoT could be applied to shipping containers, the equipment that handles them, the trucks that carry them and, even, the ships that move them around the globe [4].

According to the European Commission (EC) the IoT represents the next step towards the digitisation of our society and economy, where objects and people are interconnected through communication networks, and report about their status and/or the surrounding environment. Furthermore, IoT can also benefit the European economy generating economic growth and employment; according to a recent European Commission study revenues in the EU28 will increase from more than €307 billion in 2013 to more than €1,181 billion in 2020 [3, 4].

IoT is an emerging area that not only requires development of infrastructure but also deployment of new services capable of supporting multiple, scalable and interoperable applications. The focus is today associated with cloud deployments, virtualizations and the elimination of silos avoiding the existence of application domain specific developments, AIOTI and EC are pressing in this line. IoT has evolved from sensor networks and wireless sensor networks to a most clear description and definition referring to objects and the virtual representations of these objects on the Internet and associated infrastructures. It defines how the physical things and virtual objects will be connected through the Internet and their interaction, and how they communicate with other systems and platforms, in order to expose their capabilities and functionalities in terms of services and accessibility through open APIs and frameworks. IoT is not only linking connected devices by the Internet; it is also web-enabled data exchange in order to enable systems with more capacities to become smart and accessible, creating webs of objects and allowing integration of data, services and components [5].

There are several challenges associated with IoT and its evolution, but one major issue is related with interoperability [6–8]. IoT is mainly supported by continuous progress in wireless sensor and actuator networks and by manufacturing low cost and energy efficient hardware for sensor and device communications. However, heterogeneity of underlying devices and communication technologies and interoperability in different layers, from communication and seamless integration of devices to interoperability of data generated by the IoT resources, is a challenge for expanding generic IoT solutions to a global scale, with the further aim of avoiding silos and provide solutions that are application domain agnostic, like those proposed in INTER-IoT and that will be reflected in the rest of the book [9].

2 INTER-IoT at a Glance

Achieving interoperability is one of the main objectives of the IoT. It is all about connecting things and make them easily accessible just like the Internet today. Broadly speaking, interoperability can be defined as a measure of the degree to which diverse systems, organizations, and/or individuals are able to work together to achieve a common goal” [6]. However, interoperability is a complex thing and there are many aspects to it. In literature, there exists quite a lot of different classifications of these aspects of interoperability, often also called levels of interoperability. One of the most important classification of levels of interoperability for technical systems is called Levels of Conceptual Interoperability Model (LCIM). It defines six levels of interoperability: technical, syntactic, semantic, pragmatic, dynamic and conceptual interoperability. INTER-IoT follows a similar layered structure, however the approach has been different in terms of identification of the layers.

INTER-IoT as a whole has been the result of a Research and Innovation Action under H2020 EC Framework Programme. The project has designed, implemented and experimented with an open cross-layer framework, an associated methodology and tools to enable voluntary interoperability among heterogeneous Internet of Things (IoT) platforms (all these components will be reflected in the next chapters of this book) [10]. The proposal has allowed effective and efficient development of adaptive, smart IoT applications and services, atop different heterogeneous IoT platforms, spanning single and/or multiple application domains. The project will be tested in two application domains: transport and logistics in a port environment and mobile health, additionally it will be validated in a cross-domain use case supported by the integration in the project of twelve third parties. The INTER-IoT approach is general-purpose and may be applied to any application domain and across domains, in which there is a need to interconnect IoT systems already deployed or add new ones. Additionally, INTER-IoT is one of the seven RIAs and two CSA composing IoT-EPI, supporting the creation of a European common space for IoT interoperability [11–13].

INTER-IoT is based on three main building blocks, with different subcomponents that have been identified and classified in different exploitable products adequate to the needs of the different stakeholders involved in the project and also addressing the main needs of the potential customers of the entities participating in INTER-IoT. This three main building blocks, that will be further explained in the following chapters of the book are:

- **INTER-LAYER:** methods and tools for providing interoperability among and across each layer (virtual gateways/devices, network, middleware, application services, data and semantics) of IoT platforms. Specifically, we will explore real/virtual gateways, for device-to-device communication, virtual switches based on SDN for network-to-network interconnection, super middleware for middleware-to-middleware integration, service broker for the orchestration of the service layer and a semantics mediator for data and semantics interoperability [11].

- INTER-FW: a global framework (based on an interoperable meta-architecture and meta-data model) for programming and managing interoperable IoT platforms, including an API to access INTER-LAYER components and allow the creation of an ecosystem of IoT applications and services. INTER-FW will provide management functions specifically devoted to the interconnection between layers. The provided API includes security and privacy features and will support the creation of a community of users and developers [14].
- INTER-METH: an engineering methodology based on CASE (Computer Aided Software Engineering) tool for systematically driving the integration and interconnection of heterogeneous non-interoperable IoT platforms [15, 16].

INTER-IoT provides an interoperable mediation component (i.e. INTER-LAYER) to enable the discovery and sharing of connected devices across existing and future IoT platforms for rapid development of cross-platform IoT applications. INTER-IoT allows flexible and voluntary interoperability at different layers. This layered approach can be achieved by introducing an incremental deployment of INTER-IoT functionality across the platform's space, which will in effect influence the level of platform collaboration and cooperation with other platforms. INTER-IoT does not pretend to create a new IoT platform but an interoperability structure to interconnect different IoT platforms, devices, applications and other IoT artifacts [11, 17].

Syntactic and semantic interoperability represent the essential interoperability mechanisms in the future INTER-IoT ecosystem, while organizational/enterprise interoperability has different structures/layers to enable platform providers to choose an adequate interoperability model for their business needs. It will be supported by INTER-FW that may allow the development of new applications and services atop INTER-LAYER and INTER-METH, to provide a methodology in order to coordinate interoperability supported by the definition of different interoperability patterns and a CASE tool [16] (Fig. 1).

INTER-LAYER, which will be addressed in detail in Chap. 4, is composed by five layers, supported by cross-layer components as needed for the interaction of the different layers:

- Device layer (D2D): At the device level, D2D solution will allow the seamless inclusion of novel IoT devices and their interoperability with already existing ones. D2D solution is a modular gateway that supports a vast range of protocols as well as raw forwarding. It is composed on a physical part that only handles network access and communication protocols, and a virtual part that handles all other gateway operations and services (gw virtualization). When connection is lost, the virtual part remains functional and is capable to answer the API and Middleware requests. The gateway follows a modular approach to allow the addition of optional service blocks to adapt to the specific case, allowing a fast growth of smart objects ecosystems [18, 19].
- Network layer (N2N): N2N solution enables seamless Network-to-Network interoperability, allowing transparent smart object mobility, and information routing support. It will also allow offloading and roaming, what implies the interconnection of gateways and platforms through the network. Interoperability is achieved

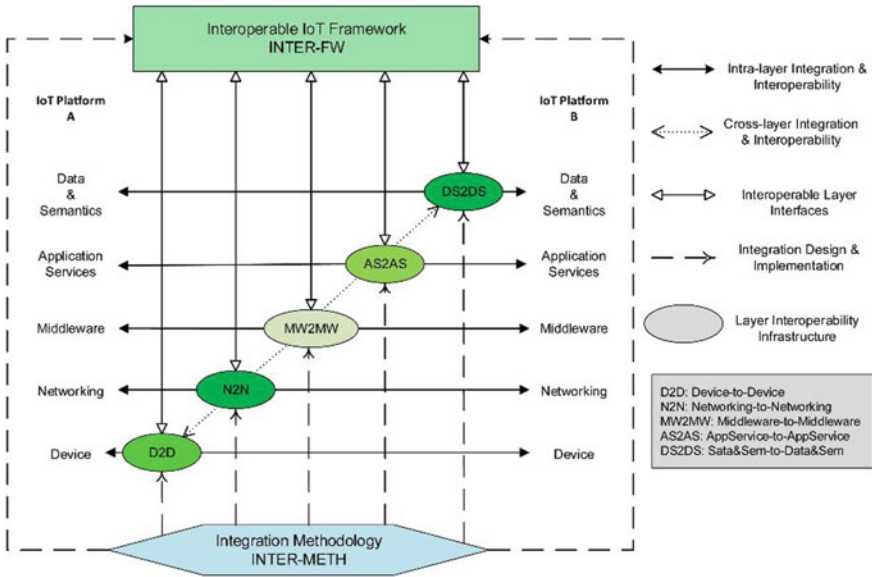


Fig. 1 INTER-IoT concept and vision

through the creation of a virtual network, using SDN and NFV paradigms, with the support of the N2N API. The N2N solution will allow the design and implementation of fully interconnected ecosystems, and solve the smart object mobility problem [20].

- **Middleware layer (MW2MW):** At the middleware level INTER-IoT solution will enable seamless resource discovery and management system for the IoT devices in heterogeneous IoT platforms. Interoperability at the middleware layer is achieved through the establishment of an abstraction layer and the attachment of IoT platforms to it. Different modules included at this level provide services to manage the virtual representation of the objects, creating the abstraction layer to access all their features and information. Those services are accessible through a general API. Interoperability at this layer will allow a global exploitation of smart objects in large scale multi-platform IoT systems [21].
- **Application and Services layer (AS2AS):** INTER-IoT will enable the use of heterogeneous services among different IoT platforms. Our approach will allow the discovery, catalogue, use and even composition of services from different platforms. AS2AS will also provide an API as an integration toolbox to facilitate the development of new applications that integrate existing heterogeneous IoT services [22].
- **Data and Semantics level (DS2DS):** INTER-IoT guarantees a common interpretation of data and information among different IoT platforms and heterogeneous data sources that typically employ different data formats and ontologies, and are unable to directly share information among them. INTER-IoT DS2DS approach

is the first solution that provides universal semantic and syntactic interoperability among heterogeneous IoT platforms. It is based on a novel approach, a semantic translation of IoT platforms' ontologies to/from a common Central Ontology that INTER-IoT employs, instead of direct platform-to-platform translations. This technique reduces dramatically the number of potential combinations of semantic translations required for universal semantic interoperability. INTER-IoT semantic interoperability tools work with any vocabulary, or ontology. INTER-IoT own modular Central Ontology, called GOIoTP, for all IoT platforms, devices and services, is available at <http://docs.inter-iot.eu/ontology>. Also, syntactic translators allow interoperability between different data formats, such as JSON, XML, and others. Although the pilot deployments of INTER-IoT realize the Core Information Model with Extensions approach to semantic interoperability, INTER-IoT supports any solutions between its pilot approach and Core Information Model [23, 24].

- **Cross-Layer:** INTER-IoT also guarantees non-functional aspects that must be present across all layers: trust, security, privacy, and quality of service (QoS). As well, INTER-IoT provides a virtualized version of the solution for each layer, to offer the possibility of a quick and easy deployment. Security is guaranteed inside each individual layer, and the external API access is securitized through encrypted communication, authentication and security tokens. INTER-IoT accomplishes the new European Data Privacy Law, and in the specific case of e-Health, in which information is highly sensitive, the Medical Device Regulation law [11, 25].

And INTER-FW which provides the wrapping environment for INTER-LAYER component coordination and new services development using INTER-API. Open interoperability delivers on the promise of enabling vendors and developers to interact and interoperate, without interfering with anyone's ability to compete by delivering a superior product and experience. In the absence of global IoT standards, the INTER-IoT project will support and make it easy for any company to design IoT devices, smart objects, or services and get them to market quickly, and create new IoT interoperable ecosystems. INTER-IoT may provide a solution to any potential interoperability problem within the IoT landscape [13] (Fig. 2).

3 INTER-IoT Use Case-Driven

The INTER-IoT approach is use case-driven, implemented and tested in three realistic large-scale pilots: (i) Port of Valencia transportation and logistics involving heterogeneous platforms with 400 smart objects; (ii) an Italian National Health Center for m-health involving 200 patients, equipped with body sensor networks with wearable sensors and mobile smart devices and (iii) a cross domain pilot involving IoT platforms from different application domains and enlarged by the collaboration of the solutions associated to the different layers and sublayers from the third parties that have attended the open call. The use cases are:

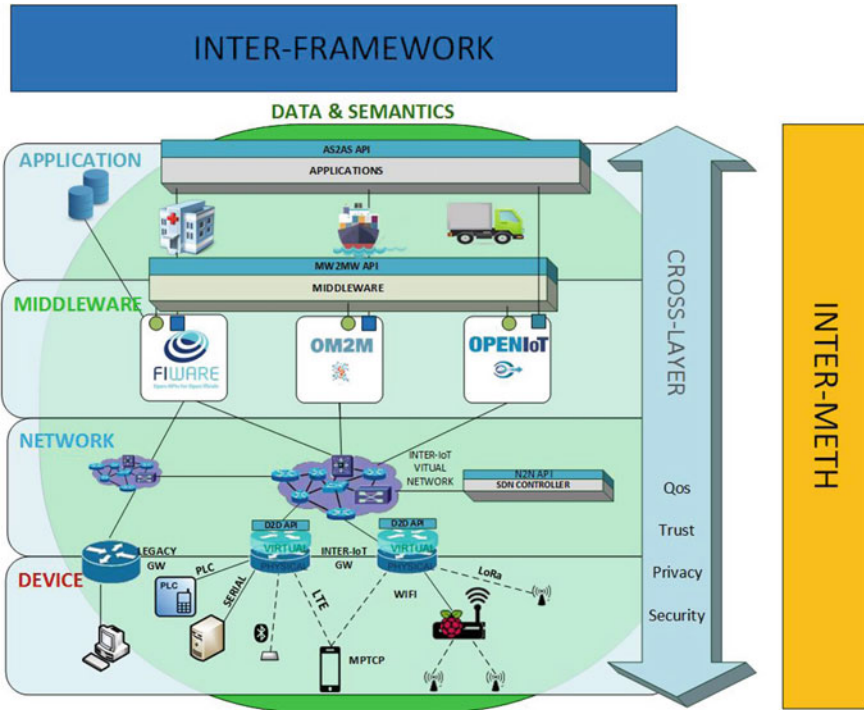


Fig. 2 INTER-IoT layered approach

- **INTER-LogP:** The use of IoT platforms in the ports of the future will enable locating, monitoring, and handling different transport and cargo equipment and storage areas. This use case will address the need to seamlessly handle IoT platforms interoperation within port premises: container terminal, transportation companies, warehouses, road hauliers, port authorities, customs, and outside the port [26].
- **INTER-Health:** The Decentralized and Mobile Monitoring of Assisted Livings' Lifestyle use case, aims to develop an integrated IoT system for monitoring humans' lifestyle in a decentralized mobile way to prevent chronic diseases. The aforementioned monitoring process can be decentralized from the health-care center to the monitored subjects' homes, and supported in mobility by using on-body physical activity monitors [27].
- **INTER-DOMAIN,** composed by IoT platforms from the two application domain oriented pilots and the IoT platforms and the specific layer-oriented solutions from different application domains selected in the open call. SENSINACT and OM2M platforms with Smart Cities orientation have been selected, and contributions from the different layers may complement INTER-IoT [28, 29].

The project has analyzed requirements provided by the stakeholders of the project and usability of the provided solutions from the perspective of IoT platform creators,

IoT platform owners, IoT application programmers and users investigating business perspectives and creating new business models. These results have allowed to start INTER-IoT ecosystem and new features and components: methodologies, tools, protocols and API. The variety and cross availability of the results could be used to build and integrate services and platforms at different layers according to the needs of the stakeholders and developers. The availability of more and new data will stimulate the creation of new opportunities and products.

3.1 INTER-LogP: Interoperability for Transport and Logistics in a Port Environment

In the ports of the future, port users, equipment and infrastructures will achieve a zero distance interaction offering more sustainable transport solutions. The use of IoT platforms will enable locating, monitoring, and handling different transport and cargo equipment and storage areas. The requirements for a better management of equipment and resources and the huge complexity of interactions involving large quantity of simultaneous transport movements around big logistics nodes (e.g., container terminals, ports, warehouses and dry ports) originates the need to introduce IoT platforms with multiple sensors in all logistics elements to control and monitor the several operations like energy consumption, gas emissions, or machine status. With these platforms, logistics service providers will be able to monitor and control in real time all the operations and movements of freight, equipment, vehicles and drivers on logistics nodes.

The Port of Valencia premises extend for several square kilometres. It is the largest Mediterranean port in terms of container handling. The port contains five container terminals (e.g., NOATUM and MSC), and several other facilities (e.g., train freight station, warehouses, and parking spaces). The port includes several kilometres of road within the premises. The Port Authority has several deployed IoT platforms connected to different HMI and SCADA with different goals (e.g., traffic management, security, safety and environmental protection, or vessels identification). Some of these platforms provide selected data to the Port Community System (PCS) like tamper proof RFID tags and e-seals that are installed on trucks and semi-trailers. In particular, A Port Community System is an electronic platform that connects the multiple systems operated by a variety of organisations that make up a seaport, airport or inland port community. It is shared in the sense that it is set up, organised and used by firms in the same sector—in this case, a port community. There is an increasing need that trucks, vehicles and drivers seamlessly interoperate with the port infrastructures and vice versa. All deployed IoT platforms do not interoperate as they are based on different standards, and remain isolated with a clear lack of interoperability at all layers.

NOATUM Container Terminal is one of the biggest container terminals in the Mediterranean located at the port of Valencia. It is the fifth largest European port in

container handling, i.e. it deals with more than 50,000 movements per day, produced by more than 200 container handling units (e.g., cranes, forklifts, RTGs, internally owned tractors and trailers, etc.); more than 4,000 trucks and other vehicles visit terminal premises; with more than 10,000 containers involved in these movements. These values show the complexity of this environment and the opportunities that the information compiled by the sensors installed on the equipment, trucks and containers; and the IoT interconnected architectures can bring to the terminal (e.g., in terms of optimization in the operations, safety, security or cost and energy savings). Container terminals like the one managed by the NOATUM have a huge number of sensors, CPS (Cyber Physical Systems) and smart objects; fixed and mobile deployed and exchanging information within one or between several platforms deployed in their premises. The sensors from the internal equipment (i.e., container terminal IoT ecosystem), constitute 5% of total vehicles moving daily within terminal premises, and they generate more than 8,000 data units per second. The other 95% of the vehicles are external trucks and other vehicles, with sensors belonging to other IoT ecosystems, currently unable to interact with the terminal IoT solution. Additionally, containers (mainly used to transport controlled temperature cargoes) have their own IoT architecture, which cannot be accessed by the terminal, when the container is stored in the yard or moved across it. This lack of interoperability of outdoor ambulatory IoT things based on heterogeneous architectures represents a big barrier that INTER-IoT aims at removing.

This use case illustrates the need to seamlessly IoT platforms interoperation within port premises, e.g., container terminal, transportation companies, warehouses, road hauliers, port authorities, customs, border protection agencies, and outside the port. Port IoT ecosystems use to be operated by a large number of stakeholders, and typically require high security and trust, due to mobility and seamless connectivity requirements, that currently are not available with the exception of proprietary and isolated solutions. Introduction of interoperability mechanisms and tools will therefore bring about new solutions and services leading to developments of the ports of the future.

3.1.1 INTER-IoT Approach to INTER-LogP

INTER-LogP will be an INTER-IoT outcome to facilitate interoperability of heterogeneous Port Transport and Logistics-oriented IoT platforms already in place, i.e., VPF and NOATUM and other components that will be brought to the use case in order to achieve the INTER-IoT proposed goals, e.g., I3WSN from UPV and other IoT platforms from companies operating in the Port managed premises.

The Port Authority of Valencia will provide its own IoT platform ecosystem to the project, including (i) the climate and weather forecast infrastructures, which monitor the environmental conditions in real-time and maintain historical data; (ii) beacon data acquisition system, which monitors and controls whenever necessary all the buoys distributed on the sea side; (iii) PCS-IoT platform, developed to cover different transportation and logistics components throughout the port premises, integrates an

internal communication network and connects (more than 400) operating companies in the port.

NOATUM provides the SEAMS platform to be included in the INTER-LogP use case. SEAMS is an outcome from the Sea Terminals action (Smart, Energy Efficient and Adaptive Port Terminals) co-funded by the Trans-European Transport Network (TEN-T). It is an operational tool based on the reception of real-time energy and operative data coming from the whole machinery and vehicle fleets of NOATUM Container Terminal Valencia (NCTV). SEAMS integrates the whole set of machines (including Rubber Tyre Gantry cranes (RTG), Ship-To-Shore cranes (STS), Terminal Tractors (TT), Reach Stackers (RS) and Empty Container Handlers (ECH)) and vehicles deployed and available in the terminal premises.

INTER-IoT will help to expand the possibilities offered by not only SEAMS and the sensors installed on its own container terminal vehicles and container handling equipment units, but also sensors available on third party equipment (i.e., reefer containers) and vehicles (i.e., external trucks picking up and delivering containers). Finally, it will allow installation of sensors on legacy equipment that does not have them available. Moreover, INTER-IoT will allow to seamlessly connecting the container terminal IoT ecosystem with other ecosystems owned by other parties, e.g., the port authority, road hauliers, the individual trucks, vehicles, containers and vessels through intelligent objects offered by different vendors, some of them managed by the PCS [30].

On the other hand, UPV provided I3WSN, semantic IoT methodology and platform deployed in application domains like factories, automotive and defence. This generic architecture was developed within a large Spanish National project FASyS and has been extended to be used in different areas like port transportation and m-health. The framework provides interoperability at different layers and includes reliability, privacy and security by design. Additionally, devices from the partners will be added to the trials and devices from the users (e.g., truck drivers or terminal operators) like smart phones will be added to the system following BYOD (Bring Your Own Device) philosophy, allowing the integration of COTS devices in the large scale trials.

Although the different platforms that the transport and logistics use case integrates (in particular, IoT-PCS from VPF, NOATUM TOS, I3WSN UPV and the IoT platforms from other stakeholders) share some characteristics, they have different aims (i.e., focused on the particular benefits of the administrator/operator and use different technologies). All of them gather data, using different M2M and P2M protocols; some of them are cloud-based and others will be, but the most important thing is that they lack interoperability in terms of the five identified layers. There is a potential integration using one of the platforms (i.e., IoT-PCS) as a matrix architecture; however interoperability and integration will not profit the power of the proposed approach neither the capabilities of interoperable architectures rather than interconnected architectures. The use case, mainly focused in the transportation of containers, as it is the most sensorized in port transportation (especially reefer and International Maritime Organization—IMO safe containers), may improve efficiency, security and benefits to the whole transport chain. Additionally, INTER-IoT will provide the pos-

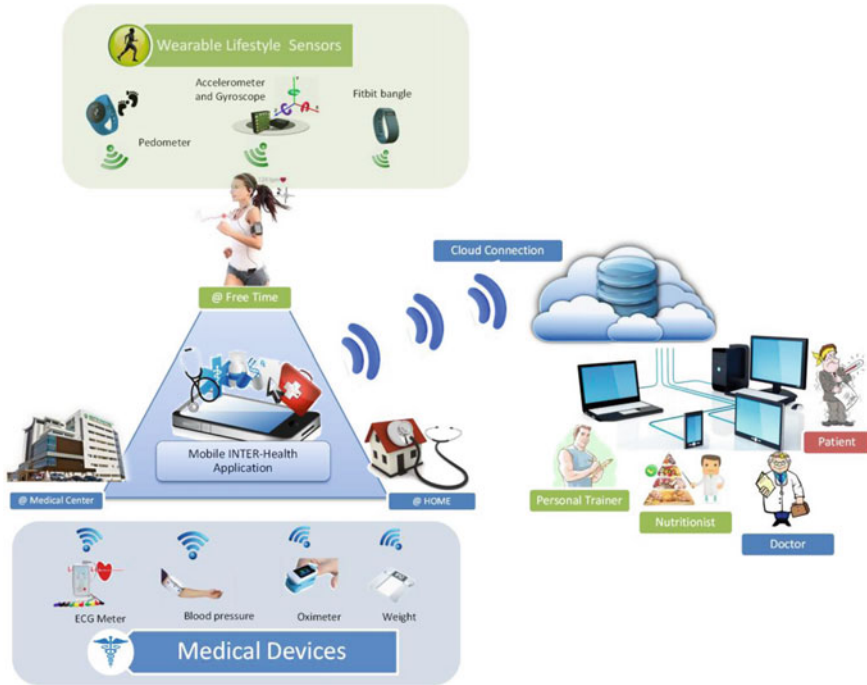


Fig. 3 INTER-IoT interconnection for m-Health (INTER-Health)

sibility to interact with other IoT platforms available in the port surroundings like Valencia City FIWARE infrastructure (i.e., VLCi) that is an open platform that will provide contextual information for different services and interactions at data and services layers [31–33] (Fig. 3).

3.2 *INTER-Health: Interoperability for Mobile Health for Chronic Patients*

The Decentralized and Mobile Monitoring of Assisted Livings' Lifestyle use case, aims at developing an integrated IoT system for monitoring humans' lifestyle in a decentralized way and in mobility, to prevent health issues mainly resulting from food and physical activity disorders. Users that attend nutritional out-patient care centres are healthy subjects with different risk degrees (normal weight, overweight, obese) that could develop chronic diseases. Only the obese (in case of second and third level obesity) need, at times, hospital care and get into a clinical and therapeutic route. The medical environment in which the pilot will be developed and deployed is the Dept. of Prevention/Hygiene Nutrition Unit at ASLTO5.

The use case will focus in the fact that in main chronic diseases, such as cardiovascular diseases, stroke, cancer, chronic respiratory diseases and diabetes, there are common and modifiable risk factors that are the cause of the majority of deaths (and of new diseases). Between the common and modifiable risk factors there are wrong lifestyles such as improper and hyper caloric diet and, in particular, the lack of physical activity. Every year in the world (World Health Organization and others, 2013): 2.8 million people die for obesity or overweight; 2.6 million people die for high cholesterol levels; 7.5 million people die for hypertension; 3.2 million people die for physical inactivity. These wrong lifestyles are expressed through the intermediate risk factors of raised blood pressure, raised glucose levels, abnormal blood lipids, particularly Low Density Lipoprotein (LDL cholesterol) and obesity (body mass index superior to 30kg/m²).

According to the reference standard medical protocol for the global prevention and management of obesity, written by the World Health Organization, in order to assess the health status (underweight, normal weight, overweight, obesity) of the subject (of a given age) during the visit at the healthcare center, objective and subjective measurements should be collected (and/or computed) by a health-care team (doctor, biologist nutritionist, dietician, etc.). The objective measurements are: weight, height, body mass index (enabling diagnosis of overweight and obesity), blood pressure or waist circumference. The subjective measurements reported by the subject, are collected through computerized questionnaires, and concern the eating habits: quality and quantity of food consumed daily and weekly, daily consumption of main meals (breakfast, lunch, dinner and snacks) and the practice of physical activity (quality and quantity of physical activity daily and weekly). The physical activity degree is detected subjectively during the first visit and could be objectively monitored through wearable monitoring devices. On the basis of these measurements, the caloric needs are automatically calculated, and the diet of the subject is defined. From this point forward, the subject must be monitored periodically (for example, every three months) for a period of at least one year. Usually monitoring is carried out at the health-care center, where the objective and subjective measurements are cyclically repeated. Based on the results, and depending on the health status reached by the subject (improved or worsened), the possibility of redefining his diet and his physical activity is analyzed.

By exploiting an integrated IoT environment, the aforementioned monitoring process can be decentralized from the healthcare center to the monitored subjects' homes, and supported in mobility by using on-body physical activity monitors. Specifically, the system will be created by using a new IoT platform, named INTER-Health, obtained by integrating two already-existing heterogeneous, non-interoperable IoT platforms for e-Health according to the approach proposed in the INTER-IoT project, based on the INTER-FW and its associated methodology INTER-METH: (i) Uni-versAAL, developed by UPV, and (ii) BodyCloud [34], developed by UNICAL.

3.2.1 INTER-IoT Approach to INTER-Health

There is a need of integrating different IoT platforms as proposed in the INTER-Health use case. The effective and efficient integration of heterogeneous e-Health IoT Platforms will provide an appropriate answer to the challenges described in INTER-IoT proposal. The two platforms considered are UniversAAL and BodyCloud, and the result of the integration will allow developing a novel IoT m-Health system for Lifestyle Monitoring [27].

This flexibility allows deploying universAAL-based solutions in multiple configurations, such as local-only nodes, mobile nodes connected to server instances, or non-universAAL nodes connecting to a multi-tenant server. Communication between applications and/or sensors happens through three different buses. Messages and members are always described semantically using the domain ontologies at hand: (i) Context bus—An event-based bus for sharing contextual information from context publishers to context subscribers; (ii) Service bus—A request-based bus for on-demand execution and information retrieval from service callers to service providers and (iii) User Interface bus—A centrally-managed bus that allows applications to define abstract interfaces to be rendered by different User Interface (UI) modalities. In each bus, semantic reasoning is used to match the transferred messages to the appropriate destination. This way, applications and sensors only need to describe what they provide and what they require from others. There is no need to specify recipients, connections nor addresses explicitly [30].

BodyCloud is a SaaS architecture that supports the storage and management of body sensor data streams and the processing (online and offline analysis) of the stored data using software services hosted in the Cloud. In particular, BodyCloud endeavours to support several cross-disciplinary applications and specialized processing tasks. It enables 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 [34].

The BodyCloud approach is centered around four main decentralized components (or sides), namely Body, Cloud, Viewer, Analyst: (i) Body-side is the component, currently based on the SPINE Android, that monitors an assisted living through wearable sensors and stores the collected data in the Cloud by means of a mobile device; (ii) Cloud-side is the component, based on SaaS paradigm, being the first general-purpose software engineering approach for Cloud-assisted community BSNs; (iii) Viewer-side is the Web browser-enabled component able to visualize data analysis through advanced graphical reporting; and (iv) e Analyst-side is the component that supports the development of BodyCloud applications.

The two platforms, UniversAAL and BodyCloud, share some high-level characteristics while differ in objectives and technology. Specifically, they are both e-Health platforms, based on Bluetooth technology to interact with measurement devices, and based on Cloud infrastructures to enable data storing, off-line analysis, and data visualization. However, they have different specific objectives and are not interoperable from a technological point of view (at each layer and at the global level). Their spe-

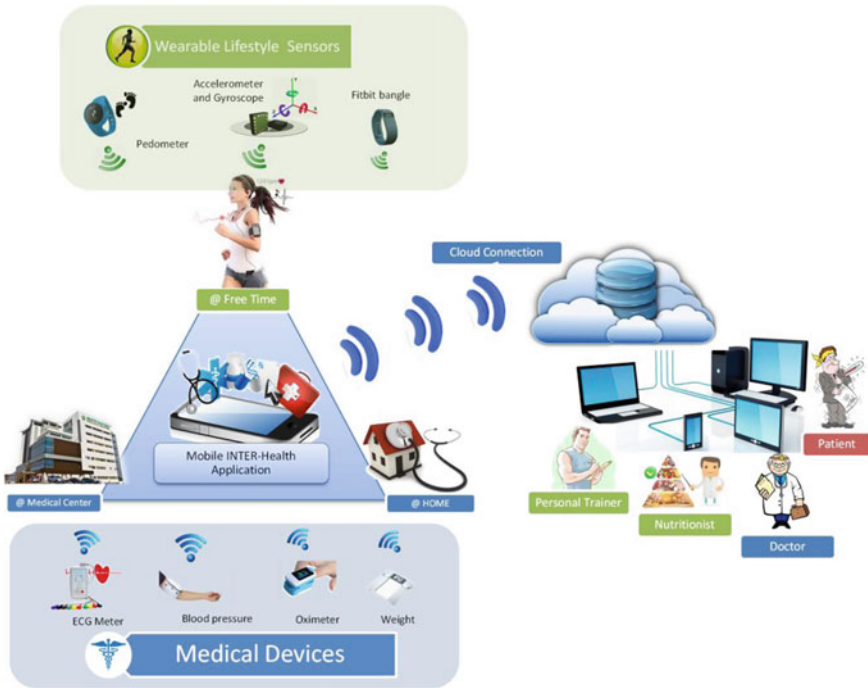


Fig. 4 INTER-IoT interconnection for m-Health (INTER-Health)

cific objectives are complementary: UniversAAL is focused mainly on non-mobile remote monitoring based on non-wearable measurement devices, whereas Body-Cloud provides monitoring of mobile subjects through wearable devices organized as body sensor networks. Thus, their integration will produce a full-fledged m-Health integrated platform on top of which multitudes of m-Health services could be developed and furnished. The use case will be fully deployable atop the integration of UniversAAL and BodyCloud: (i) the automated monitoring at the health-care center and the decentralization of the monitoring at the patients’ homes will be supported by UniversAAL remote services; (ii) the monitoring of mobile assisted livings would be enabled by the BodyCloud mobile services; (iii) new cross-platform services will be developed for enabling complete analysis of the measurement streams coming from assisted livings [28, 35] (Fig. 4).

4 INTER-IoT Progress Beyond the State of Art

The overall concept of the INTER-IoT project targets a full-fledged robust approach for seamless integration of different IoT platforms within and across different appli-

cation domains. Interoperability will be achieved at different layers, depending on the requirements of the specific scenario or the use case. The main outcomes of the project will be infrastructures for layer-oriented interoperability and a reference interoperable meta-architecture; an interoperability framework and an API along with an engineering methodology driven by a toolbox to be used by third parties to integrate heterogeneous IoT platforms and thus implement interoperable IoT applications. INTER-IoT will focus on two application domains (m-health and port transportation and logistics) and on their integration. The project outcome will optimize different operations in these two domains and in their integration. However, the INTER-IoT approach will be easily reused in any application domain, in which there is a need to interconnect already deployed (heterogeneous) IoT platforms. Or even in cross application domains, where IoT platforms and smart objects from different application domains will require co-operation or interoperability between them. Based on these principles, INTER-IoT targets the following innovations [5, 6].

4.1 Global Platform Interoperability

Global interoperability of hardware/software infrastructures is usually based on standards. However, as IoT is an evolving technology without specific central technical coordination and control, it is foreseen that many solutions and (pseudo) standards will be developed and proposed in the coming years. This will lead to massive heterogeneity. Indeed, currently many different (quasi) standards do exist in the IoT arena addressing: communications, hardware, software, and data. However, they mainly refer to specific IoT objects (sensors, sensor networks, RFID, nanocomputers, etc.) or contexts (smart grid, health-care, logistics, etc.). From the communications viewpoint, standards protocols at different level (MAC, network, application) are available: IEEE 802.11—WiFi, IEEE 802.16—WiMax, IEEE 802.15.4—LR-WPAN, 2G/3G/4G—Mobile Communication, Zigbee, Bluetooth, ANT+, NFC, M2M communications (M-Bus, WM-Bus, UWB, ModBus, Z-Wave), M2M ETSI, IPv4, IPv6, 6LowPAN, TCP, UDP, ISO/IEEE 11073 for medical devices, CoAP, HTTP, MQTT, XMPP, DDS, AMQP, Websocket, etc. From the hardware perspective, the technological state of the art is also heterogeneous: Arduino, BeagleBoard, TelosB sensor mote, RaspberryPI, pcDuino, Cubieboard, Libelium sensor/gateway, etc. The software realm is even richer including many base software technology (TinyOS, Contiki, FreeRTOS, eCos, Android, Ubuntu, Java, WebRTC, REST, WAMP, Django, etc.) and middleware solutions (FedNet, UbiComp, SmartProducts, ACOSO, SkyNet, etc.), including cloud computing-based infrastructures (Amazon EC2, Google App Engine, Xively, MS Windows Azure). Also the data (and semantics) level presents high heterogeneity: XML (and XML-based like WSDL), JSON, UDCAP, uCode Relational Model, RDF, OWL, W3C SSN (Semantic Sensor Network) [4, 11].

It is worth noting that, when we consider a complete IoT platform, the complexity of technologies used to build up the platform further arises as each defined layer (device, networking, middleware, application service, data and semantics) exploits

specific solutions that need to be holistically adapted to form the final platform. For instance, several available platforms, each of which was designed and deployed to fulfil quasi similar goals but exploiting heterogeneous IoT technological solutions, or providing any (or even limited) interoperability [36]. Thus, it is critical to provide bottom-up “voluntary” approaches able to integrate, interconnect, merge, heterogeneous IoT platforms to build up extreme-scale interoperable ecosystems on top of which large-scale applications can be designed, implemented, executed and managed [37].

INTER-IoT will provide the first full-fledged methodological and technological suite to completely address the fundamental issue of “voluntary interoperability”. The suite will be composed of three main building blocks: (i) Layer-oriented infrastructures to adapt heterogeneous peer layers (device-to-device, networking-to-networking, middleware-to-middleware, application services-to-application services, data-to-data, and semantics-to-semantics); (ii) Interoperable open framework to program and manage integrated IoT platforms; (iii) Engineering methodology and tools to drive the integration process of heterogeneous IoT platforms. By using INTER-IoT, IoT heterogeneity will be turned from the most limiting factor for IoT technology diffusion to its greatest advantage due to the exploitation of specific benefits and characteristics derived from multiple heterogeneous IoT platforms [15, 38, 39].

4.2 Gateway and Device Interoperability

As sensors, actuators and smart devices become smaller, more versatile, lower cost and more power efficient, they are being deployed in greater numbers, either as special-purpose devices or embedded into other products. The unification and convergence of the vast number of platforms already deployed, the accessibility (API and interfaces) of the platform to app developers, requires interoperability. Smartphones are key components in Device-to-Device (D2D) communication and interoperability, however there are many other types of devices that are currently deployed, both independently (e.g. smart watches and other wearables) and as part of other devices and platforms (e.g. consumer electronics or Cyber-Physical Systems). Different communication protocols are used at device level. Here, Cellular and WiFi that are ubiquitous; they are evolving to support higher bandwidths and lower cost. Bluetooth is also becoming lower cost. New communication technologies like Bluetooth low energy (Bluetooth LE) and NFC are opening new possibilities for IoT. However, also traditional communication protocols and mechanisms for sensors, actuators and smart objects have to be considered (e.g. ZigBee, ISA100, WirelessHart), in addition to other non-standard proprietary protocols developed by individual vendors, or even to new emerging protocols, e.g. [40, 41]. Different classes of IoT objects need different communication supports: e.g. ‘deterministic’ communication protocols (MAC and Routing layers) are not possible using current Internet protocols, but may be needed by some application. Standardization on these topics is just starting

(e.g., detnet working group in IETF). Yet, deterministic communications will hardly meet the interoperability requirements of all IoT objects. Typically device-level interconnection of IoT architectures has been performed using gateway-based solutions. FP7 Butler project proposed a device-centric architecture where a SmartGateway allows interconnection between smart objects (sensors, actuators, gateways) using IPv6 as communication protocol. Different approaches have been developed to integrate and interoperate devices in IoT architectures. Basic devices (e.g. sensors, tags, actuators) are virtualized and can be composed in more complex smart systems. The idea has been to create virtual objects, allowing object composition, considering a virtual object as a counterpart of existing smart objects [19, 42, 43].

INTER-IoT will provide fundamental benefits and competitiveness improvements in the way IoT devices will communicate with each other and will interface with different IoT platforms and subsystems. One of the proposed progresses regarding D2D interaction is to complement standardized communication protocols (which are mostly deterministic and reactive) with an ability for objects to make sense of their surroundings in order to understand how to best interplay with their neighbours. This requires new ‘proactive’ and ‘predictive’ communications capabilities, whereby a node can determine its communication requirements and those of its neighbours well before communication is required. It has recently been proven that machine learning capabilities can run even on small sensors (with as little as 20 KB of RAM). INTER-IoT will develop an interoperable communication layer, even based on lightweight machine learning that also accommodate for opportunistic communications among heterogeneous nodes/devices, based on prediction mechanisms [21, 44].

4.3 Networking Mobility and Interoperability

IoT products will encompass different data communication scenarios. Some may involve sensors that send small data packets infrequently and do not prioritize timely delivery. Others may involve storage in order to sustain periods when the communication link is down (e.g. Delay Tolerant Networks). Others may need high bandwidth but be able to accept high latency. And others may need high quality, high bandwidth, and low latency. Large amounts of traffic with relatively short packet sizes will require sophisticated traffic management. More efficient protocols can help reduce overhead but may present challenges to system integrity, reliability and scalability. Interface standardization is desirable so that IoT objects can communicate quickly and efficiently, and allow mobility between interoperable IoT platforms. IoT objects will need a way to quickly and easily discover each other and learn their neighbour’s capabilities [28, 45].

At networking and communications layer different protocols can be used like 6LoWPAN, TCP/HTTP, UDP/CoAP. Communication between real objects and the gateway can be based on universal plug and play (UPnP) or DLNA. Use of buses based on MQTT protocol can also be used to implement asynchronous communications

between entities. The most promoted networking protocol in IoT environments is IPv6 and its version for constrained devices 6LoWPAN, even though its adoption is slow, and without global adoption it will be impossible for IoT to proliferate. IPv6 provides the following benefits to IoT configurations: (i) IPv6 auto-configuration; (ii) Scalable address space (sufficiently large for the enormous numbers of IoT objects envisioned); (iii) Redefined headers that enable header compression formats; (iv) Easy control of the system of things; (v) Open/Standard communications; (vi) IPv6 to IPv4 transition methods; (vii) IPv6 over constrained node networks (6LO, 6LoWPAN [11, 46]).

IoT platforms have usually mechanisms for integrating with external systems, but they are all based on specific point to point connections, usually with legacy systems in the area of interest of the IoT platform (city, neighbourhood, factory, hospital, port, house, etc.). The integration between IoT platforms will allow tracking the behaviour of these objects when they move outside the intrinsic area of interest and get into the area of interest of another IoT platform. The pub/sub mechanism usually available in the communication buses at the core of these IoT platforms and the possible object context sharing allow a powerful and easy way to track the behaviour of these objects among different IoTs scope areas.

INTER-IoT will provide support for as many networks as possible, including as many networks as possible in the INTER-FW definition. Main contributions of the project will be focused to multihoming capabilities among the different IoT objects in order to provide network offloading connectivity and seamless mobility between different IoT platforms of moving objects. INTER-IoT will use SDN components to configure interconnection at network level and also will use ICN/CCN as support for interoperability and roaming of smart objects between different platforms of the same ecosystem while keeping secure connectivity and also guaranteed quality of service. Resource management and scalability so as reliability, trust, privacy and security will be non-functional requirements that will be addressed by the project to provide optimal interoperability at network layer [6, 30, 47].

4.4 Middleware Platform Interoperability

Middleware, widely used in conventional distributed systems, is a fundamental tool for the design and implementation of both IoT devices and IoT systems. They provide general and specific abstractions (e.g., object computation model, inter-object communication, sensory/actuation interfaces, discovery service, knowledge management), as well as development and deployment tools, through which IoT devices, IoT systems and their related applications can be easily built up. Indeed, middleware (i) enable connectivity for huge numbers of diverse components comprised at Device Layer, (ii) realize their seamless interoperability at Networking Layer, and (iii) ensure operational transparency at Application Service Layer. In such a way, heterogeneous, often complex and already existing IoT devices and IoT systems, belonging to different application domains and not originally designed to be con-

nected, can be easily integrated, effectively managed and jointly exploited. It follows that the role of middleware within the cyberphysical, heterogeneous, large scale and interconnected IoT scenario is even more crucial than within conventional distributed systems. Over the years, many IoT middleware have been proposed, so much so that only in [11, 34] more than 70 contributions have been surveyed and compared. The best way to analyse such plethora of middleware, regardless of the specific detail or technology, is to build up comparison frameworks around well-defined criteria to effectively highlight their salient differences and similarities.

In very few words, LinkSmart is service-based middleware for ambient intelligence (AmI) systems, supporting devices communication, virtualization, dynamic reconfiguration, self-configuration, energy optimization and security by means of Webservice-based mechanisms enriched by semantic resolution. UbiROAD is semantic, context-aware, self-adaptive agent-based middleware for smart road environments, aiming at collecting, analysing and mining real time data from in-car and roadside heterogeneous devices. ACOSO is an agent-oriented middleware with a related methodology fully supporting the development (from the modelling to the implementation phase), management and deployment of smart objects and IoT systems, as well as their integration with the Cloud. IMPReSS, finally, is a middleware conceived for the rapid development of context-aware, adaptive and real-time monitoring applications to control and optimize energy usage in smart cities [2, 39, 48, 49].

In particular, INTER-IoT will focus on defining component-based methods for middleware interoperability/integration; in particular, we focus on discovery, management and high-level communication of IoT devices in heterogeneous IoT platforms. We will define two main approaches: (i) definition of overlay middleware components able to couple the middleware components of the heterogeneous IoT platforms; (ii) virtualization of the heterogeneous middleware components. In the first approach, we will design overlay middle components such as mediators and brokers [6].

4.5 *Semantic Interoperability*

Semantic interoperability can be conceptualized as an approach to facilitate “combining” multiple IoT platforms. The simplest case, of combining two IoT platforms, could be addressed by developing a one-to-one translator (a “gateway”) to allow “semantic understanding” between them. However, this approach does not scale, as for every subsequent entity joining an assembly of N platforms. Thus, N translators would have to be created. The two main approaches to avoid this problem, and deal with semantic interoperability are: (i) common communication standards; (ii) ontology and semantic data processing [50]. Developing a common communication standard, is tried in the travel domain with the OTA message specification a standard consisting on a set of (XML-demarcated) messages; or in the healthcare domain (and thus related to the INTER-Health use case) with OpenEHR, which is an

open domain-driven platform for developing flexible e-health systems. Here, multiple projects strive to establish interoperability between already known standards and the OpenEHR, e.g. establishing semantic interoperability of the ISO EN 13606 and the OpenEHR archetypes. Similarly the Think!EHR Platform (health data platform based on vendor-neutral open data standards designed for real-time, transactional health data storage, query, retrieve and exchange); aims at establishing interoperability of the OpenEHR and the HL7 standard (a framework for the exchange, integration, sharing, and retrieval of electronic health information). Interestingly, development of the Think!EHR Platform had to deal with the data standards problem caused by existence of HL7 RIMv3, ISO13606, and OpenEHR standards. While it is possible to envision an approach similar to this, applied to individual domains, it is not likely to be easily generalizable to support interactions between domains. Therefore, approaches based on ontologies and semantic data processing will be used in the project [51, 52]. INTER-IoT approach will be based on development of a generic ontology of an IoT Platform (GOIoTP). The GOIoTP will be used as the centerpiece for establishing platform interoperability (allowing for, among others, data interoperability, message translation, etc.). It should be stressed that, state-of-the-art ontologies of the IoT, will constitute the starting point for construction of the GOIoTP, needed in our project. The proposed approach will require, (i) ontology matching, (ii) merging, noting that ontology merging is often reduced to ontology matching, as well as (iii) techniques for establishing semantic distance (needed for ontology matching). Observing that this approach allows “understanding” and adaptability (handled through ontology adaptation) of heterogeneous data [39, 53, 54].

The creation of GOIoTP in INTER-IoT, combined with the state-of-the-art approaches to ontology matching/merging will allow development of a comprehensive support for facilitation of semantic interoperability between IoT platforms. The resulting approach, based on conducted research, will consist both of the methods and supporting tools and will include, among others, methods for:

- Combining two (or more) IoT platforms with explicitly defined ontologies. Here, among others, the following issues will be researched: (i) bringing multiple ontologies to a common format / language (for example, transforming XML into RDF and further transforming it into OWL-delineated ontology using XLST), (ii) ontology matching, to allow for (iii) ontology merging into the extended GOIoTP (as the top-level ontology).
- Combining two (or more) IoT platforms with explicitly defined ontologies. Here, among others, the following issues will be researched: (i) bringing multiple ontologies to a common format / language (for example, transforming XML into RDF and further transforming it into OWL-delineated ontology using XLST), (ii) ontology matching, to allow for (iii) ontology merging into the extended GOIoTP (as the top-level ontology) [55].
- Joining an “incoming” IoT platform (with an explicitly defined ontology) to an existing federation of IoT platforms (with an already defined common ontology). Here the process would be somewhat a simplified version of the previous method as only two ontologies will be integrated.

- Dealing with IoT platforms without an explicitly defined ontology / taxonomy / etc. Here, appropriate set of tools will be adapted to help instantiate an ontology for the multi-IoT-platform under construction. Specifically, the ontology will be built on the basis of information contained in one, or more: (i) definition of used data; (ii) structure of the database(s); (iii) queries issued on the database(s); and (iv) exchanged messages [54].

4.6 Cross-Layer Approach for Interoperability

INTER-IoT specifically aims at creating cross-layer interoperability and integration between heterogeneous IoT platforms. Cross-layer approaches are fundamental to made interoperable/integrate the whole layer stack (device, networking, middleware, application service, data and semantics) of IoT platforms. Cross layering will be therefore based on the outcomes of the previous points.

Moreover, important requirements and features such as Quality of Service (QoS), Quality of Experience (QoE), Security, Privacy, Trust and Reliability, require to be addressed at each layer with different mechanisms. Such transversal approach allows retaining the benefits of a layered architecture (e.g., modularity, interoperability, etc.) but adding, at the same time, flexibility (e.g., optimization, tunable design, etc.) to those components that require it. Considering the heterogeneity and spread of IoT devices and IoT applications, it is straightforward that such design choice is more than suitable to properly support (i) dynamic QoS and QoE (the former, basically aiming at splitting traffic up into priority classes and trying to guarantee a particular performance metric, the latter at combining more subjective aspects related to user perception into evaluating a service); (ii) novel security and privacy techniques (that consider the cyber-physical nature of IoT devices as well as of the IoT application contexts); extended trust models (in which unconventional actors, like social networks, play an important role) and (iv) enhanced reliability mechanisms (to deal with failure of resource-limited IoT device, lack of coverage from access networks in some region, rapid application context switches, etc.) [11, 56].

5 Conclusions

In this chapter we have presented the INTER-IoT systemic approach, which is being created within the INTER-IoT project together with necessary software tools and enduser applications. It will provide ways of overcoming interoperability problems between heterogeneous IoT systems across the communication/software stack, including: devices, networks, middleware, application services, data/semantics. Henceforth, reuse and integration of existing and future (even standard) IoT systems will be facilitated and made possible to obtain interoperable ecosystems of IoT platforms.

As the ecosystem of interoperable devices and services expands, so will increase the value of building new devices for and applications working within this ecosystem. This emerging ecosystem is not owned by any business or entity, but rather it exists to enable many entities to pool their resources together to create larger opportunities for all. Open interoperability delivers on the promise of open source software, enabling vendors and developers to interact and interoperate, without interfering with anyone's ability to compete by delivering a superior product and experience. In the absence of global IoT standards, the INTER-IoT project and results will support and make it easy for any company to design IoT devices, smart object, or services and get them to market quickly, to a wider client-base, and to create new IoT interoperable ecosystems. In the long term, ability for multiple applications to connect to and interact with heterogeneous sensors, actuators, and controllers, thus making them interoperable, will become a huge enabler for new products and services.

References

1. Fortino, G., Savaglio, C., Spezzano, G., Zhou, M.C.: Internet of things as system of systems: a review of methodologies, frameworks, platforms, and tools. *IEEE Trans. Syst. Man Cybern. Syst.* **51**(1), 223–236 (2021)
2. Savaglio, C., Ganzha, M., Paprzycki, M., Badica, C., Ivanovic, M., Fortino, G.: Agent-based Internet of Things: state-of-the-art and research challenges. *Future Gener. Comput. Syst.* **102**, 1038–1053 (2020)
3. Vermesan, O., Friess, P. (eds.): *Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds*. Riverpublishers (2016)
4. Vermesan, O., Friess, P. (eds.): *D3.2. Methods for Interoperability and Integration*, vol. 2. INTER-IoT H2020 project, October 2017. <https://inter-iot.eu/deliverables>
5. Broring, A., Zappa, A., Vermesan, O., Främling, K., Zaslavsky, A., Gonzalez-Usach, R., Szmeja, P., Palau, C., Jacoby, M., Zarko, I.P., Sour-sos, S., Schmitt, C., Plociennik, M., Krco, S., Georgoulas, S., Larizgoitia, I., Gligoric, N., García-Castro, R., Serena, F., Orav, V.: *Advancing IoT Platform Interoperability*. River Publishers, The Netherlands (2018)
6. Gravina, R., Palau, C.E., Manso, M., Liotta, A., Fortino, G. (eds.): *Integration, Interconnection, and Interoperability of IoT Systems*. Internet of Things. Springer International Publishing (2018)
7. Fortino, G., Palau, C.E., Guerrieri, A., Cuppens, N., Cuppens, F., Chaouchi, H., Gabillon, A. (eds.): *Interoperability, Safety and Security in IoT—Third International Conference, InterIoT 2017, and Fourth International Conference, SaSeIoT 2017, Valencia, Spain, November 6–7, 2017*. Proceedings. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 242. Springer (2018)
8. Fortino, G., Garro, A., Russo, W.: Achieving mobile agent systems interoperability through software layering. *Inf. Softw. Technol.* **50**(4), 322–341 (2008)
9. Fortino, G., Garro, A., Russo, W.: *D4.5. Interoperable IoT Framework API and tools v1*. INTER-IoT H2020 project, October 2017. <https://inter-iot.eu/deliverables>
10. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: Towards multi-layer interoperability of heterogeneous IoT platforms: the inter-IoT approach. *Internet of Things* 199–232 (2018)
11. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: *D3.3. Methods for Interoperability and Integration Final*. INTER-IoT H2020 project, June 2018. <https://inter-iot.eu/deliverables>

12. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: D4.2. Final Reference IoT Platform Meta-architecture and Meta-data Model. INTER-IoT H2020 project, January 2018. <https://inter-iot.eu/deliverables>
13. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: D4.6. Interoperable IoT Framework API and Tools, Model and Engine v2. INTER-IoT H2020 project, June 2018. <https://inter-iot.eu/deliverables>
14. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: D4.3. Initial Reference IoT Platform Meta-Architecture and Meta Data Model Interoperable IoT framework Model and Engine v1. INTER-IoT H2020 project, October 2017. <https://inter-iot.eu/deliverables>
15. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: D5.1. Design Patterns for Interoperable IoT Systems. INTER-IoT H2020 project, December 2017. <https://inter-iot.eu/deliverables>
16. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: D5.2. INTER-Meth: Full-fledged Methodology for IoT Platforms Integration. INTER-IoT H2020 project, December 2017. <https://inter-iot.eu/deliverables>
17. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: D6.1. System Integration Plan. INTER-IoT H2020 project, August 2017. <https://inter-iot.eu/deliverables>
18. Yacchirema, D.C., Esteve, M., Palau, C.E.: Design and implementation of a gateway for pervasive smart environments. In: 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), pp. 004454–004459, October 2016
19. Aloï, G., Fortino, G., Gravina, R., Pace, P., Caliciuri, G.: Edge computing-enabled body area networks. In: 2018 32nd International Conference on Advanced Information Networking and Applications Workshops (WAINA), pp. 349–353, Krakow, May 2018. IEEE
20. Mocanu, D.C.: On the synergy of network science and artificial intelligence. In: Proceedings of the Twenty-Fifth International Joint Conference on Artificial Intelligence, IJCAI'16, pp. 4020–4021, New York, New York, USA, July 2016. AAAI Press
21. Cankar, M., Gorriti, E.O., Marković, M., Fuat, F.: Fog and cloud in the transportation, marine and eHealth domains. In: Heras, D.B., Bougé, L., Mencagli, G., Jeannot, E., Sakellariou, R., Badia, R.M., Barbosa, J.G., Ricci, L., Scott, S.L., Lankes, S., Weidendorfer, J. (eds.) Euro-Par 2017: Parallel Processing Workshops. Lecture Notes in Computer Science, vol. 10659, pp. 292–303. Springer International Publishing, Cham (2018)
22. Belsa, A., Sarabia-Jacome, D., Palau, C.E., Esteve, M.: Flow-based programming interoperability solution for IoT platform applications. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 304–309, Orlando, FL, April 2018. IEEE
23. Ganzha, M., Paprzycki, M., Pawłowski, W., Szmaja, P., Wasielewska, K.: Semantic technologies for the IoT: an inter-IoT perspective. In: 2016 IEEE First International Conference on Internet-of-Things Design and Implementation (IoTDI), pp. 271–276, April 2016
24. Ganzha, M., Paprzycki, M., Pawłowski, W., Szmaja, P., Wasielewska, K.: Semantic interoperability in the Internet of Things: an overview from the INTER-IoT perspective. *J. Netw. Comput. Appl.* **81**, 111–124 (2017)
25. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: D7.3. Final Evaluation Report. INTER-IoT H2020 project, December 2018. <https://inter-iot.eu/deliverables>
26. Yacchirema, D., Gonzalez-Usach, R., Esteve, M., Palau, C.E.: Interoperability of IoT platforms applied to the transport and logistics domain. In: Transport Arena Research Conference 2018, Austria. TRA (2018)
27. Margherita, G., Claudio, M., Anna, C., Marina, M., Ilaria, D.L., Monica, M., Massimo, U., Luciano, B., Massimo, C., Angelina, D.T., Bartolomeo, A., Anna, A., Domenica, P., Lucia, A. Fortunata, M., Rina: Telemedicine and prevention: electromedical devices experimentation in the nutritional counseling. Mattioli 1885 SpA Italy (2017)
28. Margherita, G., Claudio, M., Anna, C., Marina, M., Ilaria, D.L., Monica, M., Massimo, U., Luciano, B., Massimo, C., Angelina, D.T., Bartolomeo, A., Anna, A., Domenica, P., Lucia, A.

- Fortunata, M., Rina: D6.3. Site Acceptance Test Plan. INTER-IoT H2020 project, September 2018. <https://inter-iot.eu/deliverables>
29. Margherita, G., Claudio, M., Anna, C., Marina, M., Ilaria, D.L., Monica, M., Massimo, U., Luciano, B., Massimo, C., Angelina, D.T., Bartolomeo, A., Anna, A., Domenica, P., Lucia, A. Fortunata, M., Rina: D7.2. Technical Evaluation and Assessment Report. INTER-IoT H2020 project, September 2018. <https://inter-iot.eu/deliverables>
 30. Yacchirema, D., Sarabia-Jácome, D., Palau, C.E., Esteve, M.: System for monitoring and supporting the treatment of sleep apnea using IoT and big data. *Pervasive Mobile Comput.* **50**, 25–40 (2018)
 31. Drozdowicz, M., Alwazir, M., Ganzha, M., Paprzycki, M.: Graphical interface for ontology mapping with application to access control. In: Nguyen, N.T., Tojo, S., Nguyen, L.M., Trawiński, B. (eds.) *Intelligent Information and Database Systems. Lecture Notes in Computer Science*, vol. 10191, pp. 46–55. Springer International Publishing, Cham (2017)
 32. Drozdowicz, M., Alwazir, M., Ganzha, M., Paprzycki, M.: D6.2. Factory Acceptance Test Plan. INTER-IoT H2020 project, February 2018. <https://inter-iot.eu/deliverables>
 33. Drozdowicz, M., Alwazir, M., Ganzha, M., Paprzycki, M.: D8.6. Report on Impact Creation at M36. INTER-IoT H2020 project, December 2018. <https://inter-iot.eu/deliverables>
 34. Fortino, G., Parisi, D., Pirrone, V., Di Fatta, G.: Bodycloud: a SaaS approach for community body sensor networks. *Future Gener. Comput. Syst.* **35**, 62–79 (2014)
 35. Drozdowicz, M., Ganzha, M., Paprzycki, M.: Semantically enriched data access policies in eHealth. *J. Med. Syst.* **40**(11), 238 (2016)
 36. Fortino, G., Savaglio, C., Zhou, M.: Toward opportunistic services for the industrial Internet of Things. In: 2017 13th IEEE Conference on Automation Science and Engineering (CASE), pp. 825–830, Xi'an, August 2017. IEEE
 37. Vargas, D.C.Y., Salvador, C.E.P.: Smart IoT gateway for heterogeneous devices interoperability. *IEEE Lat. Am. Trans.* **14**(8), 3900–3906 (2016)
 38. Fortino, G., Gravina, R., Russo, W., Savaglio, C.: Modeling and simulating Internet of Things systems: a hybrid agent-oriented approach. *Comput. Sci. Eng.* **19**(5), 68–76 (2017)
 39. Ganzha, M., Paprzycki, M., Pawlowski, W., Szymeja, P., Wasielewska, K.: Alignment-based semantic translation of geospatial data. In: 2017 3rd International Conference on Advances in Computing, Communication and Automation (ICACCA) (Fall), pp. 1–8, Dehradun, India, September 2017. IEEE
 40. Smart, G., Deligiannis, N., Surace, R., Loscri, V., Fortino, G., Andreopoulos, Y.: Decentralized time-synchronized channel swapping for ad hoc wireless networks. *IEEE Trans. Veh. Technol.* **65**(10), 8538–8553 (2016)
 41. Pace, P., Fortino, G., Zhang, Y., Liotta, A.: Intelligence at the edge of complex networks: the case of cognitive transmission power control. *IEEE Wirel. Commun.* **26**(3), 97–103 (2019)
 42. Gonzalez-Usach, R., Collado, V., Esteve, M., Palau, C.E.: AAL open source system for the monitoring and intelligent control of nursing homes. In: 2017 IEEE 14th International Conference on Networking, Sensing and Control (ICNSC), pp. 84–89, May 2017
 43. Pace, P., Aloï, G., Gravina, R., Fortino, G., Larini, G., Gulino, M.: Towards interoperability of IoT-based health care platforms: the inter-health use case. In: Proceedings of the 11th EAI International Conference on Body Area Networks, BodyNets'16, pp. 12–18, Brussels, BEL, December 2016. ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering)
 44. Exarchakos, G., Oztelcan, I., Sarakiotis, D., Liotta, A.: plexi: adaptive re-scheduling web-service of time synchronized low-power wireless networks. *J. Netw. Comput. Appl.* **81**, 62–73 (2017)
 45. Pradilla, J., González, R., Esteve, M., Palau, C.: Sensor observation service (SOS)/constrained application protocol (COAP) proxy design. In: 2016 18th Mediterranean Electrotechnical Conference (MELECON), pp. 1–5, April 2016. ISSN: 2158-8481
 46. Kotian, R., Exarchakos, G., Stavros, S., Liotta, A.: Impact of transmission power control in multi-hop networks. *Future Gener. Comput. Syst.* **75**, 94–107 (2017)

47. Kotian, R., Exarchakos, G., Stavros, S., Liotta, A.: D7.1. Evaluation Plan. INTER-IoT H2020 project, March 2018. <https://inter-iot.eu/deliverables>
48. Savaglio, C., Fortino, G., Zhou, M.: Towards interoperable, cognitive and autonomic IoT systems: an agent-based approach. In: 2016 IEEE 3rd World Forum on Internet of Things (WF-IoT), pp. 58–63, December 2016
49. Savaglio, C., Fortino, G., Gravina, R., Russo, W.: A methodology for integrating Internet of Things platforms. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 317–322, Orlando, FL, April 2018. IEEE
50. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K., Palau, C.E.: From implicit semantics towards ontologies—practical considerations from the INTER-IoT perspective. In: 2017 14th IEEE Annual Consumer Communications Networking Conference (CCNC), pp. 59–64, January 2017. ISSN: 2331-9860
51. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K.: Streaming semantic translations. In: 2017 21st International Conference on System Theory, Control and Computing (ICSTCC), pp. 1–8, Sinaia, October 2017. IEEE
52. Pileggi, S.F., Palau, C.E., Esteve, M.: Building semantic sensor web: knowledge and interoperability. In: Proceedings of the International Workshop on Semantic Sensor Web (SSW), (IC3K 2010), vol. 1, pp. 15–22. INSTICC, SciTePress (2010)
53. Tkaczyk, R., Szymeja, P., Ganzha, M., Paprzycki, M., Solarz-Niesluchowski, B.: From relational databases to an ontology—practical considerations. In: 2017 21st International Conference on System Theory, Control and Computing (ICSTCC), pp. 254–261, Sinaia, October 2017. IEEE
54. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K.: Identifier management in semantic interoperability solutions for IoT. In: 2018 IEEE International Conference on Communications Workshops (ICC Workshops), pp. 1–6, Kansas City, MO, May 2018. IEEE
55. Moreira, J.L., Daniele, L.M., Pires, L.F., van Sinderen, M.J., Wasielewska, K., Szymeja, P., Pawłowski, W., Ganzha, M., Paprzycki, M.: Towards IoT platforms integration: semantic translations between W3C SSN AND ETSI SAREF. In: Semantics. Workshop Semantic Interoperability and Standardization in the IoT (SIS-IoT), November 2017
56. Frustaci, M., Pace, P., Aloï, G.: Securing the IoT world: issues and perspectives. In 2017 IEEE Conference on Standards for Communications and Networking (CSCN), pp. 246–251, Helsinki, Finland, September 2017. IEEE

INTER-IoT Requirements



Pablo Giménez, Miguel Llop, Regel Gonzalez-Usach, and Miguel A. Llorente

Abstract There are some significant tasks in the first stage of a project in order to achieve success when taking out a new product, a service or release a software. It is essential to know what is in the market and what the potential customers want. Therefore, during the project, we performed a market analysis regarding all the products related with IoT and we had interviews with all the involved actors in all the domains, such as developers, integrators, operators, domain users, clients, etc. Furthermore, from the previous information we define the requirements in order to determine how the system should work and what it should do. This chapter presents the process and the results of these three activities developed in the first stage of the INTER-IoT project: market analysis, stakeholders analysis, and requirements analysis. This task was done for the five different products defined in the project: INTER-LAYER, INTER-FW, INTER-METH, INTER-LogP, and INTER-Health. These tasks have been made using the VOLERE methodology, which is an excellent methodology to extract conclusions and provide results following a systematic approach.

1 Introduction

Requirements are an essential part of any software project and it is critical to devote enough time to identify all project requirements [1]. The gathering of and compiling of requirements for a software project requires a very tight collaboration between the clients, the developers and the software integrators. This is because the behaviour attributes and properties of the future system rely dramatically on a correct identification of requirements [2].

P. Giménez (✉) · M. Llop
Fundación Valenciaport, Valencia, Spain
e-mail: pgimenez@fundacion.valenciaport.com

R. Gonzalez-Usach
Universitat Politècnica de València, Valencia, Spain

M. A. Llorente
Prodevelop, Valencia, Spain

There are two main reasons for specifying requirements. First, it guides design and decision-making processes towards a correct solution [3]. In addition, it provides a basis of testing the implemented solution is correct [1].

It must be noted that the creation of the INTER-IoT framework represents a vast and very complex development task that encompasses all layers from IoT systems, and novel aspects in IoT-related software [4].

Therefore, requirements on each level of the IoT stack [5] must be identified and collected, including the platform layer [6], middleware [7], gateway or fog domain [8], network [8], semantics [9, 10], service and application layer [11, 12], security [13] and cross-layer elements [14] and other IoT-related aspects [5].

The first stage of the project was focused on the definition of all the elements necessary for the development of the project. The objective of the initial phase was to gather and define the set of technical requirements for the development of INTER-IoT framework and for each of its core components. The work in this work package was structured in five tasks:

- Identify and analyse stakeholders and products for both use cases addressed in the project: smart port logistics and smart mobile m-health.
- Collect and analyse requirements from targeted stakeholders, and other involved actors to develop the INTER-IoT products.
- Identify and design suitable business models for the INTER-IoT system.
- Define comprehensive suitable scenarios for the use cases and the targeted stakeholders.
- Analyse legal and regulatory requirements that are relevant to INTER-IoT.

1.1 Methodology

The methodology selected as a reference for most of the above tasks is VOLERE¹ [15], as it represents a solid methodology widely used by thousands of organizations around the world in order to define, discover, communicate and manage all the necessary requirements for any type of system development (e.g. software, hardware, commodities, services, organizational, etc.). The VOLERE methodology [3, 15] is used in INTER-IoT mainly because it helps project partners to describe, formalize and track the project market analysis, requirements, use cases and scenarios in an explicit and unambiguous manner, and has widely proven to be effective and reliable for this type of tasks. It is also quite clear and simple to apply.

VOLERE can be applied in almost all kinds of development environments, with any other development methods or with most requirements tools and modelling techniques. To produce accurate and unambiguous requirements, the VOLERE methodology uses techniques that are based on experience from worldwide business analysis projects, and are continually improved [16].

¹ <http://www.volere.co.uk/>.

The VOLERE methodology provides several templates to deal with the different techniques and activities that it includes. In a quick view, the VOLERE Requirement Process² that this methodology suggests can be summarised as follows:

1. Define the Purpose of the Project
2. Stakeholders Identification and Analysis
3. Business Use Cases
4. Scenarios
5. Writing the Requirements (functional requirements and non-functional requirements)
6. Validation of requirements (completeness, relevance, testability, coherency, traceability, and several other qualities before they allow it to be passed to the developers)
7. Communicating the Requirements
8. Requirements Completeness.

The consequent application of the VOLERE methodology is not only useful in the initial phases of the project but it is also helpful in specifying a reference point for the later stages. During the implementation and management, it can be used to track and evaluate the progress of the individual work packages and the overall project. Besides being efficient and easy to use, the VOLERE methodology provides a mechanism for all partners to specify requirements, needs, use cases and scenarios in a standard format. Thereby, specifying additional context of an element such as the rationale and the acceptance criteria helps to build a common understanding of the overall system [16]. Furthermore, defining priorities helps to clarify the focus of the project [17].

1.2 Repository

The implementation of this methodology in the INTER-IoT project has been done using a commercial software called Jira.³ Jira is an online tracking tool to manage the whole project progress, where all the information compiled and produced is accessible through to the different partners. Following the Volere recommendations, all the requirements, scenarios, stakeholders, and products were registered.

This repository help to search, access, review, and register new elements (stakeholders, requirements, etc.) identified after completion of the initial stage as the project progresses.

² Volere Getting Started <http://www.volere.co.uk/pdf%20files/VolereGettingStarted.pdf>.

³ <https://es.atlassian.com/software/jira>.

2 Stakeholders Analysis

2.1 Definition

Stakeholders' analysis [18] was the first task in the project and it is part of a rigorous and complete requirements specification [16]. It describes the process of targeting the people who have an interest in the new products and results that are planned for the project. The stakeholders' group being involved in this task has also included anyone who may have any influence on the project's outcomes, may be affected by the product or may have any knowledge needed to uncover the requirements of these products. The identification and involvement of relevant stakeholders is very important to be able to capture the requirements for the interoperability of heterogeneous IoT platforms [19].

The stakeholders' analysis was carried out through interviews with stakeholders, following an INTER-IoT product oriented approach. With this analysis, we have identified the initial needs for the five components of the project: INTER-LAYER, INTER-FW, INTER-METH, INTER-LogP and INTER-Health. The identification of stakeholders also helped us to start developing cooperation with them, to focus on the requirements gathering process and, ultimately, to ensure a successful outcome for the project. The analysis took into account both demand and supply points of view and both qualitative and quantitative aspects [20].

Although there could be other kind of stakeholders, the following list of potential classes of stakeholders was initially defined for the identification process carried out by all the partners in the consortium:

1. Client/sponsor	10. Representatives of external associations
2. Customer	11. Business analysts
3. Subject-matter experts	12. Designers and developers
4. Members of the public	13. Testers
5. Users of the current system	14. Systems engineers
6. Marketing experts	15. Software engineers
7. Legal experts	16. Technology experts
8. Domain experts	17. System designers
9. Usability experts	

As far as the European Commission (EC) is the sponsor of INTER-IoT, as it is the funding entity of the project, it represents one of the most relevant stakeholders and we established the optimal value for it. Although the EC influenced on the outcomes of the project, it is not the customer of the project's products and results.

To easily identify the stakeholders of each product, we have used a stakeholder's map as it has been described in the publication of "Mastering the Requirements Process: Getting Requirements Right" used as a reference for the VOLERE methodology. The stakeholder's map shows the organizational rings surrounding the product and the classes of stakeholders who inhabit on these rings. The stakeholder map

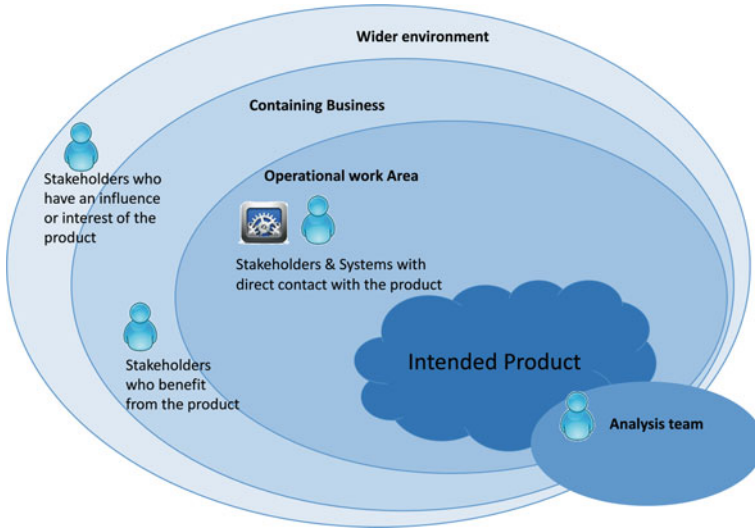


Fig. 1 Stakeholders' map template

determines which classes of stakeholders are relevant to the project and which roles are needed to represent them.

Each product being considered in INTER-IoT has a stakeholders' map representation in this document. The picture below shows the stakeholders map template.

At the centre of the stakeholder map is the intended product (i.e. INTER-LAYER, INTER-FW, INTER-METH, INTER-LogP, INTER-Health). Surrounding the intended product is a ring representing the operational work area—stakeholders who have some direct contact with the product. In the next ring, the containing business include the stakeholders who benefit from the product in some way, even though they are not in the operational area. Finally, the outer ring, the wider environment, contains other stakeholders who have an influence on or an interest in the product. Note also the detailed and multiple involvement of the core team members is emphasized by the fact they span all the rings (Fig. 1).

Each stakeholder identified for each product has been characterised through the template (shown in Fig. 2) designed for the project. This template has helped to easily identify, classify and manage potential stakeholders. The process of characterising the stakeholders has helped to start developing cooperation between these stakeholders and the project team.

The stakeholder's template has also helped to identify other stakeholders who may also be involved in the product design and implementation. The file has also helped to identify existing products and systems being used, produced or provided by the stakeholders related with the INTER-IoT products as well as new products


Product Name: <i>Name of the product analysed (INTER-LAYER, INTER-FW, INTER-METH, INTER-LogP, INTER-Health)</i>		
Stakeholder's Name: <i>Name of the stakeholder</i>	Stakeholder's Acronym: <i>For inclusion in the map</i>	
Stakeholder's Profile & Role: Profile: <i>Stakeholder's profile</i> Role: <i>Description of the role within the product</i>		
Contact Person: <i>Stakeholders' contact</i>	Email: <i>Stakeholder contact's e-mail</i>	Position: <i>Contact position</i>
Stakeholder's Class: <i>Sample list provided above</i>	<input type="checkbox"/> Can appear in public reports <input type="checkbox"/> Shall remain anonymous	<input type="checkbox"/> IoT Demand side <input type="checkbox"/> IoT Supply side
Stakeholder's Needs: <i>Description of the needs of the stakeholder for the Inter-IoT product analysed</i> <input type="checkbox"/> Interested in participate in INTER-IOT open calls		
Existing Products & Systems involved: <i>Identification of existing products and adjacent systems of the product</i>		New products & Systems required: <i>Identification of additional products and systems required for the introduction of the product</i>
New Stakeholders <i>New stakeholders suggested or required for the design and implementation of the product to comply with the needs identified</i>		Stakeholder's class <i>Class of the new stakeholders identified</i>
Reason of involvement: <i>Why the stakeholder has been identified</i>		Identified by: <i>Partner who has identified the stakeholder</i>
		Registration Date: <i>Date of registration</i>

Fig. 2 Stakeholder template

or systems which might be required before or during the adoption of INTER-IoT solutions. Many of the products identified during the stakeholder analysis have been considered in the market analysis.

2.2 Analysis

The stakeholders' analysis has been carried out through an INTER-IoT product-oriented approach [18], as stakeholders have been identified separately with regard to each product (INTER-LAYER, INTER-FW, INTER-METH, INTER-LogP and INTER-Health). The addressed stakeholders are interested from one product to all. From the data gathered in the stakeholder form, we carried out an analysis by product. Therefore, for each of the five products we analysed the following topics:

- Stakeholders by company type (research institutions, public authorities, non-profit organizations, private companies, etc.)
- Stakeholders by country
- Stakeholders map
- Stakeholders by class (client, system, engineers, software engineers, IoT operators, etc.)
- Stakeholders by IoT Demand/Supply
- Stakeholders with interest in Open Call participation
- Products identified by Stakeholders
- Stakeholders needs.

During the first phase of the project, the partners identified and interviewed 93 stakeholders. For each of them one or more stakeholder forms depending on their interest in the different INTER-IoT products.

One of the most interesting conclusions is the type of stakeholders with interest in the project. In Fig. 3 can be seen that a quarter of the stakeholders were potential clients or customers of the INTER-IoT products. After that are the technology experts, sub-matter experts, system and software engineers which will use the INTER-IoT products developing new features and updating the last versions.

For each product, we completed a stakeholder map, like the one in the Fig. 4 for INTER-LAYER. Stakeholders are classified by how involved they are in the design, development and execution of the product. There are four main areas in this map, corresponding to the degree of influence that each stakeholder could have:

- **Analysis team:** Is the core team in the design and development of INTER-LAYER. It is comprised by all project partners that lead the project and are directly involved with INTER-LAYER (e.g. UPVLC, PRO, XLAB, VPF, etc.).
- **Operational work area:** Every stakeholder that has direct a contact with INTER-LAYER (e.g. SigFox, ValenciaPort PCS), has enough knowledge in the product (e.g. AFT) or has a main role in the development and execution of INTER-LAYER (e.g. NOATUM, ASL TO5) fall under this ring.
- **Business area:** Stakeholders that are affected in some way by INTER-LAYER (but not enough to have a main role) are located in this ring. Some stakeholders are interested in contributing to INTER-LAYER (e.g. Infoport, ETRA) or testing and adopting INTER-LAYER (e.g. Valencia Port Authority). Their business models may influence the business models developed within INTER-IoT regarding INTER-LAYER.

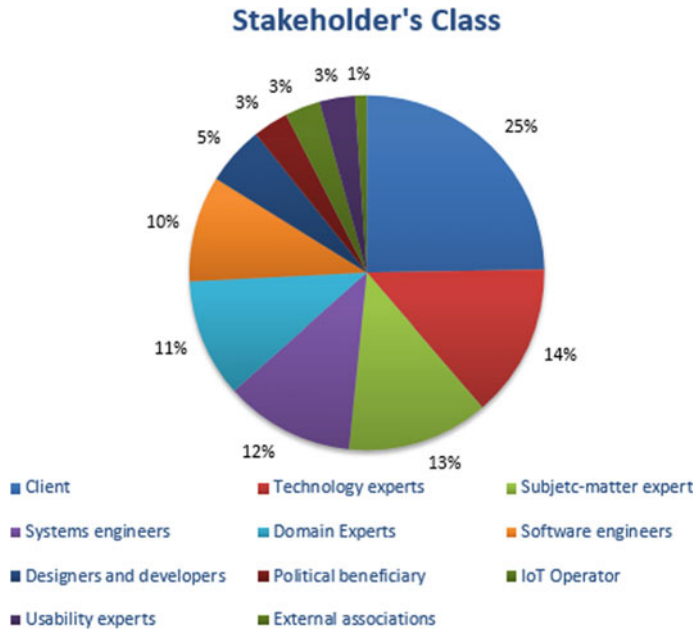


Fig. 3 Total stakeholders by class

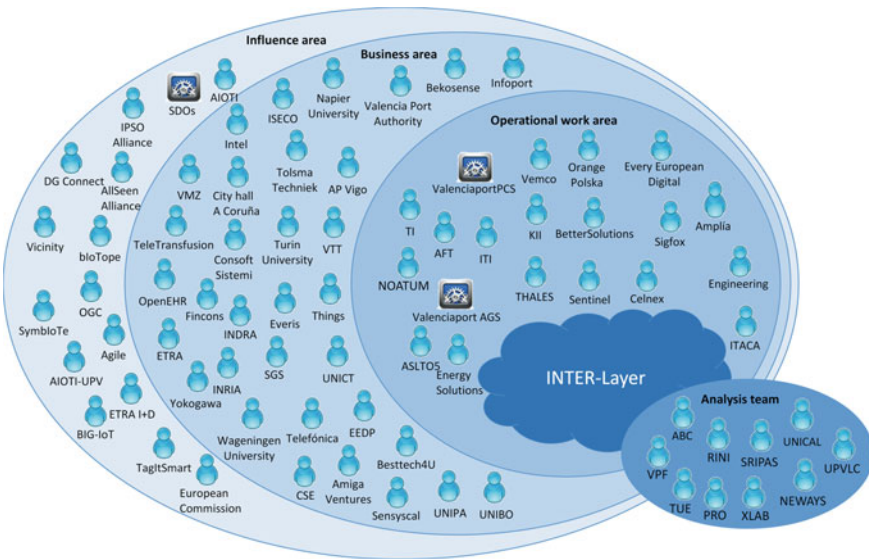


Fig. 4 INTER-LAYER stakeholder's map

- **Influence area:** In the outer ring, stakeholders that have an influence or some interest with INTER-LAYER are located. Other IoT related projects (e.g. BIG-IoT, Vicinity), IoT alliances (e.g. AIOTI, AllSeen), I+D companies (e.g. ETRA I+D), etc. Standardization organizations also play an important role in this ring, as the developed INTER-LAYER product should follow or be in line with the recommendations and working groups implied in these bodies [21].

The whole list of stakeholders and a complete analysis can be found in the INTER-IoT deliverable D2.1 INTER-IoT Stakeholders and market analysis report [22].

3 Market Analysis

3.1 Definition

The aim of the market analysis [23] is to provide an insight of the current IoT market landscape and the vision of different technologies supporting it [24]. The market analysis process is a must to do task in order to identify products that are being introduced or are already in the market which are related with the project in one of the following ways [23]:

- as a component or module of the solution
- as a complementary product
- as a beneficiary, client or consumer of the solution
- as a concurrent product

The products identified during the stakeholders analysis need to be taken into account to identify characteristics, capabilities, objectives and needs of these products under the point of view of interoperability of heterogeneous IoT platforms. The market analysis has been done in combination with the stakeholders analysis, this has allowed us to identify the readiness and willingness of different stakeholders to participate in interoperable IoT scenarios, different systems and products that could be involved and systems and products that would be required to participate in those scenarios [23].

This process has been quite relevant, as we have identified that many existing products are not yet ready to participate in an interoperable IoT environment and they need to be transformed and complemented with other components like IoT gateways and platforms to meet the interoperability requirements. This represents a new market niche, as there do not exist yet a wide adoption of IoT aware solutions and interoperable IoT products. The market analysis also helps to identify relevant standards and protocols that products are supporting and that INTER-IoT products would need to assess.

The Fig. 5 shows the product's template, which includes the name of the product; the product class; the acquisition or licence options; references or web addresses to access further information; a brief description and the services provided by the


Market Analysis			
Product's Name: <i>Name of the identified product</i>			
Product Class: <i>Hardware, Software, Methodology, Platform, Standard ...</i>	Context: <i>Local, national, European, international, ...</i>	Access mode: <i>Open, Close, subscription, license, ...</i>	
Web address:		(Logo)	
Product Description: <i>Brief description of product</i>			
Product Services: <i>Main services of product</i>			
Links and Documents: <i>Useful links</i>			
Reason of involvement: <i>Partner project</i>	Related to IoT Product: <i>Name of the IoT product associated (INTER-LAYER, INTER-FW, INTER-METH, INTER-LogP, INTER-Health)</i>	Identified by: <i>Partner who has identified the stakeholder</i>	Registration Date: <i>Date of registration</i>

Fig. 5 Product template

product. The file also records the partner who has identified the product and the reason why it is registered, including its relation with any of the INTER-IoT products.

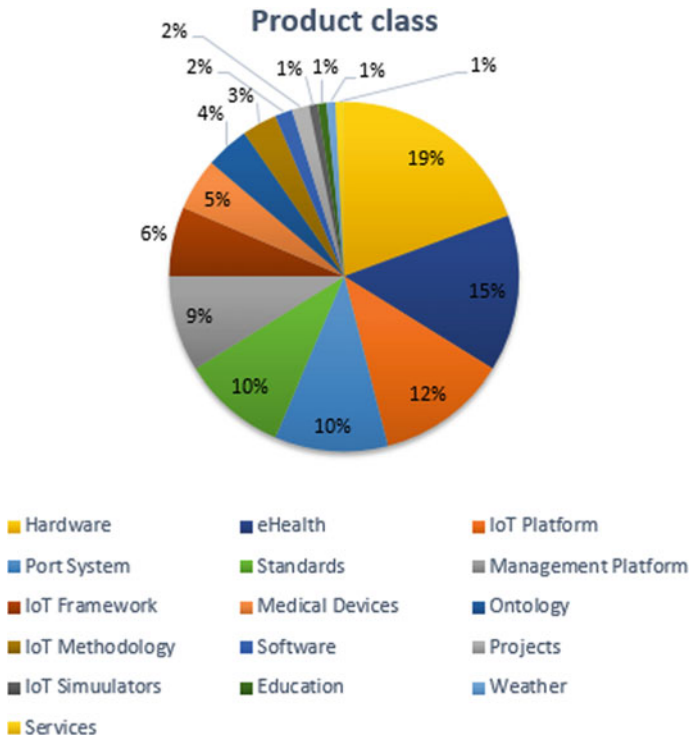


Fig. 6 Products by class

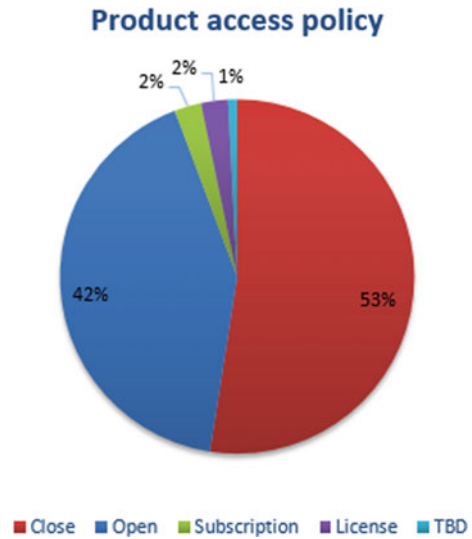
3.2 Analysis

The market analysis was performed mainly through desk research, workshops, market studies, report analysis and consultation with IoT market experts, including our participation in IoT-EPI and AIOTI forums. The market analysis comprised existing solutions and trends, including vendor specific solutions, research projects and existing and proposed standards. We analysed more than 110 representative products, although we were aware than a high spread of products exist. We classified the products in 16 different domains, which can be seen in Fig. 6.

The most prominent effect of Fig. 6 is the fact it shows a high level of heterogeneity, with many product classes accounting for very low portions of the overall product class spread (e.g. hardware, standards, simulators, IoT platforms etc.). There is no general rule that can emerge from this view though other than the fact that IoT educated stakeholders' awareness of existing products is quite scattered.

However, another interesting fact emerges, showing that the quantity of known products can be closely tied to domain specific classes. The figure above indeed shows that 20% of products identified are linked to either the medical sector (15% to eHealth, 5% to medical devices) or to port systems (10%). This observation enables

Fig. 7 Products by access policy



us to assume there is a high level of domain application specialization of the products being identified.

Other conclusion extracted from the analysis is product access policy. It allows to a certain degree to acquire a clearer idea of the economic and administrative constraints to overcome in order to be able to use the products. The access mode also provides a certain indication as to the structural interoperability the products bear.

Figure 7 confirms the need for INTER-IoT solutions as over half of the identified products are used in a closed environment mode, thus hindering the economic efficiency and competitive advantages that could be gained by fostering interoperability. Luckily, among those products that can theoretically be accessed more openly, few require to overcome additional hurdles such as subscription or licensing.

The whole list of products and a complete analysis can be found in the INTER-IoT deliverable D2.1 INTER-IoT Stakeholders and market analysis report [22].

4 Requirements Analysis

4.1 Definition

The set of requirements define how should work the different products from the needs of end users, suppliers and developers. A rigorous assessment of requirements before design allows a clear idea of what you want to implement and reduces delays due to design flaws. It also allows estimating more accurately the costs and the risks [3].

The requirements are the basis for the design stage, so a well-defined requirement reduces the development effort [16]. During the specification of the requirements, we should involve almost all the departments of the organization in order to define the necessary requirements for a specific product or service [25]. A complete and correct requirement process reduces the effort wasted on redesign, recoding and retesting. It also provides an efficient mechanism for the product validation and verification [3].

The characteristics that the requirements should have following ISO 2011 are: necessary, appropriate, unambiguous, complete, singular, feasible, verifiable, correct, consistent and comprehensible. There are several requirements categorizations, but the Volere methodology categorises requirements into three main groups⁴:

- Functional requirements are the fundamental subject matter of the system and are measured by concrete means like; data values, decision-making logic and algorithms.
- Non-functional requirements are the behavioural properties that the specified functions must have, such as performance, usability, etc. Non-functional requirements can be assigned to a specific measurement. The methodology also includes a rich catalogue of non-functional requirements to be taken into account, which will be reviewed afterwards.
- Project constraints identify how the eventual product must fit into the world. For example, the product might have to interface with or use some existing hardware, software or business practice, or it might have to fit within a defined budget or be ready by a defined date.

In the project, we followed 5-step iterative process for identifying, capturing, defining, analysing, and reconciling requirements (see Fig. 8). As it is an iterative process, during the whole project the requirements were reviewed and redefined to be more focused on the real development of the different components [26].

4.1.1 Identify Sources of Requirements

The partners identified the sources of information to collect requirements such as previous research projects, partners' knowledge, stakeholders, regulation, standards, etc.

4.1.2 Requirement Capturing

This step generated an inventory of identified requirements by INTER-IoT product, including requirement name and brief description.

⁴ Volere Requirements Specification Template https://www.st.cs.uni-saarland.de/edu/se/2009/slides/volere_specification_template_v6.pdf.

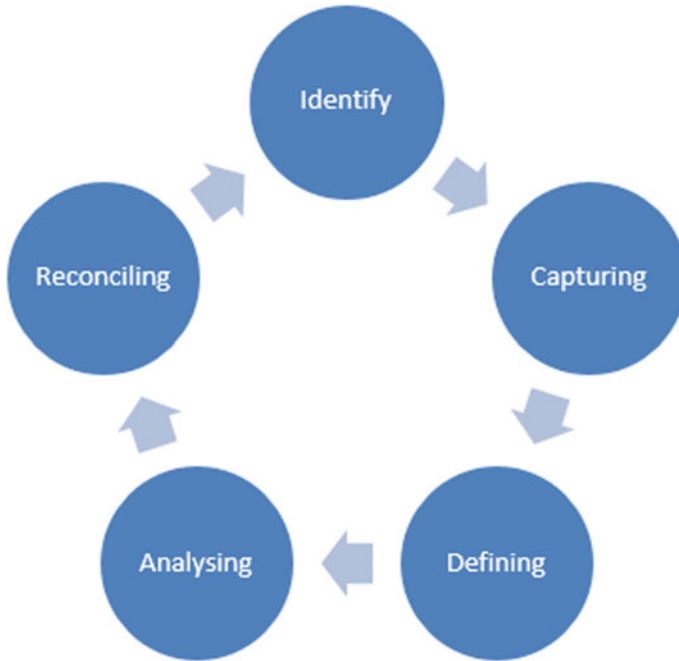


Fig. 8 Requirements capture methodology

4.1.3 Defining

This step produced a detailed requirement specification following the requirement template and taking into account the characteristics of good requirements specification. A requirement template was defined with the main information needed in order to be collected from the requirements identified (see Fig. 9).

4.1.4 Analysing

This phase consisted in assessing the requirements obtained. For the analysis and assessment of requirements, we created five task forces composed of the different partners. These task forces produced the following improvements in the requirements:

- Improving the quality of the description
- Correcting and homogenizing the relevant classifications
- Grouping similar requirements
- Validating the requirements
- Detecting new requirements not identified in other sources of information.

Requirements			
Requirement's Name: <i>Name of the identified requirement</i>		Identifier: #1	
Category: <i>Functional, Non-functional, or Design constraints</i>	Type: <i>Security, Privacy, QoS, Semantics, Resilience, Interoperability, Data model, Communications, Commercial, Operational ...</i>	Priority: <i>Must, May or Nice to have</i> MoSCoW Priority: <i>Must have, Should have, Could have or Won't have</i>	Status: <i>Approved or Out of scope</i>
Product: <i>INTER-LAYER, INTER-FW, INTER-METH, INTER-LogP, INTER-Health</i>	Affected Layer: <i>In the INTER-Layer product</i>	Scenario: <i>Involved scenario (in case of business pilot)</i>	
Rationale: <i>Reason of involvement</i>			
Requirement Description: <i>Brief description of the requirement</i>			
Acceptance criteria: Conditions that requirement must satisfy to be accepted			
Source: <i>EU project, One-M2M, IOT-A, Partner's expertise, ...</i>	Identified by: <i>Partner who has identified the requirement</i>	Registration Date: <i>Date of registration</i> <i>Date of update</i>	

Fig. 9 Requirement template

4.1.5 Reconciling

This was the final step in which there were an agreement among the partners to incorporate the requirement into the definitive list.

When implementing the functionality of a system it is important to prioritize the requirements [17] to first develop the essential parts and remove the less significant ones if necessary due to lack of time or resources [25].

In INTER-IoT, the requirements have two priority types. On the one hand are the initial needs of stakeholders and the final users of the products. On the other hand, it is necessary to prioritize what is essential for the operation of the product for the development [16, 17]. The prioritization technique that has been used as a reference to prioritizing the requirements is MoSCoW [27, 28].

The MoSCoW method [28] is a prioritization technique [17] used in management, business analysis, project management, and software development to reach a common understanding with stakeholders on the importance they place on the delivery of each requirement - also known as MoSCoW prioritization or MoSCoW analysis [27]. The prioritization categories are typically understood as: must have, should have, could have, and won't have [28].

4.2 Analysis

From the interviews with the stakeholders during the first stage of the INTER-IoT project, we could extract their main needs. We also did a thorough analysis of existing products on the market. All this information was used to start the identification of requirements, together with the knowledge of the partners and some regulations and standards.

In first stage, we defined 201 requirements. However, since the requirements are an iterative process that takes place throughout the duration of the project, during the development tasks the number of requirements was reduced to 185.

Analysing the type of requirements, we can see in Fig. 10 that there are a large number of requirements about security, privacy and interoperability. This is due to INTER-IoT is a framework for the interoperability of platforms, so some requirements are focused on these issues. In some products there are types of requirements more numerous such as communications in INTER-LAYER, API in INTER-FW, methodology in INTER-METH, or operational in the application domains. There are also other relevant types such as semantics, architecture, usability, etc.

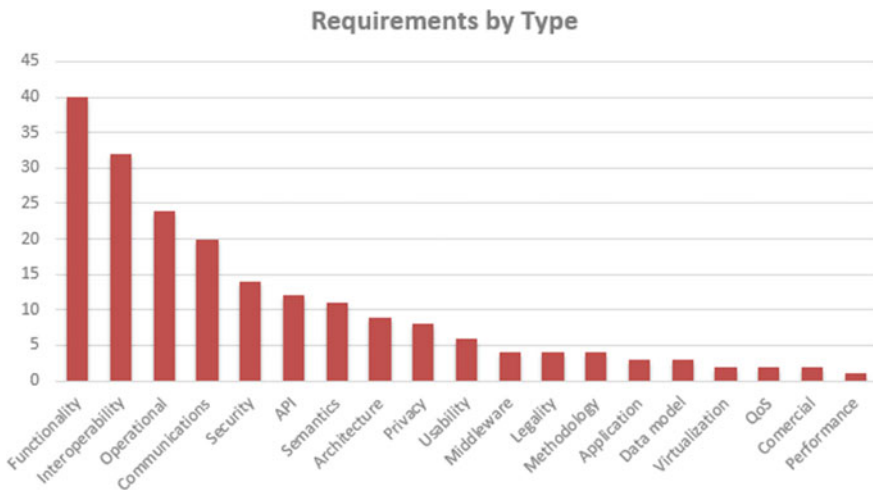
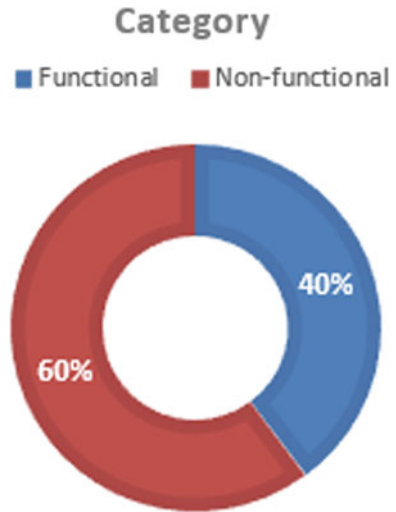


Fig. 10 Requirements by type

Fig. 11 Requirements by category



From the results obtained, we can note that more than half of the requirements are non-functional (Fig. 11). This may be because the project does not attempt to develop a product or platform, but a framework for interoperability between platforms [29]. Therefore, many of the requirements describe system characteristics and features that should be provided. We have also two application domains (INTER-LogP and INTER-Health) where there are most of the functional requirements, since in this case we try to develop a more specific product.

Concerning the requirements priority, there are two kind of classification (Figs. 12 and 13). First, in the classification from the needs of the stakeholders, where approximately two thirds are mandatory, and only 10% are interesting to be included.

Otherwise, in the requirement prioritization based on the developers' criteria, around 40% are high priority. This is because we have identified the essential features and functionalities that the different products must have. Furthermore, we have also defined other requirements that might be interesting to have (39%).

The main sources of data we have taken into account when defining the requirements have been the stakeholders' needs and the partner's expertise. Nevertheless, it has also been quite important to consider other sources such as IoT associations and projects (IOT-A, AIOTI, etc.), standardization organizations recommendations (IEEE, ITU, ISO, etc.) and national and European regulations.

The whole list of requirements and a complete analysis can be found in the INTER-IoT deliverable D2.3 INTER-IoT Requirements and business [30] (Fig. 14).

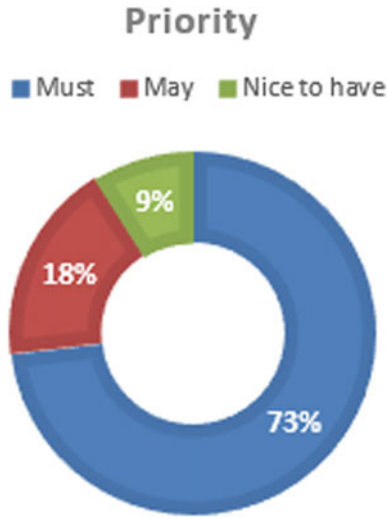


Fig. 12 Requirements by priority

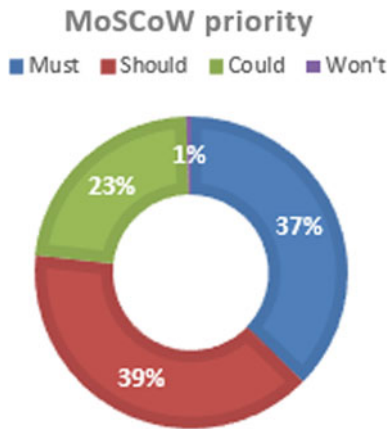


Fig. 13 Requirements by MoSCoW priority

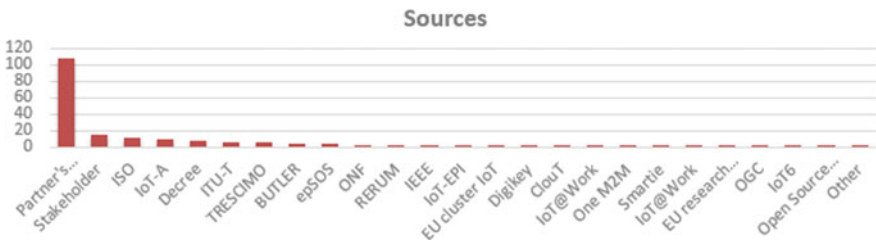


Fig. 14 Requirements by source

5 Conclusions

For any product, design, or project is necessary to begin by establishing what we want to achieve. For that aim, we must perform a thorough analysis of what is on the market and what our customers may need. With this information, we have to specify in detail the features and functionality of our product, which is reflected in the requirements.

The requirements are used to establish the basis for agreement between the customers and the suppliers on what the software product is intended to do. Once known the needs of customers, it is easier to develop a successful business model.

Thus, that is the main reason to not assume customer's needs and to perform a thorough analysis to find them out. It is important to involve the end users as soon as possible and to get in touch with the actual people that will be using the product on a daily basis as they are the ones that truly understand what is needed.

In the INTER-IoT project, we identified 93 stakeholders interested in the INTER-IoT products and we performed a market analysis with 110 products. All this data was used to define a set of requirements used to design and implement INTER-LAYER, INTER-FW and INTER-METH in their respective tasks.

One of the main conclusions extracted from the stakeholders' needs and the market analysis performed is the existence of a critical necessity for interoperability between IoT platforms. The need for interoperability appears in terms of new standards being proposed and in the interest from developers and designers to add to their products new plugins, services and connectors allowing the interoperability with open platforms like FIWARE, sensiNact or universAAL.

References

1. Wiegers, K., Beatty, J.: Software Requirements. Pearson Education, August 2013. Google-Books-ID: nbpCAwAAQBAJ. <https://ptgmedia.pearsoncmg.com/images/9780735679665/samplepages/9780735679665.pdf>
2. Vermesan, O., Friess, P. (eds.) Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds. Riverpublishers (2016). https://www.riverpublishers.com/pdf/ebook/RE_E9788793379824.pdf
3. Robertson, S., Robertson, J.: Mastering the Requirements Process: Getting Requirements Right. Addison-Wesley (August 2012). Google-Books-ID: yE91LgrpaHsC. https://books.google.es/books/about/Mastering_the_Requirements_Process.html?id=yE91LgrpaHsC&redir_esc=y
4. Fortino, G., Savaglio, C., Palau, C.E., Suarez de Puga, J., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: Towards multi-layer interoperability of heterogeneous IoT platforms: The INTER-IoT approach. *Internet of Things*, 0(9783319612997), 199–232 (2018)
5. Pattar, S., Buyya, R., Venugopal, K.R., Iyengar, S.S., Patnaik, L.M.: Searching for the IoT resources: fundamentals, requirements, comprehensive review, and future directions. *IEEE Commun. Surv. Tutor.* **20**(3), 2101–2132 (2018). Conference Name: IEEE Communications Surveys Tutorials
6. Palade, A., Cabrera, C., Li, F., White, G., Razzaque, M.A., Clarke, S.: Middleware for Internet of Things: an evaluation in a small-scale IoT environment. *J. Reliab. Intell. Environ.* **4**(1), 3–23 (2018)

7. Mi Jung, H., Jeong, K., Jin Cho, H.: A Design for security functional requirements of IoT middleware system. *J. Korea Converg. Soc.* **8**(11), 63–69 (2017). Publisher: Korea Convergence Society
8. Byers, C.C.: architectural imperatives for fog computing: use cases, requirements, and architectural techniques for fog-enabled IoT networks. *IEEE Commun. Mag.* **55**(8), 14–20 (2017)
9. Mazayev, A., Martins, J.A., Correia, N.: Interoperability in IoT Through the Semantic Profiling of Objects. *IEEE Access* **6**, 19379–19385 (2018)
10. Xhafa, F., Kilic, B., Krause, P.: Evaluation of IoT stream processing at edge computing layer for semantic data enrichment. *Futur. Gener. Comput. Syst.* **105**, 730–736 (2020)
11. Ahmed, A.I.A., Gani, A., Hamid, S.H.Ab., Abdelmaboud, A., Syed, H.J., Mohamed, R.A.A.H., Ali, I.: Service management for IoT: requirements, taxonomy, recent advances and open research challenges. *IEEE Access* **7**, 155472–155488 (2019)
12. Milovanovic, D.: Cloud-based IoT healthcare applications : requirements and recommendations. *Int. J. Internet Things Web Serv.* (2017)
13. Oh, S., Kim, Y.: Security requirements analysis for the IoT. In: 2017 International Conference on Platform Technology and Service (PlatCon), pp. 1–6 (February 2017)
14. Hameed, S., Idris Khan, F., Hameed, B.: understanding security requirements and challenges in Internet of Things (IoT): a review. *J. Comput. Netw. Commun.* 2019, e9629381 (2019). Publisher: Hindawi
15. Santarremigia, F.E., Poveda-Reyes, S., Hervás-Peralta, M., Molero, G.D.: A decision-making method for boosting new digitalization technologies. *Int. J. Inf. Technol. Decis. Mak.*, 1–35, February 2021. Publisher: World Scientific Publishing Co
16. Elijah, J., Mishra, A., Udo, M.C., Abdulganiyu, A., Musa, A.: Survey on requirement elicitation techniques: it's effect on software engineering. *Int. J. Innov. Res. Comput. Commun. Eng.* **5**(5), 15 (2007)
17. Hudaib, A., Masadeh, R., Qasem, M., Alzaqebah, A.: Requirements prioritization techniques comparison. *Mod. Appl. Sci.* **12**(2), 62 (2018). Number: 2
18. Brugha, R., Varvasovszky, Z.: Stakeholder analysis: a review. *Health Policy Plan.* **15**(3), 239–246 (2000)
19. Ganzha, M., Paprzycki, M., Pawłowski, W., Szmaja, P., Wasielewska, K.: Semantic interoperability in the Internet of Things: an overview from the INTER-IoT perspective. *J. Netw. Comput. Appl.* **81**, 111–124 (2017)
20. Pileggi, S.F., Palau, C.E., Esteve, M.: Building semantic sensor web: knowledge and interoperability. In: Proceedings of the International Workshop on Semantic Sensor Web—Volume 1: SSW, (IC3K 2010), pp. 15–22. INSTICC, SciTePress (2010)
21. Broring, A., Zappa, A., Vermesan, O., Främling, K., Zaslavsky, A., Gonzalez-Usach, R.P., Szmaja, C. Palau, M. Jacoby, I. P. Zarko, S. Sour- sos, C. Schmitt, M. Plociennik, S. Krco, S. Georgoulas, I. Larizgoitia, N. Gligoric, R. García-Castro, F. Serena, V. Orav. *Advancing IoT Platform Interoperability*. River Publishers, The Netherlands, 2018
22. Broring, A., Zappa, A., Vermesan, O., Främling, K., Zaslavsky, A., Gonzalez-Usach, R., Szmaja, P., Palau, C., Jacoby, M., Zarko, I.P., Soursos, S., Schmitt, C., Plociennik, M., Krco, S., Georgoulas, S., Larizgoitia, I., Gligoric, N., García-Castro, R., Serena, F., Orav, V.: D2.1. Stakeholders and Market Analysis Report vol 1.1. INTER-IoT H2020 project, March 2016
23. Day, G.S.: Strategic market analysis and definition: an integrated approach. *Strat. Manag. J.* **2**(3), 281–299 (1981). <https://onlinelibrary.wiley.com/doi/pdf/10.1002/smj.4250020306>
24. Ilin, I.V., Izotov, A.V., Shirokova, S.V., Rostova, O.V., Levina, A.I.: Method of decision making support for it market analysis. In: 2017 XX IEEE International Conference on Soft Computing and Measurements (SCM), pp. 812–814, Saint Petersburg, Russia, May 2017. IEEE
25. Mougouei, D., Powers, D.M.W.: Modeling and selection of interdependent software requirements using fuzzy graphs. *Int. J. Fuzzy Syst.* **19**(6), 1812–1828 (2017)
26. Giménez, P., Molina, B., Palau, C.E., Esteve, M.: Swe simulation and testing for the iot. In: 2013 IEEE International Conference on Systems, Man, and Cybernetics, pp. 356–361 (2013)
27. Ahmad, K.S., Ahmad, N., Tahir, H., Khan, S.: Fuzzy Moscow: A fuzzy based Moscow method for the prioritization of software requirements. In: 2017 International Conference on Intelligent Computing, Instrumentation and Control Technologies (ICICT), pp. 433–437 (2017)

28. Miranda, Eduardo: Time boxing planning: buffered Moscow rules. *ACM SIGSOFT Softw. Eng. Notes* **36**(6), 1–5 (2011)
29. Fortino, Giancarlo, Garro, Alfredo, Russo, Wilma: Achieving mobile agent systems interoperability through software layering. *Inf. Softw. Technol.* **50**(4), 322–341 (2008)
30. Fortino, G., Garro, A., Russo, W.: D2.3. Requirements and Business Analysis, vol. 2. INTER-IoT H2020 project (January 2017)

INTER-IoT Architecture for Platform Interoperability



Alessandro Bassi, Miguel A. Llorente, Miguel Montesinos,
and Raffaele Gravina

Abstract The number and diversity of IoT platforms is huge, causing an unsustainable ecosystem fragmentation. INTER-IoT proposes a solution to bridge heterogeneous IoT platforms, promoting interoperability at different abstraction layers. Modeling is a key engineering mechanism to abstract existing platforms requirements and functionalities, to capture common, reusable properties, so to realize general approaches that are capable of addressing different degrees of interoperability. Based on the significant results of the IoT-A EU funded project, INTER-IoT has therefore defined a Reference Model for IoT Platforms Interoperability, a Reference Architecture, and a complete interoperability system.

1 Introduction

The focus of INTER-IoT is to bridge different IoT platforms and make them fully interoperable at different levels [1]. The number of solutions at time of writing is simply unsustainable, heading towards 500 different platforms. Even looking only at the “big players”, there is not a clear winner or a superior set of solution, and this situation is not likely to change in the near future.

In such fragmented ecosystem, there is no doubt that being able to use the same modeling for different system is the first important step towards true interoperability. Modeling is a valuable mechanism to abstract commonalities of existing platforms, extracting the main features that define the IoT domain and to build general approaches to face the interoperability in a universal way.

For this reason, INTER-IoT has defined a Reference Model for IoT Platforms Interoperability, a Reference Architecture based on this model and a complete interoperability system.

A. Bassi (✉)
ABC Consulting, Prague, Czech Republic
e-mail: alessandro@bassiconsulting.eu

M. A. Llorente · M. Montesinos
Prodevelop, Valencia, Spain

R. Gravina
DIMES, University of Calabria, Rende, Italy

© Springer Nature Switzerland AG 2021

C. E. Palau (eds.), *Interoperability of Heterogeneous IoT Platforms*, Internet of Things,
https://doi.org/10.1007/978-3-030-82446-4_3

2 INTER-IoT Reference Model

The INTER-IoT project adopted the OASIS definition for describing the reference model. As a reminder, OASIS (Organization for the Advancement of Structured Information Standards) gives the following definition of a reference model:

[Reference model is] 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 [2].

Central features that a reference model needs to exhibit are:

- Clearly defined concepts
- Clearly defined concept relationships
- Clearly defined concept properties
- Does not describe concrete entities (instances)
- Restricted to a specific problem space
- Promotes understanding of the problem space
- Technology agnostic and implementation independent
- Used as reference for implementation
- Enables understanding (i.e. common semantics) in communication.

The INTER-IoT Reference Model (RM) description and fundamentals are based in the successful results of the IoT-A project [3], however, unlike the RM proposed in IoT-A [4], 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 discussed in this chapter proposes a reference model for IoT interoperability mechanisms in real scenarios.

2.1 Domain Model

In general, a Domain model is a class diagram that is used to describe specific aspects of a set of knowledge or activities. The main use for it is to represent use cases and real-world concepts in a way that can then be used by the technical people to develop

a service, application or a product. The main purpose of a generic domain model is to generate a common understanding of the target domain in question. Such a common understanding is important, not only within a specific project, but also to be able to discuss with stakeholders and external parties. Only with a common understanding of the main concepts it is possible to choose between different architectural solutions and to evaluate them.

2.1.1 The IoT-A Domain Model

The IoT-A project defines a domain model [3] as a description of concepts belonging to a specific area of interest. The domain model also defines basic attributes of these concepts, such as name and identifier, and relationships between concepts, for instance “Services expose Resources”. The IoT-A domain model also provides a common lexicon and taxonomy that can be used in the IoT domain. In an IoT domain, the most generic scenario is that of a generic User who needs to interact with a Physical Entity (PE) in the physical world. Here we can already see two of the main entities in IoT:

- a *User* which can be a human person or a Digital Artifact (e.g., a Service, an application, or a software agent) that needs to interact with;
- a *Physical Entity*, which is an object that is under observation and can be modified by automatic means. Physical Entities can be almost any object or environment; from humans or animals to cars; from store or logistics chain items to computers; from electronic appliances to clothes.

While in a physical environment, interactions can only happen directly (for instance, by moving a pallet from location X to Y manually), within the IoT world it is possible to interact indirectly, through a “mediator”. The mediator can be a Service that will either provide information about the Physical Entity or act on it. When a Human User is accessing a service, he/she does so through a service client, a User Interface for instance. For the scope of the IoT Domain Model, the interaction is usually characterised by a goal that the User pursues. Physical Entities are represented in the digital world by a Virtual Entity, which can be seen as a “virtual counterpart”. There are many kinds of digital representations of Physical Entities: 3D models, avatars, database entries, objects (or instances of a class in an object-oriented programming language). Virtual Entities are associated to a single Physical Entity and the Virtual Entity represents this very Physical Entity. While there is generally only one Physical Entity for each Virtual Entity, it is possible that the same Physical Entity can be associated to several Virtual Entities. Each Virtual Entity must have one and only one ID that identifies it univocally. Virtual Entities are Digital Artefacts that can be classified as either active or passive. Active Digital Artefacts (ADA) are running software applications, agents or Services that may access other Services or Resources. Passive Digital Artefacts (PDA) are passive software elements such as database entries that can be digital representations of the Physical Entity. Please

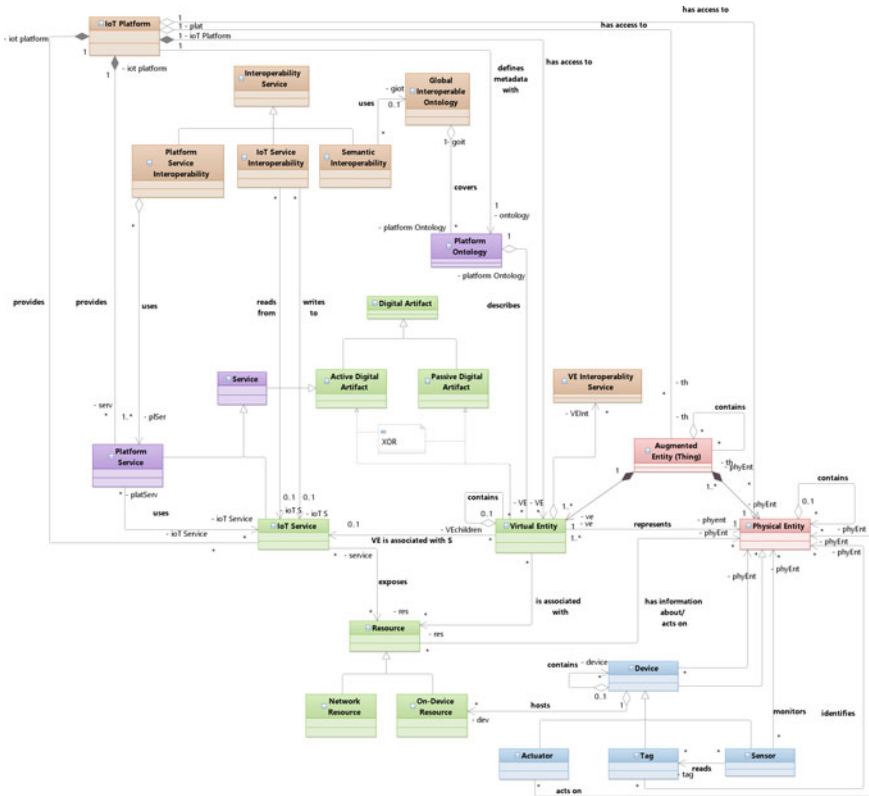


Fig. 1 INTER-IoT generic domain model

note that all Digital Artefacts can be classified as either Active or Passive Digital Artefacts.

2.1.2 The INTER-IoT Domain Model

The INTER-IoT Domain Model is based upon the Domain model developed by IoT-A [3]. It represents an extension of that model to capture the full interoperability between different systems.

In Fig. 1 the proposed Inter-IoT 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.

The clearest entity is an IoT Platform itself. In any IoT Domain Model, the platform is intrinsically implicit, as it really reflects “the whole model”. When dealing with platform interoperability, different IoT Platforms appear, thus need arises to model them independently. Any IoT Platform relates with the underlying entities, like Services, Things or Physical Entities, and so on. Therefore, an IoT Platform can be modeled as a set of composed entities, the entities the platform manages [5].

An IoT Platform is comprised of several collections of entities:

- **IoT Service:** An IoT Service is equivalent to a Service in the IoT-A Model. It provides an open and standardised interface, offering all necessary functionalities for interacting with the Resources/Devices associated with Physical Entities through Virtual Entities. They are Active Digital Artifacts such as query, update, or subscribe.
- **Platform Service:** A Platform Service is an Active Digital Artifact that exposes functionalities about Resources related to Virtual Entities. However, rather than being attached to specific Physical Entities (and its related Virtual Entities), they offer more elaborated services built on IoT Services. Therefore, they belong to higher layers of an IoT Architecture, allowing more complex processing like CEP (Complex Event Processing), Stream Processing, Historical Data Management, or Monitoring. They are similar to IoT processes in the IoT-A Model.
- **Virtual Entity:** A Virtual Entity is a representation of a Physical Entity in the digital world. IoT platforms tend to use this digital representation, especially those based on cloud platforms, regardless to the fact whether it is connected to the Internet or not.
- **Physical Entity:** A Physical Entity is an object or environment of interest for an external user, application, or service. It is something that can be observed and connected to one or more IoT Platforms to interact with it.
- **Augmented Entity:** As of IoT-A definition, an Augmented Entity is “the composition of one Virtual Entity and the Physical Entity it is associated to, in order to highlight the fact that these two concepts belong together. The Augmented Entity is what actually enables everyday objects to become part of digital processes, thus, the Augmented Entity can be regarded as constituting the ‘thing’ in the Internet of Things.”

A new concept is the **Platform Ontology**. It is conceived to store the definition concerning its inner structure and different components such as devices, or sensors. The Platform Ontology defines also the ontology used for modelling the observations made by each sensor, in our case available from the Virtual Entities. These ontologies will usually be different for each Physical Entity type (and its related Virtual Entity). This entity is thus the one that is responsible for handling the corresponding semantics. Once we have modelled the different entities of the platforms and the platform itself, we have added two entities related, specifically, to defining the interoperability services that can be created at different layers. The Platform Interoperability Service handles the definition of new compound services that appear as a consequence of using, and mixing in any way, Platform Services from one or more IoT platforms. An example of this would be the creation of an alert service that may throw an event

when weather sensors from platform A exceed predefined thresholds using a rule engine service at platform A, or when weather sensors from platform B send an alert using a CEP (Complex Event Processing) within platform B. So, the Platform Interoperability Service is linked with the different Platform Services it uses. Used Platform Services are just part of IoT Platforms. From the interoperability point of view, they are the building blocks of more complex interoperability services among different IoT platforms: Platform Interoperability Services. The VE Interoperability Service has a similar role as the Platform Interoperability Service, but it defines interoperability services among devices rather than platform services. It is responsible for defining the interoperability at the device layer, what we call D2D (Device to Device) interoperability at INTER-IoT. The VE Interoperability Service can handle the rules for performing the D2D interoperability. For instance, it could have the definition of a rule triggered when a proximity sensor detects presence, switching a light on.

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 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.

2.2 Information Model

The Information Model is another of the five Models composing the IoT-A Reference Model. The main aspects are represented by the elements *VirtualEntity*, *ServiceDescription* and *Association*. As a *VirtualEntity* models a Physical Entity, a *ServiceDescription* describes a Service that provides information about the Physical Entity itself or the environment. Through an *Association*, the connection between an Attribute of a Virtual Entity and the *ServiceDescription* is modelled; in other words, the Service acts as a “get” function for an Attribute value.

Every Virtual Entity needs to have a unique identifier (identifier) or entity type (entityType), defining the type of the Virtual Entity representation, for instance, a human, a car or a temperature sensor. Furthermore, a Virtual Entity can have any number of different attributes (Attribute class). The entityType of the VirtualEntity class may refer to concepts in an ontology that defines what attributes a Virtual Entity of this type has. Each Attribute has a name (attributeName), a type (attributeType), and one to many values (ValueContainer). The attributeType specifies the semantic type of an attribute, for example, that the value represents temperature. It can reference an ontology-concepts. This way, it is possible to model an attribute or a list of values, which itself has several values. Each ValueContainer groups one Value and zero to

many metadata information units belonging to the given Value. The metadata can, for instance, be used to save the timestamp of the Value, or other quality parameters, such as accuracy or the unit of measurement. The Virtual Entity (Virtual Entity) is also connected to the ServiceDescription via the <Service Description/Virtual Entity> Association.

A ServiceDescription describes the relevant aspects of a Service, including its interface. Additionally, it may contain one or more ResourceDescription(s) describing a Resource whose functionality is exposed by the Service. The ResourceDescription in turn may contain information about the Device on which the Resource is hosted.

According to IoT-A, the IoT Information Model defines the structure of all the information for Virtual Entities on a conceptual level. This description utilizes meta-data coming from appropriate ontologies. The INTER-IoT project uses semantic technologies to deal with meta-level interoperability. Specifically, the semantic interoperability will be established through the use of a modular ontology, ontology alignments, and semantic transformations.

The Inter-IoT reference meta-data model is a set of ontologies, that can also be viewed as one modular ontology, with both horizontal and vertical modules. This ontology needs to cover fundamental concepts in IoT, such as thing, device, observation and deployment.

The meta-data model needs to conform to OASIS guidelines enumerated in Sect. 2. OWL ontologies naturally exhibit some of those, such as: clear (and formal) descriptions of concepts and relationships between them; independence of implementation technology; and enabling common semantics. Other than that, the reference meta-data model does not contain references to any specific instances, and is limited to the scope defined by meta-data requirements [6].

2.2.1 Scope of Meta-Data Model

The scope of the Inter-IoT reference meta-data model is defined in a process that defines meta-data items (entities) that need to be included in the model. Simple examples of meta-data entities are Service, Device (with sub-types Actuator, Tag and Sensor) that are declared in the IoT-A domain model. The meta-data reference model expands those declarations into definitions by defining properties, class attributes, taxonomy and other elements structuring the meta-data entities. Once the scope (defined by the entities) is prepared, the reference model is constructed by choosing and adjusting (expanding or reducing) modular ontologies.

2.2.2 IoT Platforms

Some IoT platforms, like OpenIoT or UniversAAL provide explicit ontologies that model the meta-data used within those platforms. The knowledge contained within those models is an indirect source of meta-data entities. Since INTER-IoT is a set

of tools for interoperability between platforms, rather than a platform itself, the models of platforms should not simply be copied. That being said, the analysis of existing platform ontologies provides valuable insight that augments explicit meta-data requirements from other sources, and puts them in context.

2.2.3 Comparing IoT-Related Ontologies

The space of ontologies is fragmented, regardless of the domain of interest. The richer an ontology is, the larger area it spans. Hence, uniqueness and intersections with other ontologies become more intricate and complex. Internet of Things spans enormous number of domains, and rapidly expands with the growing popularity of “smart devices”. Use of ontologies in the IoT mimics this expansiveness. There are many ontologies that represent models relevant to the IoT, including, but not limited to, devices, units of measurement, data streams, data processing, geolocation, data provenance, computer hardware, methods of communication, etc. We assume that the centrepiece of the IoT is a smart device capable of communication. Therefore, the first iteration of the reference meta-data model is in the form of a device ontology and forms a cornerstone for other ontology modules (that cover other meta-data requirements). The ontologies were selected for analysis based on a simple criterion that they describe some (at least one) of the meta-data requirements identified at the beginning of the Inter-IoT project.

From this perspective, from the identified ones, we have selected ontologies that capture the idea of a device, and are well established in the IoT space: SSN, SAREF, oneM2M Base Ontology, IoT-Lite, and OpenIoT. Each of them takes a different approach to modelling the IoT space but, despite the differences in conceptualization, they cover intersecting fragments of the IoT landscape. Below, we discuss divergence, contrariness and similarities between these ontologies.

SSN, or “Semantic Sensor Network” [7] is an ontology centered around sensors and observations. It is a de-facto extension of the SensorML language. *SSN* focuses on measurements and observations, disregarding hardware information about the device. Specifically, it describes sensors in terms of capabilities, performance, usage conditions, observations, measurement processes, and deployments. It is highly modular and extendable. In fact, it depends on other ontologies in key areas (e.g. time, location, units) and, for all practical purposes, needs to be extended before actual implementation of an *SSN*-based IoT system. *SSN*, formulated on top of *DUL*, is an ontological basis for the IoT, as it tries to cover any application of sensors in the IoT [8].

SAREF [9], or “The Smart Appliances REference” ontology covers the area of smart devices in houses, offices, public places, etc. It does not focus on any industrial or scientific implementation. The devices are characterized predominantly by the function(s) they perform, commands they accept, and states they can be in. Those three categories serve as building blocks of the semantic description in *SAREF*. Elements from each can be combined to produce complex descriptions of multi-functional devices. The description is complemented by device services that offer

functions. A noteworthy module of SAREF is the energy and power profile that received considerable attention, shortly after its inception. SAREF uses WGS84 for geolocation and defines its own measurement units.

oneM2M Base Ontology (*oneM2M BO*; [10]) is a recently created ontology, with first non-draft release in August 2016. It is relatively small, prepared for the release 2.0 of *oneM2M* specifications, and designed with the intention of providing a shared ontological base, to which other ontologies would align. It is similar to the SSN, since any concrete system necessarily needs to extend it before implementation. It describes devices in a very broad scope, enabling (in a very general sense) specification of device functionality, networking properties, operation and services. The philosophy behind this approach was to enable discovery of semantically demarcated resources using a minimal set of concepts. It is a base ontology, as it does not extend any other base models (such as DOLCE+DnS Ultralite DUL or Dublin Core). However, alignments to other ontologies are known [11].

IoT-Lite [12] is an instantiation of the SSN, i.e. a direct extension of some of its modules. It is a minimal ontology, to which most of the caveats of the SSN apply. Specifically: focus on sensors and observations, reliance on other ontologies (e.g. time or unit ontologies), high modularity and extensibility. The idea behind the *IoT-Lite* was to create a small/light semantic model that would be less taxing (than other, more verbose and broader models) on devices that process it. At the same time, it needed to cover enough concepts to be useful. The ontology describes devices, objects, systems and services. The main extension of the SSN, in the *IoT-Lite*, lies in addition of actuators (to complement sensors, as a device type) and a coverage property. It explicitly uses concepts from a geolocation ontology [13] to demarcate device coverage and deployment location.

OpenIoT ontology [13] was developed within the *OpenIoT* project. However, here, we use the term “*OpenIoT*” to refer to the ontology. It is a comparatively big model that (re)uses and combines other ontologies. Those include all modules of the SSN (the main basis for the *OpenIoT*), SPITFIRE (including sensor networks), Event Model-F, PROV-O, LinkedGeoData, WGS84, CloudDomain, SIOC, Association Ontology and others, including smaller ontologies developed at the DERI (currently, Insight Centre). It also makes use of ontologies that provide basis for those enumerated earlier, e.g. DUL. Other than concepts from the SSN, *OpenIoT*, uses a large number of SPITFIRE concepts, e.g. network and sensor network descriptions. While some mentioned ontologies are not imported by the *OpenIoT* explicitly, they appear in all examples, documentation, and project deliverables. Therefore, one can treat *OpenIoT* as a combination of parts of all of those. Similar to the SSN, *OpenIoT* does not define its own location concepts and does not explicitly import geolocation ontologies. It relies on other ontologies for that but, in contrast to the SSN, it clearly indicates LinkedGeoData and WGS84 as sources of geolocation descriptions. It defines a limited set of units of measure (e.g. temperature, wind speed), but only when they were relevant to the *OpenIoT* project pilot implementation.

The rich suite of used ontologies means that *OpenIoT* provides a very extensive description of devices, their functionalities, capabilities, provenance, measurements, deployments and position, energy, relevant events, users and many others.

Interestingly enough, it does not explicitly describe actuators or actuating properties/functions. It can be observed that the broad scope of the ontology makes it rather complicated. This is also because, it is not documented well-enough, i.e. the detail level and ease-of-access of the documentation do not match the range of coverage of concepts in the model. Moreover, it is not clearly and explicitly modularized, despite being an extension of the SSN. Let us note that, while there are other IoT models of potential interest (such as OGC Sensor Things, UniversAAL ontologies, FAN FPAI, IoT Ontology,¹ M3 Vocabulary), we have decided that they are of less importance or relevance to INTER-IoT. This was either because they have generated much less “general interest”, or had scope well outside that of the project.

Each of considered ontologies proposes a different approach to modelling the IoT space. The biggest differences are in the details.

1. OneM2M BO proposes a small base ontology, similar to upper ontologies that provides only a minimal set of highly abstract entities. This allows for a very broad set of domain ontologies to be easily aligned with it. It also means that the BO itself is not enough to model any concrete problem (or solution) in the IoT. Furthermore, it does not capture some aspects (device, sensor and actuator properties) that are very common in other ontologies.
2. OpenIoT contrasts the oneM2M BO philosophy by providing a detailed model for a specific problem (i.e. pilot implementations from the OpenIoT project) that can be also be applied in a more general case, or in other solutions. Its heavy usage of external ontologies provides high semantic interoperability by design.
3. SSN is a developed model of the IoT in general, but with strong focus on sensor networks. It is based on DUL, and is clearly modularized, which makes it a good candidate for extensions into concrete systems and implementations. This is evidenced by the fact that other ontologies, evaluated here, make good use of it. When it comes to specificity, it places itself in the middle between oneM2M BO and OpenIoT.
4. IoT-Lite is an extension of selected SSN modules, mainly to include actuators. Rather than focusing on providing a detailed description of a delimited problem space within the IoT, it approaches the modelling problem from the perspective of an implementation device. It aims to deliver a small, but complete, model in order to simplify processing of semantic information. This is also its distinctive characteristic.
5. SAREF is a model with a strong focus on its own area—of smart appliances. Even though mappings to other standards exist, SAREF was developed from scratch to represent a specific area of application of the IoT. In this area, it delivers a strong and detailed base, that is also clear and easy to understand. At the same time, it is general enough to be used when extended to other domains, or solutions. Interestingly, all these ontologies almost completely disregard hardware specifications. It seems that the “place” of a device in an IoT system is much more important to ontology engineers than its hardware specification and resulting capabilities.

¹ <http://ai-group.ds.unipi.gr/kotis/ontologies/IoT-ontology>.

Results of our investigations show how different the existing conceptualizations of the same domain can be, depending on the context of the approach, and the applied ontology engineering methodology. Separately, we conclude that, while each considered ontology has its uses and caveats, two of them stand out in the context of the INTER-IoT project. These are SSN and SAREF. The first presents a model focused on sensors, but still robust enough, and with strong ontological basis. Those features make it a good choice in terms of interoperability (which is the focus of the project). In addition, the SSN is modular, extendable, and has been actually implemented and extended in other systems and ontologies (e.g. IoT-Lite and OpenIoT). SAREF, on the other hand, is a thoroughly modern ontology with many recommendations and relatively large scope, despite targeting only smart appliances. It already has alignments with other models, thus improving its interoperability. In light of the facts and analysis presented in this section, the SSN ontology will be used as the basis of INTER-IoT reference meta-data model. We have found that it covers many meta-data requirements identified in INTER-IoT and is designed to be modular and extendible. It is also a core ontology, with many implementations in already deployed and well-tested systems. This makes it, in our view, the best currently available core IoT ontology for INTER-IoT.

2.3 *Functional Model*

The focus of the INTER-IoT Functional Model 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).

The Functional Model is divided into 9 FGs:

- At the bottom, we put the FG **Device**. This FG regroups all the Devices according to the definition in the Domain Model.
- As INTER-IoT also deals with Platforms that opaquely cover some of the devices, the **IoT Platform** FG is defined. 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 and the VE Interoperability Service.
- In the **Ontology** FG there are the Platform Ontology and the Global Interoperability Ontology.

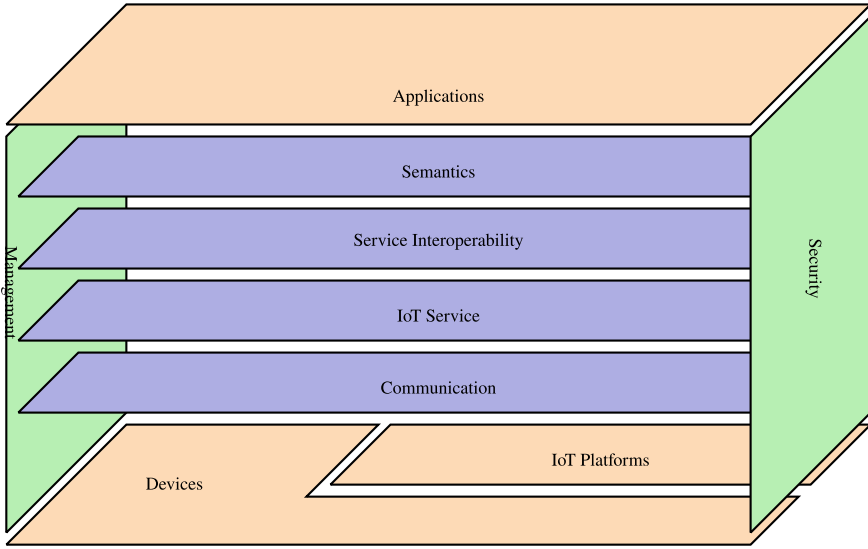


Fig. 2 Overall INTER-IoT functional model

- 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 an orthogonal **Security** FG is identified.
- Finally, the orthogonal **Management** FG is required for the management of and/or interaction between the functionality groups (Fig. 2).

2.3.1 Communication

The Communication FG is fundamental 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. This FG is represented by the Communication Model (see Sect. 2.4). Its focus is to provide a bridge 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.

2.3.2 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 [14].

The IoT Service Functional Group contains IoT Services as well as functionalities for discovery, look-up, and name resolution of IoT Services.

2.3.3 Service Interoperability

The Service Interoperability FG is central for any kind of interaction between different entities [15], 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. 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 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.

2.3.4 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. The Semantics FG 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. The Semantics FG also performs the ontology alignment, or the translation from an origin to a target ontology. 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.

2.3.5 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 wish 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.

2.3.6 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 be 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 our end users will face, 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.

2.4 *Communication Model*

2.4.1 **Communication Protocols on IoT Platforms**

The INTER-IoT Communication Model aims at defining the paradigms for connecting the elements that compose any IoT system. As the IoT domain is composed by different kind of nodes, and includes resources that are constrained, the communication needs to focus on the whole communication spectrum from the device to the application level.

The adoption of gateways and brokers is necessary given the variety of environments presents in different IoT systems. 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 all communication endpoints are perfectly covered as mediators that understand several protocols are implemented at each of the layers; in particular:

- *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.

2.4.2 INTER-IoT Domain Model Element Communications

Device to Device Interactions

In Device to Device interactions, the Device is a Sensor and Actuator, or both, and with a unique direction for it to be addressable. This is the main component for the interaction and with this is representing the Physical Device in our Domain Model. Additionally, other resources of the D2D gateway, the N2N network or the Platform will appear in the communication. In INTER-IoT view, the Device (Sensor, Actuator or Tag) represents the objects we desire to interconnect. Physical and Virtual Entities are concepts implemented by the gateway as well as the Augmented Entity and the Interoperability resources or Services are provided by the components within the D2D gateway solution.

Network to Network Interactions

In this interaction, the elements or entities involved in the communication at network level can be easily explained by means of the following use case:

- **Device communicates with resource in the network:** In this case, the interaction takes place within the network, when a device or entity interacts with a resource or service hosted and running in the in the network, or even when one or both element that want to interact belong to this network (see Fig. 3).
- **Platform service or a resource communicates with another resource in the network:** The domain model element communications affecting the network layer are the IoT Platforms and Platform services (red) requesting information about the IoT Services (green) available in the resources specifically network resources (yellow) (see Fig. 4).
- **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 (Fig. 5).

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.

Middleware to Middleware Interactions

The components that interact with underlying IoT platforms are the entities represented by *IoT Platform* and *Platform Ontology*, which is extended with the *Generic Ontology of IoT Platforms* module. 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 entities: the *Platform Interoperability Service* and the

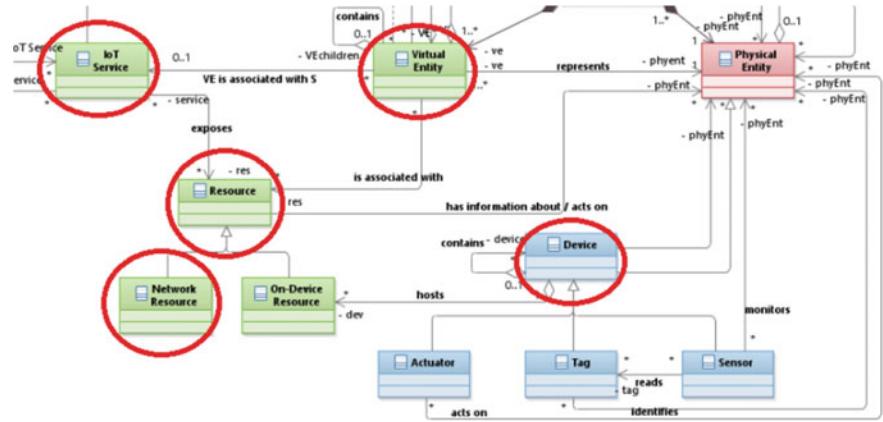
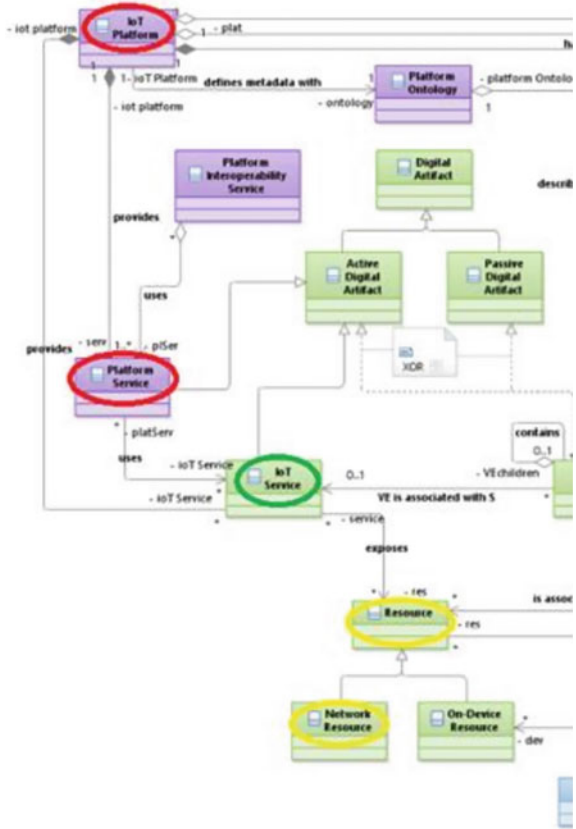


Fig. 3 Domain model entities involved in network-to-network communication when the device communicates with resource in the network

Fig. 4 Domain model entities involved in device-to-device communication when platform communicates with another resource in network



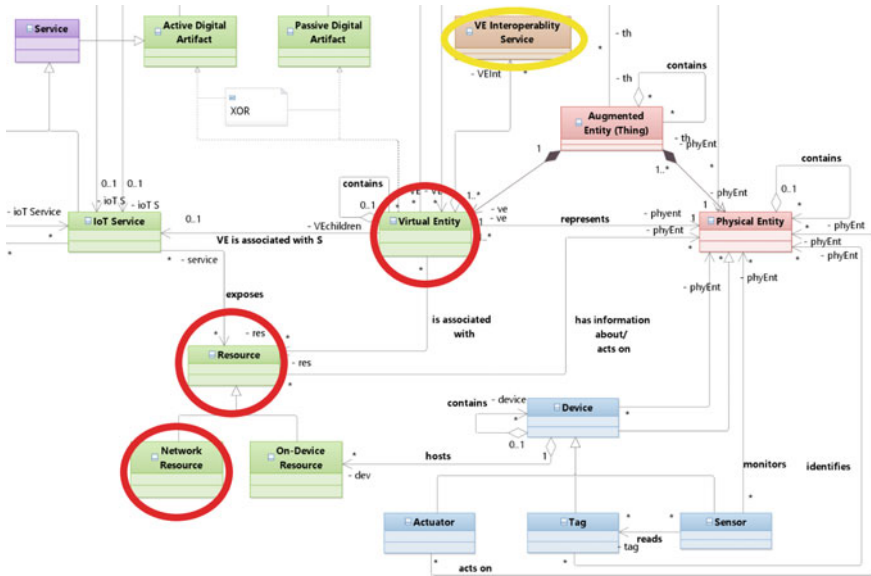


Fig. 5 Domain model entities involved in network-to-network communication

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 whose main purpose is to interoperate different Services from Platforms. The implementation of the Domain Model in the MW2MW and DS2DS architecture, 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 and 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 1 User accessing an IoT Platform through INTER-MW.

The interaction between user and IoT Platform has changed. The *Platform Ontology* used in previous interaction 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 Fig. 6) 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*).

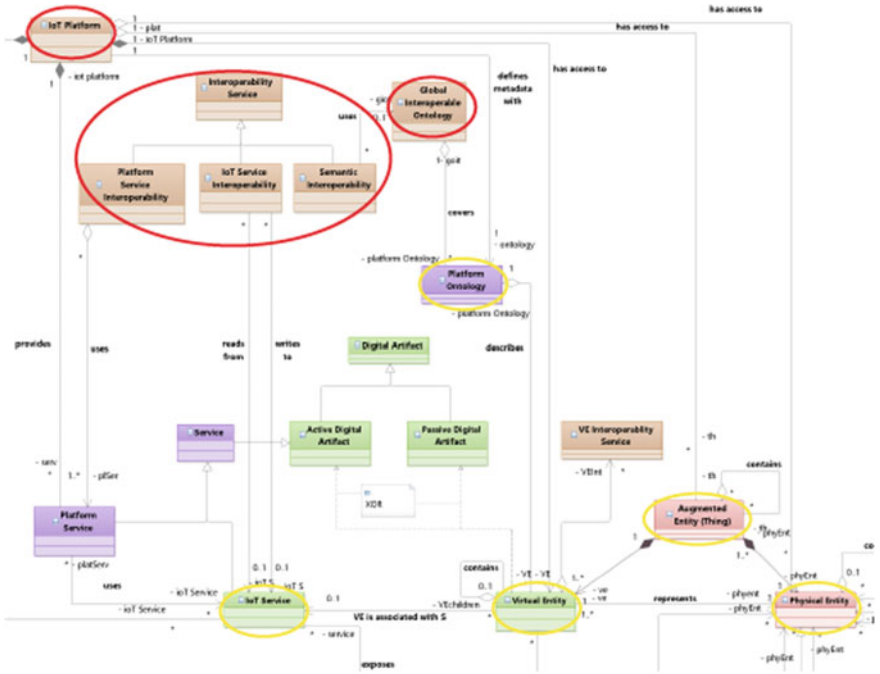


Fig. 6 Domain model entities involved in middleware-to-middleware communication when human user accesses an IoT platform through INTER-MW

Example 2 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 Fig. 7), are also needed.

Example 3 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 Fig. 8).

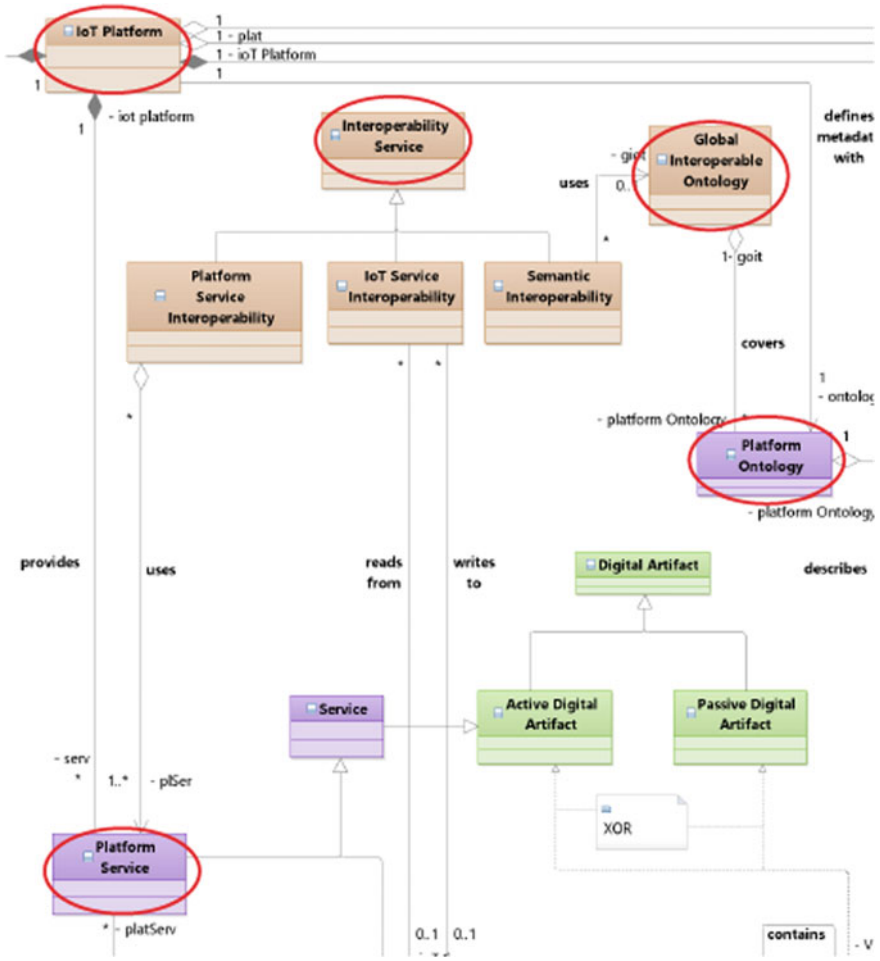


Fig. 7 Domain model entities involved in middleware-to-middleware communication when user tries to configure an IoT platform

Application and Services to Application and Services Interactions

The components of the Domain Model involved in the communication at Application and Services level include 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 [14].

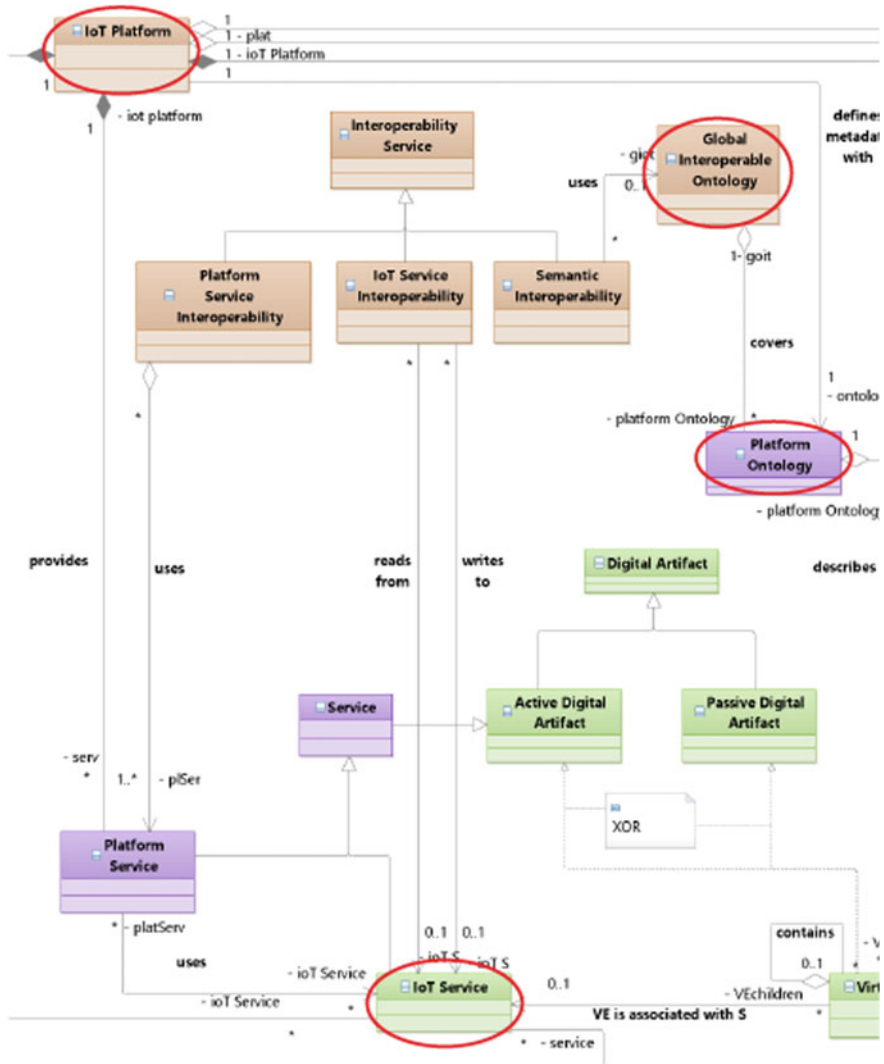


Fig. 8 Domain model entities involved in middleware-to-middleware communication between IoT platforms

Example 1 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 (Fig. 9).

Example 2 Service of an IoT Platform communicates with a service from another IoT Platform.

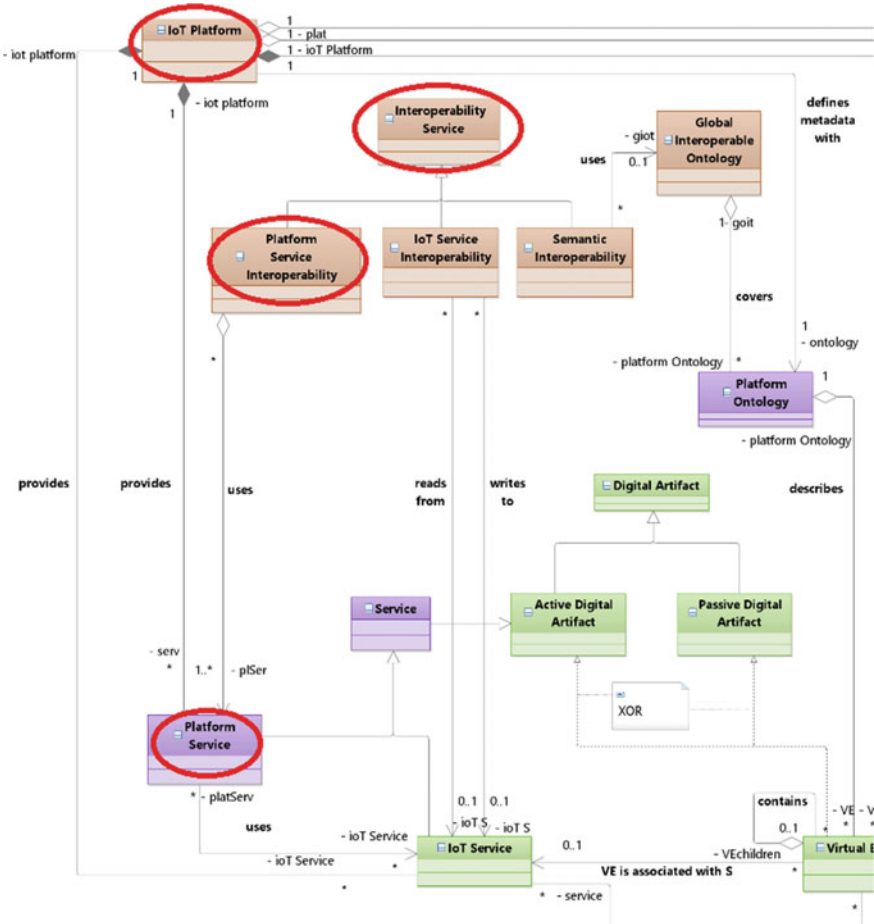


Fig. 9 Domain model entities involved in application and services-to-application and services communication creating a composed service

In the second use case it is indicated that, in some cases, it is necessary to use 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 (Fig. 10).

Data and Semantics to Data and Semantics Interactions

The domain model interactions in DS2DS do not include any new elements.

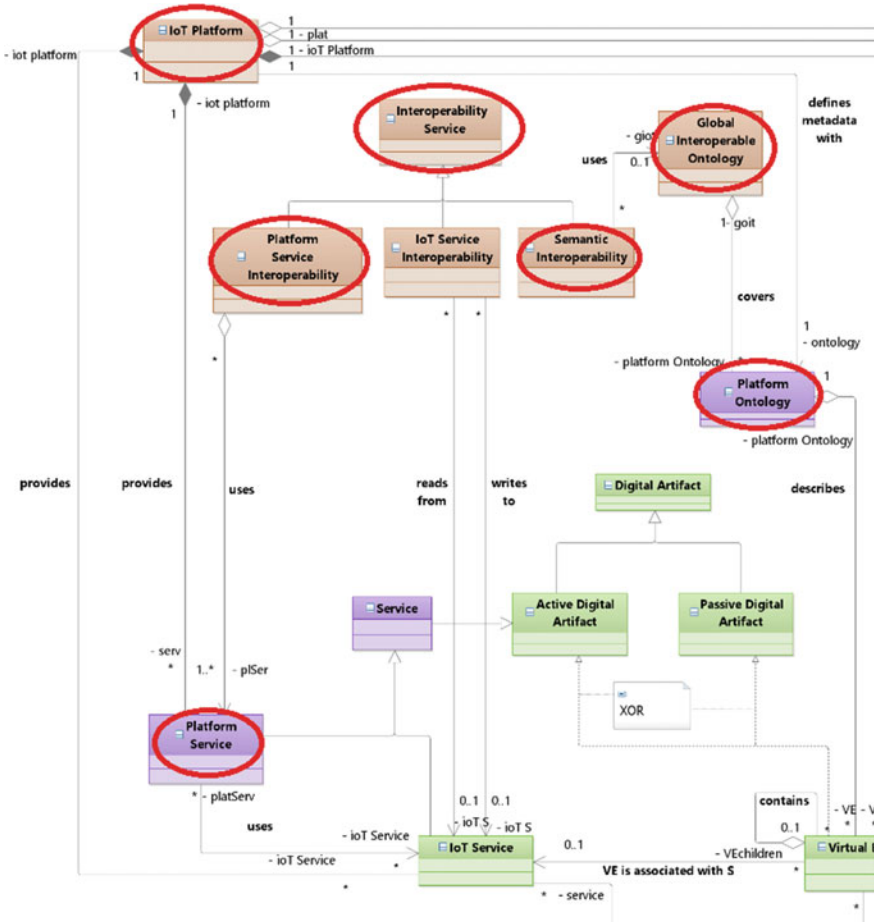


Fig. 10 Domain model entities involved in application and services-to-application and services communication between platform’s services

2.4.3 INTER-IoT Channel Model for Interoperability

Device to Device Interactions

The updated Communication Model defining device to device interactions considers two cases. The first one is shown in Fig. 12 considering device to device communication through the same physical/virtual gateway. In this case, interoperability is achieved since a sensor and actuator with different network and protocol stacks are able to interact (Fig. 11).

The second case is shown in Figs. 13 and 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.

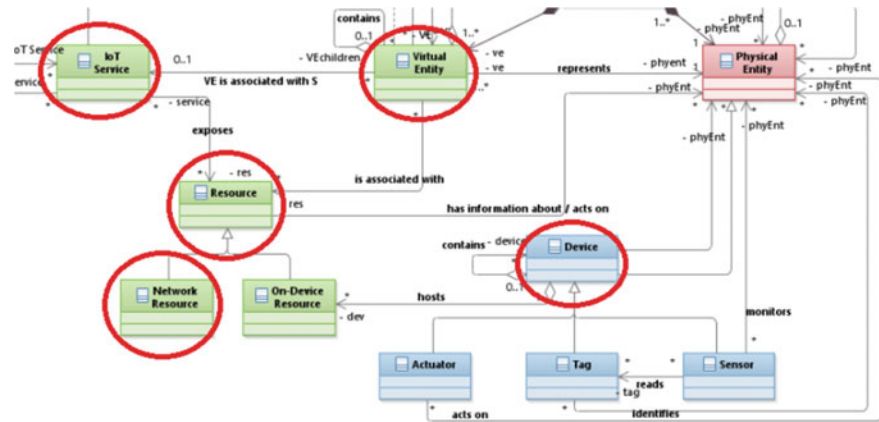


Fig. 11 Domain model entities involved in network-to-network communication when the device communicates with resource in the network

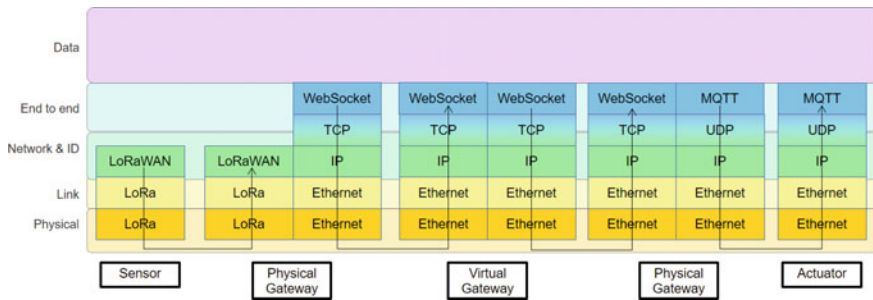


Fig. 12 Device to device channel interactions

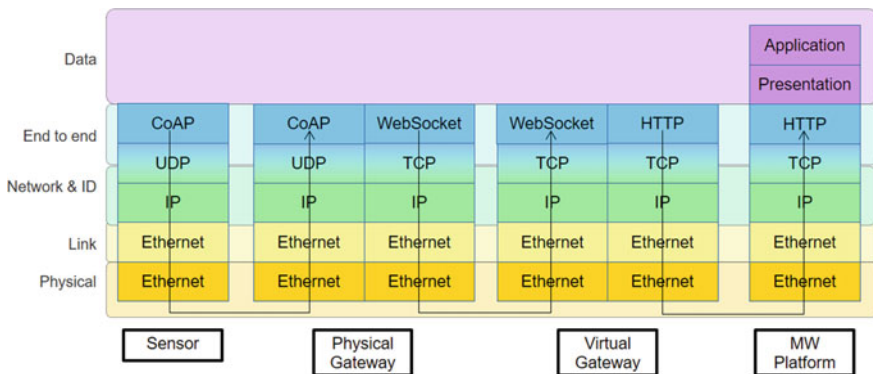


Fig. 13 Device to middleware channel interactions

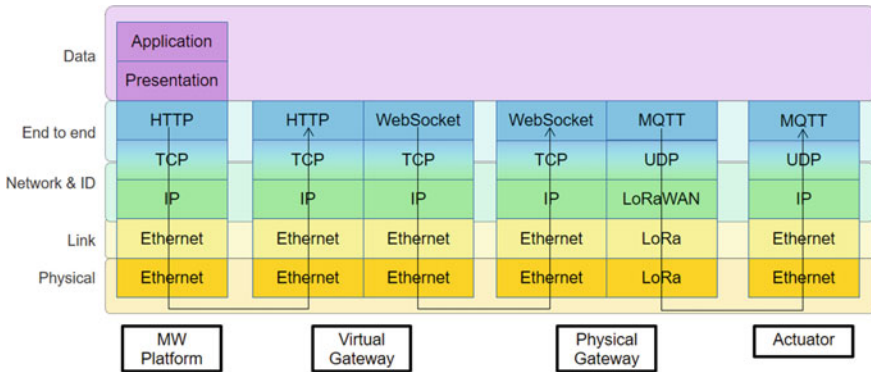


Fig. 14 Middleware to device channel interactions

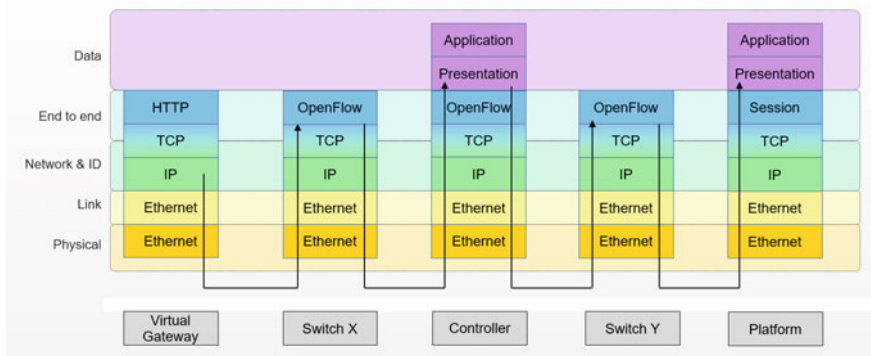


Fig. 15 Network to network channel interactions

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 Fig. 16.

In Fig. 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.

Additionally, an example of communication of the control and management information appears in Fig. 16 where, using protocols as OpenFlow and OVSDB, the new

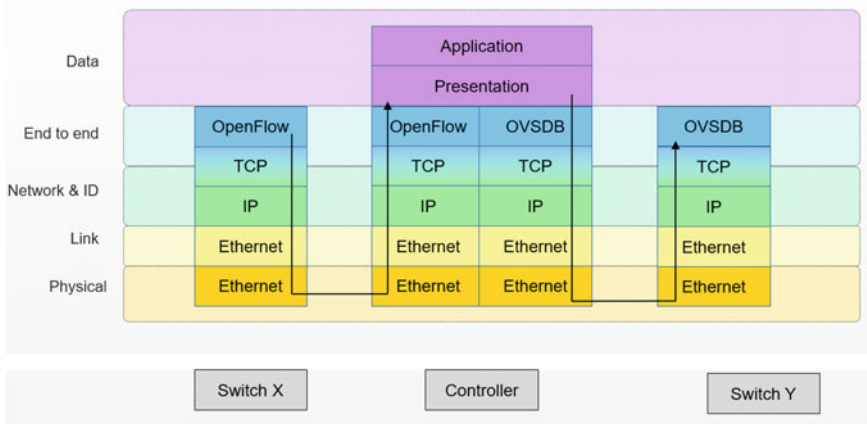


Fig. 16 Network element to controller channel interactions

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.

Middleware to Middleware Interactions

An example of middleware to middleware communication is indicated in the Fig. 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.

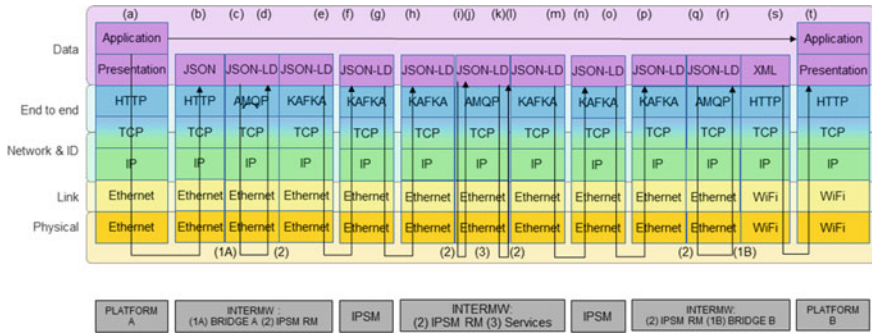


Fig. 17 Communication diagram in middleware-to-middleware interoperability

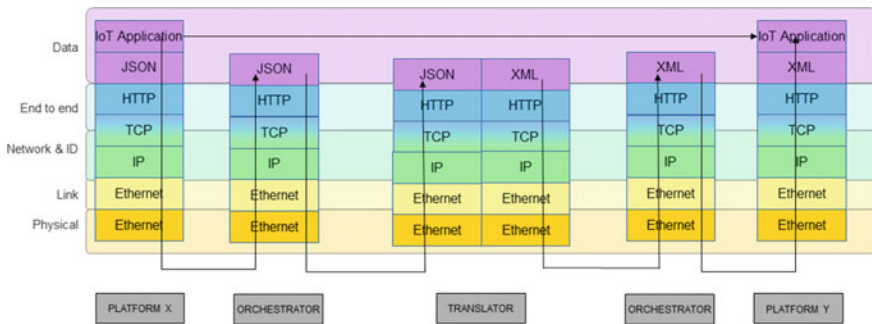


Fig. 18 AS to AS element channel interactions

Application and Services to Application and Services 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 Fig. 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.

Data and Semantics to Data and Semantics Interactions

Interactions between Domain Model elements on DS2DS layer are exclusively between an IoT artifact (platform) and its ontology. They are two-fold and divided into preparation of ontology and its usage. In the INTER-IoT approach, every platform (system, application, etc.), which would voluntarily like to interoperate with one or more other platforms needs to be prepared and willing, first. In order to enable semantic interoperability, and explicit ontology is needed. Some platforms, or middlewares (e.g. UniversAAL, OpenIoT) already need to have OWL ontologies ready before deployment. These ontologies can be used in INTER-IoT. In other cases, any semantics present in a platform needs to be extracted and formalized into an OWL ontology. In short, lifting to OWL is a process in which semantics of platforms, sometimes contained in data schemas, are made explicit and stored in an ontology. The most popular languages that can be used in lifting are: XML, RDF, JSON LD, but other formalisms are also acceptable. Such formal description must cover all aspects of data communication that will be needed for interoperability. It must represent entities (and their properties), which exist “inside” of the artefact. This formal description is to be used in creation of platform ontology, as well as in instantiation of communication channel(s) needed to send/receive messages to/from other artifacts (platforms, devices, middleware, services, or applications). Once a platform has an ontology it is then used to create alignments to and from the GOIoTP, that later serve as configuration for IPSM. The semantic translation process that takes place inside IPSM is non-discriminative when it comes to contents or intentions of communication. It simply translates the meaning of messages, according to configuration of the communication channel that received the messages. Dynamic creation of communication channels allows DS2DS interactions to serve multiple purposes and assist in operation of other INTER-IoT components, as well as other artifacts, if they wish to use IPSM as a “stand-alone” service within INTER-IoT. It should be stressed that data processing within a single artifact can be represented through more than one ontology (or, possibly, modules within a single modular ontology). Such situation can materialize when different ontologies are used in different “conversations” (concerning different aspects of data, usually with different artefacts). In this way, proposed approach gains flexibility and addresses the issue of scalability (semantic processing is applied to smaller (sub-)ontologies).

3 INTER-IoT Reference Architecture

While an Architectural Reference Model (ARM) [16] is a description of elements and relation types together with a set of constraints on how they may be used, a Reference Architecture (RA) is a blueprint for developing Concrete Architectures. Specifically, as shown in Fig. 19, a reference architecture is a Reference Model (RM) mapped

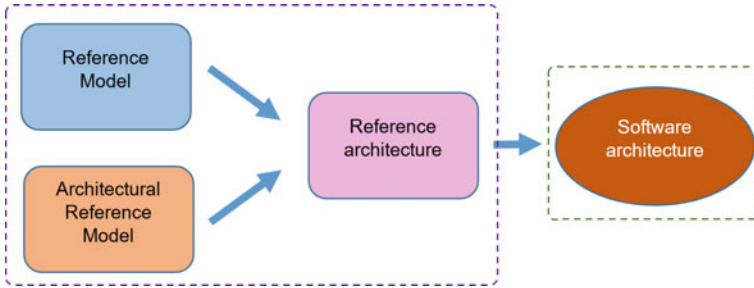


Fig. 19 Reference architecture derivation

onto software elements (that cooperatively implement the functionality defined in the reference model) and the data flows between them. The use of a RM and an ARM to create a RA to instantiate a software architecture in the domain of IoT is widely described in IOT-A [3] and appears previously in [17].

It is worth noting that a RA is not a concrete architecture; its role is not to completely specify all the technologies, components and their relationships in sufficient detail to enable direct implementation but rather to provide a reference to derive concrete system architectures.

The main scope of a RA is to apply different views to functionality areas, in order to identify the relationships between different modules and the need of different functions. In the following subsections, we describe in detail the most important of such views related to Inter-IoT RA which are the Functional View and the Information View.

3.1 Functional View

Figure 20 shows the approach we followed for defining the final INTER-IoT Functional Architecture (FA) . There are 9 different functional groups: Devices, IoT Platforms, Management, Security, Applications, Communication, IoT Services, Service Interoperability and Semantics [18].

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, 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 (Fig. 21).

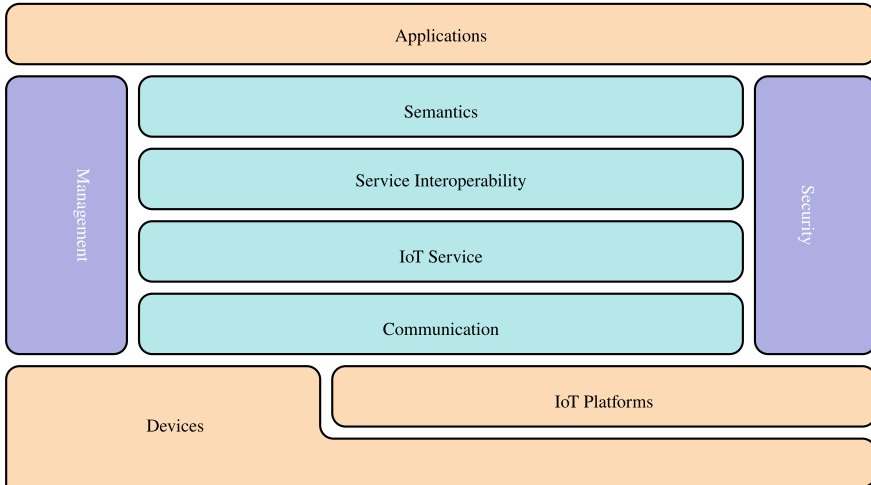


Fig. 20 Overall functional architecture

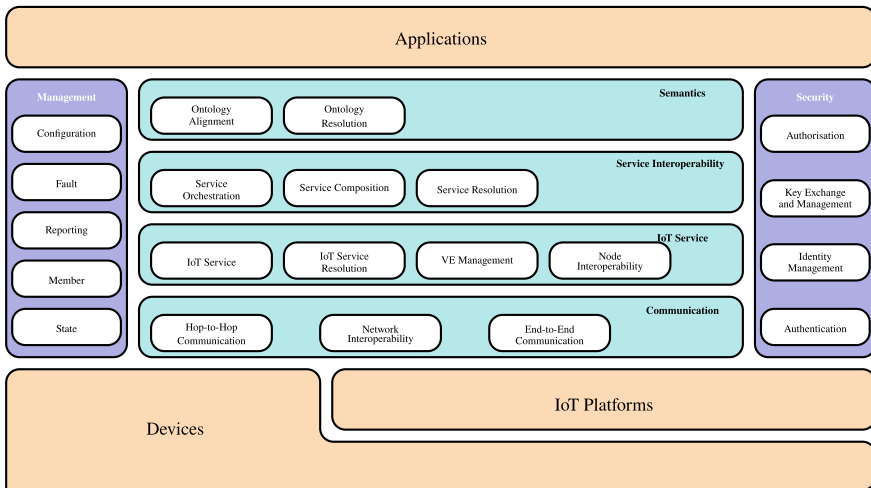


Fig. 21 Mapping of the modules in the FA groups

3.1.1 Semantics

The Semantics FG is the central Functional Group that addresses the challenges related to semantic interoperability of IoT Platforms. 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. 22)



Fig. 22 Semantics FG

- Ontology Resolution;
- Ontology Alignment.

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.

3.1.2 Service Interoperability FG

The role of the Service Interoperability FG is to support the Application and Service to Application and 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 Fig. 23):



Fig. 23 Service interoperability FG

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

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,
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 cataloging 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 cataloged 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 cataloger 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.

3.1.3 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) (Fig. 24).

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



Fig. 24 IoT service FG

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 °C 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

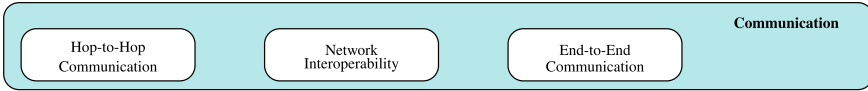


Fig. 25 Communication FG

or their actions may be linked following 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 (Fig. 25).

3.1.4 Communication

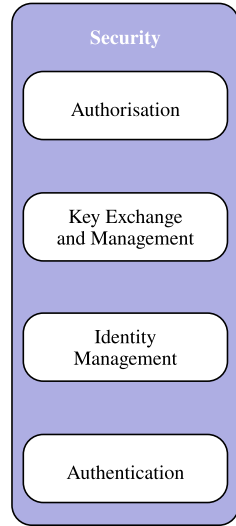
As in the original IoT-A Functional view, we thought that for simplicity matters having a FG dedicated to handle communication within different entities would greatly help the understanding and the usage of the Functional View in general.

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,

Fig. 26 Security FG



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).

3.1.5 Security

The Security FG is responsible for ensuring the security and privacy during the interaction of all systems (Fig. 26).

It consists of four functional components:

- Authorisation;
- Key Exchange and 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 basic functionalities offered by the Authorization FC (i) 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 (ii) manage policies. This refers to adding, updating or deleting an access policy.

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 functionalities provided by the Authentication FC authenticate a user based on provided credential, and (ii) 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.

3.1.6 Management

Similarly to IoT-A, we followed FCAPS for the Management FG 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.

Fig. 27 Management FG

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 (Fig. 27).

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:

- The IoT-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 runtime 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.

3.2 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.

IoT-A defined 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 Entity Type 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 Entity Type is of special importance, as it can be used to determine what attributes a Virtual Entity instance can have, defining its semantics. The Entity Type can be modelled in two different ways: either based on a flat type system or as a type hierarchy, enabling sub-type matching. Entity Types are similar to classes in object-oriented programming, so UML class diagrams are suitable for modelling Entity Types. Similarly, the generalization relation can be used for modelling sub-classes of Entity Types, creating a hierarchy of several Entity Types 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.

3.2.1 Information Handling by Functional Components

There are four message exchanges patterns considered for information exchange between IoT Functional Components: Push, Request/Response, Subscribe/Notify, Publish/Subscribe. The Push-pattern is a one-way communication between two par-

ties in which a server sends data to a predefined client that receives the data. The server hereby knows the address of the client beforehand and the client is constantly awaiting messages from the server. The communication channel in this pattern is pre-defined and meant to be applied in scenarios in which the communication partners do not change often. For example, the server can be a constrained device that sends data to a gateway dedicated to this device. The gateway is listening constantly to the device and is consuming the data received from this device.

The Request/Response pattern is a synchronous way of communication between two parties. A client sends a request to a server. The server will receive the request and will send a response back to the client. The client is waiting for the response until the server has sent it. The Subscribe/Notify pattern allows an asynchronous way of communication between two parties without the client waiting for the server response. The client just indicates the interest in a service on the server by sending a subscribe-call to the server. The server stores the subscription together with the address of the client wants to get notified on and sends notifications to this address whenever they are ready to be sent.

The Publish/Subscribe pattern allows a loose coupling between communication partners. There are services offering information and advertise those offers on a broker component. When clients declare their interest in certain information on the broker the component will make sure the information flow between service and client will be established.

3.3 Deployment and Operation View

The Deployment and Operation View aims at developing a set of guidelines to drive users through the different design choices that they must face while designing the actual implementation of their services. To this extent this view will discuss how to move from the service description and the identification of the different functional elements to the selection among the many available technologies in the IoT to build up the overall networking behaviour for the deployment. Since a complete analysis of all the technological possibilities and their combination may be extremely complex, IoT-A focus is on those categories that have the strongest impact on IoT systems realization. Starting from the IoT Domain Model, there are three main element groups: Devices, Resources, and Services. Each of them poses a different deployment problem, which, in turn, reflects on the operational capabilities of the system.

In particular, the viewpoints used in the Deployment and Operation view are the following:

- The IoT Domain Model diagram is used as a guideline to describe the specific application domain;
- The Functional Model is used as a reference to the system definition, as it defines Functional Groups;

- Network connectivity diagrams can be used to plan the connectivity topology to enable the desired networking capability of the target application; at the deployment level, the connectivity diagram will be used to define the hierarchies and the type of the sub-networks composing the complete system network;
- Device Descriptions (such as datasheets and user manuals) can be used to map actual hardware on the service and resource requirements of the target system.

Devices in IoT systems include the whole spectrum of technologies ranging from the simplest of the radio-frequency tags to the smartest objects able to understand the environment and take real-time decisions. The unifying characteristics are the connection and the capability of performing computation. These two characteristics are the subject of the first choices a system designer has to make.

Selecting the computational complexity for a given device is intrinsic to the target application and to the planned road-map: for instance, a system architect may choose to have a large amount of memory that may seem unnecessary at first, but may be used for future releases and upgrades. On the other hand, choosing among the different connectivity types is not as straightforward as different choices may provide comparable advantages, but in different areas. For the same reason, it is possible to realize different systems implementing the same or similar application from the functional view, which are extremely different from the Deployment and Operation view.

Because of the coexistence of different communication technologies in the same system, the second choice the system designer must account for is related to communication protocols. Connectivity functionalities for IoT system are defined within the ARM in the Communication FG of the FM; in addition, to better understand the application, it is important to describe it within the Functional View.

The following possibilities have been identified:

1. *IoT protocol suite*: This is supposed to be the best solution for interoperability;
2. *Ad-hoc proprietary solutions*: Whenever the performance requirements of the target application are more important than the system versatility, ad hoc solutions may be the only way to go;
3. *Other standards*: Depending on the target application domain, regulations may exist forcing the system designer to adopt standards, different from those suggested by the IoT protocol suite, that solved a given past issue and have been maintained for continuity.

After having selected the devices and their communication methods, the system designer has to account for services and resources, as defined in the IoT Service FG section. These are pieces of software that range from simple binary application and increasing their complexity up to full-blown control software. Both in the case of resources and for services the key point here is to choose where to deploy the software related to a given device. The options are as follows:

1. *On smart objects*: This choice applies to simple resource definitions and lightweight services, such as web-services that may be realized in few tens or hundreds of bytes;

2. *On gateways*: Whenever the target devices are not powerful enough to run the needed software themselves, gateways or other more capable devices have to be deployed to assist the less capable ones;
3. *In the cloud*: Software can be also deployed on web-farms. This solution improves the availability of the services, but may decrease the performance in terms of latency and throughput.

Note that this choice must be made per type of resource and service and depending on the related device. As an example, a temperature sensor can be deployed on a wireless constrained device, which can host the temperature resource with a simple service for providing it, but, if a more complex service (for instance, when the Service Organisation FG is called in) is needed, the software should be deployed on a more powerful device as per option (2) or (3).

On the same line, it is important to select where to store the information collected by the system, let their data be gathered by sensor networks or through additional information provided by users. In such a choice, a designer must take into consideration the sensitiveness (e.g.: is the device capable of running the security framework), the needed data availability and the degree of redundancy needed for data resiliency. This choice is also very important for what concerns interoperability, as the location of the data may ease the interaction between different systems—or, at the contrary, may prove very complex to overcome. The foreseen options are the following:

1. *Local only*: Data is stored on the device that produced it, only. In such a case, the locality of data is enforced and the system does not require complex distributed databases, but, depending on the location of a given request, the response might take longer time to be delivered and, in the worst-case scenario, it may get lost;
2. *Web only*: No local copy is maintained by devices. As soon as data is sent to the aggregator, they are dispatched in databases;
3. *Local with web cache*: A hierarchical structure for storing data is maintained from devices up to database servers.

Finally, one of the core features of IoT systems is the resolution of services and entities, which is provided by the Entity and Service Resolution FCs, respectively and oversees semantically retrieving resources and services, discovering new elements and binding users with data, resources, and services. This is performed adopting the definitions of the Virtual Entity FG. This choice, while one of the most important for the designer, has only two options:

1. *Internal deployment*: The core engine is installed on servers belonging to the system and is dedicated to the target application or shared between different applications of the same provider;
2. *External usage*: The core engine is provided by a third party and the system designer has to drive the service development on the third-party APIs.

Differently from the other choices, this is driven by the cost associated to the maintenance of the core engine software. In fact, since it is a critical component of the system, security, availability and robustness must be enforced. Hence, for small enterprises the most feasible solution is the external one.

4 Conclusions

The INTER-IoT Reference Model (RM) has been defined 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. In this chapter we discussed the INTER-IoT reference model. In particular, we presented the domain model, the Information Model, the Functional Model (in details, given its centrality role), and the Communication Model. We also described the INTER-IoT Reference Architecture, which is also derived from the IoT-A Reference Architecture and is a refinement with the main target of modeling interoperability by design. The 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.

References

1. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: Towards multi-layer interoperability of heterogeneous IoT platforms: the INTER-IoT approach (2018) *Internet of Things*, 0, pp. 199–232
2. Vermesan, O., Friess, P. (eds.): *Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds*. River Publishers (2016)
3. Bauer, M., Boussard, M., Bui, N., Carrez, F., Jardak, C., De Loof, J., Magerkurth, C., Meissner, S., Nettsträter, A., Olivereau, A., Thoma, M., Walewski, J.W., Stefa, J., Salinas, A.: *Internet of things—architecture IoT-a deliverable D1.5—final architectural reference model for the IoT v3.0*, July 2013
4. Bassi, A., Bauer, M., Fiedler, M., Kramp, T., Kranenburg, R.V., Lange, S., Meissner, S.: *Enabling things to talk: designing IoT solutions with the IoT architectural reference model* (2013)
5. Broring, A., Zappa, A., Vermesan, O., Främling, K., Zaslavsky, A., Gonzalez-Usach, R., Szmaja, P., Palau, C., Jacoby, M., Zarko, I.P., Sour-sos, S., Schmitt, C., Plociennik, M., Krco, S., Georgoulas, S., Larizgoitia, I., Gligoric, N., García-Castro, R., Serena, F., Orav, V.: *Advancing IoT Platform Interoperability*. River Publishers (2018)
6. Ganzha, M., Paprzycki, M., Pawłowski, W., Szmaja, P., Wasielewska, K.: Semantic interoperability in the Internet of Things: an overview from the Inter-IoT perspective. *J. Netw. Comput. Appl.* **81**, 111–124 (2017). <http://www.sciencedirect.com/science/article/pii/S1084804516301618>
7. Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W.D., LePhuoc, D., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The SSN ontology of the W3C semantic sensor network incubator group. *J. Web Semant.* **17**, 25–32 (2012). <http://www.sciencedirect.com/science/article/pii/S1570826812000571>

8. Pileggi, S.F., Palau, C.E., Esteve, M.: Building semantic sensor web: knowledge and interoperability. In: *Proceedings of the International Workshop on Semantic Sensor Web, SSW, (IC3K 2010)*, vol. 1, pp. 15.22, April 2010
9. Daniele, L., den Hartog, F., Roes, J.: Created in close interaction with the industry: the smart appliances reference (SAREF) ontology, 082015, pp. 100–112
10. Alaya, M., Medjiah, S., Monteil, T., Drira, K.: Towards semantic data interoperability in oneM2M standard, January 2015
11. Ganzha, M., Paprzycki, M., Pawłowski, W., Szmeja, P., Wasielewska, K., Fortino, G.: Tools for ontology matching—practical considerations from Inter-IoT perspective, vol. 9864, pp. 296–307, September 2016
12. Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., Taylor, K.: IoT-lite: a lightweight semantic model for the Internet of Things. In: *2016 International IEEE Conferences on Ubiquitous Intelligence Computing, Advanced and Trusted Computing, Scalable Computing and Communications, Cloud and Big Data Computing, Internet of People, and Smart World Congress (UIC/ATC/ScalCom/CBDCCom/IoP/SmartWorld)*, pp. 90–97 (2016)
13. Soldatos, J., Kefalakis, N., Hauswirth, M., Serrano, M., Calbimonte, J.-P., Riahi, M., Aberer, K., Jayaraman, P.P., Zaslavsky, A., Zarko, I.P., Skorin-Kapov, L., Herzog, R.: OpenIoT: open source Internet of Things in the cloud. In: *PodnarZarko, I., Pripuzic, K., Serrano, M. (eds.) Interoperability and Open-Source Solutions for the Internet of Things*, pp. 13–25. Springer International Publishing, Cham (2015)
14. Belsa, A., Sarabia-Jacome, D., Palau, C.E., Esteve, M.: Flow-based programming interoperability solution for IoT platform applications. In: *2018 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 304–309, Orlando (FL), February 2018
15. Fortino, G., Garro, A., Russo, W.: Achieving mobile agent systems interoperability through software layering. *Inf. Softw. Technol.* **50**(4), 322–341 (2008)
16. Bauer, M. Bui, N., Jardak, C., Nettstrater, A.: *The IoT ARM Reference Manual*, pp. 213–236. Springer, Berlin, Heidelberg (2013). <https://doi.org/10.1007/978-3-642-40403-09>
17. Bass, L., Clements, P., Kazman, R.: *Software Architecture in Practice*, January 2003
18. Fortino, G., Liotta, A., Palau, C., Gravina, R., Manso, M. (eds.): *Integration, Interconnection, and Interoperability of IoT Systems*. Springer, February 2017

INTER-Layer: A Layered Approach for IoT Platform Interoperability



Andreu Belsa, Alejandro Fornes-Leal, Clara I. Valero, Eneko Olivares, Jara Suárez de Puga, Fernando Boronat, and Flavio Fuart

Abstract A interoperability layer is fundamental to provide a global continuum interoperability among IoT platforms. To address this layer, the following activities have been carried out: (i) design of device-to-device interaction based on multiprotocol/access mechanisms; (ii) design of software defined interoperable modules for mobility and routing; (iii) development of an open management framework for smart objects; (iv) design and implementation of smart IoT application service gateway and virtualization; and (v) definition of a common ontology which will facilitate access to the heterogeneous data, data that will be collected and managed by integrated IoT platforms.

1 Introduction

The lack of interoperability in the IoT ecosystem causes many issues, from the impossibility of connecting non-interoperable devices into different IoT platforms, to difficulties in leveraging data of multiple platforms to conform applications and, to slowing the introduction of novel IoT technologies at large scale [1–3]. The INTER-IoT presents a layer-oriented solution to provide interoperability at any layer and across layers among different IoT systems and platforms. Although its design and development are more challenging in comparison to an application-level approach [4], the layered-oriented approach has a higher potential in order to provide interoperability. It facilitates a tight bidirectional integration, which in turn provides higher performance, complete modularity, high adaptability and flexibility, and presents increased reliability.

This layer-oriented solution is achieved through INTER-Layer. INTER-Layer is an instantiation of the INTER-IoT Reference Architecture (RA) presented in Chap. 3, which was designed specifically for the interoperability of IoT Plat-

A. Belsa (✉) · A. Fornes-Leal · C. I. Valero · E. Olivares · J. Suárez de Puga · F. Boronat
UPV, Universitat Politècnica de València, Camino de Vera, 46022 Valencia, Spain
e-mail: anbelpel@upv.es

F. Fuart
XLAB doo, Pot za Brdom 100, SI-1000 Ljubljana, Slovenia

© Springer Nature Switzerland AG 2021
C. E. Palau (eds.), *Interoperability of Heterogeneous IoT Platforms*, Internet of Things,
https://doi.org/10.1007/978-3-030-82446-4_4

forms. It includes several interoperability solutions (methods and tools) dedicated to specific layers: Device-to-Device (D2D), Networking-to-Networking (N2N), Middleware-to-Middleware (MW2MW), Application and Services-to-Application and Services (AS2AS), and Data and Semantics-to-Data and Semantics (DS2DS).

Each interoperability layer has a strong coupling with adjacent layers and provides an interface which can be used for interacting with the components. Interfaces are controlled by a meta-level framework to provide global interoperability. The different layers can communicate and interoperate with each other through these interfaces, therefore having cross-layering. Cross-layer components enable a deeper and more complete integration, while supporting security and privacy mechanisms for all the layers. In summary, INTER-Layer offers the following benefits at different layers or levels:

- Device level: seamless inclusion of new IoT devices and their interoperation with already existing heterogeneous ones, allowing a fast growth of smart objects ecosystems.
- Networking level: seamless support for smart objects mobility (roaming) and information routing. This will allow the design and implementation of fully connected ecosystems.
- Middleware level: a seamless resource discovery and management system for smart objects and their basic services, to allow the global exploitation of smart objects in large scale IoT systems.
- Application and Services level: the discovery, use, import, export and combination of heterogeneous services between different IoT platforms.
- Data and Semantics level: a common interpretation of data and information from different platforms and heterogeneous data sources, providing semantic interoperability.

Except for the semantic interoperability layer, which has a dedicated chapter, the solutions developed for each layer are described and explained in the following subsections, considering all the relevant components, use cases the technologies applied [5, 6].

2 Device Interoperability

The Device Layer, in the context of an IoT ecosystem, comprises the lowest level layer in the IoT stack [7]. This layer comprises a range of interconnected small devices with limited CPU, memory, and power resources, the so-called “constrained devices”. It includes sensors/actuators, smart objects, and smart devices, and are used to conform a network which in turn may exhibit constraints as well (e.g., unreliable or lossy channels, limited and unpredictable bandwidth, and a highly dynamic topology) [8]. These constrained devices are in charge of gathering information from their respective ecosystems and send this information to one or more server stations. Additionally, they could act on the information, performing some physical action

(including displaying it). Other entities on an IoT deployment, like a base station or a controlling server, might have additional computational and communication resources to support the interaction between constrained devices and applications in a more traditional network approximation.

Interoperability at the device level implies that heterogeneous IoT devices are able to interact with each other, so that IoT devices can be both accessed and controlled through a unified interface and integrated into any IoT platform. At this level, interoperability is usually achieved through gateways deployed in dedicated nodes, although it can be implemented in other elements, such as smartphones [9]. In this subsection, the approach followed in INTER-IoT for achieving this type of interoperability as well as the architecture and components considered are presented. Besides, some used cases and results are depicted.

2.1 INTER-Layer Approach for Device Interoperability

INTER-IoT, and more specifically INTER-Layer, aims to address the following device interoperability challenges:

- Applications and platforms are tightly coupled, preventing them from interacting with other applications/platforms.
- Sensors and actuators communicate only within one system.
- Certain platforms do not implement some important services (i.e. discovery), or do so in an incompatible way.
- Roaming elements can be lost or inaccessible.
- IoT Device software is never platform-independent, since companies produce proprietary/closed solutions for economical reasons. This makes interoperability hard or impossible.

Historically, there have been several approaches and communication patterns to operate at device level in IoT systems, each one of them having different application areas and characteristics [10]: *Strict Device-to-Device Communication Pattern*, *Device-to-Cloud Communication Pattern*, and *Device-to-Gateway Communication Pattern*. The novelty introduced by INTER-IoT in this layer is a new communication pattern: ***Device-to-Edge Communication Pattern***. In this communication pattern, devices interact with a gateway, similar to the *Device-to-Gateway Communication Pattern*, but with some of the *Device-to-Cloud* capabilities shifted closer to the devices at the *Edge* or *Fog*. Fog and Edge are intermediate layers between the Cloud and IoT devices where smart agents provide processing and/or storage much closer to the device layer, typically those smart agents stay in the *MAN* or *WAN*.

In order to shift IoT cloud computing capabilities to the Edge of the network, closer to the devices, some of the functionalities that need more computing power must be virtualized in the Edge [11]. In INTER-IoT, apart from studying typical gateway approaches for providing interoperability [12], it also introduces a new paradigm for IoT Gateways that adjusts to this new communication pattern: the

dual Physical-Virtual IoT Gateway. This gateway is decoupled in two parts, (i) the **Physical Gateway**, which only performs lightweight network level operations and data aggregation, typically instantiated in a resource constrained device, and (ii) the **Virtual Gateway**, which represents the virtual counterpart of the physical gateway but in a less constrained device or virtualized service [13]. With this new communication pattern, three different connection levels are present:

- **Device to Physical Gateway Network Level:** Comprises all the different radio and access network protocols that devices will use to connect to the physical gateway, usually in the PAN or LAN range.
- **Physical to Virtual Gateway Network Level:** Is the connection between the physical and virtual gateway. This network level resides in the Fog or Edge, usually in the MAN or WAN range. This connection should be fast, secure and robust and should handle sessions to allow roaming.
- **Virtual Gateway to IoT Platform Network Level:** In this network level, the virtual gateway will connect and share its devices' state and information with an external IoT Platform. In a typical scenario, an IoT Platform resides in the Cloud, therefore this network level resides in the GAN.

2.2 Architecture of the Solution and Components

The gateway architecture of INTER-Layer is shown in Fig. 1 [14]. It is designed considering always modularity in protocols and access networks, meaning that any access network (AN) can be inserted into the structure as long as it is interfacing accordingly with its corresponding controller. The same is true for the protocols and middleware (MW) modules. The gateway is build up so that once the system structure is functional, a split-up can be realized. Part of the gateway can be placed in the Cloud to allow functionalities that a physical gateway is not able to perform in an efficient way. The connector module is in charge of controlling the communication between the physical and the virtual part of the gateway. When connection is lost, the virtual part remains functional and will answer to requests of API and MW. There are three ways to connect to IoT sensors and actuators:

1. The lowest level where a connection can take place is at the Access Network controller (AN controller). This is for very simple sensors or actuators that either does not have or have very limited processing power and can be offline for longer time periods. Sensors of this kind are commonly battery powered, actuators may have a power grid connection but usually have very limited processing power. The AN controller will do all routing and will serve as a master or access point for the sensors, and afterwards the Protocol controller will manipulate the data and create the messages to be sent to the virtual counterpart of the gateway.
2. At the middle level, the dedicated sensors and actuators can be connected (COTS IoT devices). Usually these sensors and actuators have some dedicated communication protocol between the wireless sensor and some piece of electronics with

a small processing core. They are capable of handling their own access and protocol controllers, and can be connected through a dedicated extension module by implementing a Device controller.

3. At the highest level, the COTS IoT systems are found. They manage their own gateway, protocols and AN controllers. These systems can be connected via the connector directly to the Virtual Gateway of the Inter-IoT system. In any case, the related COTS system software would have to be modified to add specific connection capabilities in order to implement the reference Physical-Virtual Gateway communication protocol.

The architecture is composed of the following components:

- **Registry:** This component is responsible of registering all the devices with its multiple sensors and actuators in the gateway. It adds an entry in the Device Manager with the information about each sensor and actuator.
- **Device Manager:** The Device Manager is accessible to every other component that needs information of any sensor/actuator. The protocol and access network modules will call the Device Manager in order to resolve the metadata for each sensor/actuator.
- **Access Network Modules:** The Access Network modules provide the INTER-IoT gateway access to the following communication channels: WiFi, ZigBee, USB, LoRaWAN and other proprietary RF links accessible via SDR. They are in charge of establishing and terminating a connection with the sensor/actuator, requesting and sending data from/to them, and handling the data pushed by the sensor/actuator to the gateway, among other functionalities.
- **Protocol Module:** This components are located within the Protocol Controller and implement the specific features of any supported protocol (CoAP, MQTT, LWM2M, etc.) throughout standard interfaces towards the Protocol Controller and the Dispatcher.
- **Access Network Controller:** It allows access to the devices, providing the necessary interfaces between the devices and the protocol modules. The Device Manager configures the access network modules according to the registry.
- **Protocol Controller:** This component is located within the physical part of the gateway architecture and contains all the communication protocols supported by the gateway, implementing the common interfaces between those protocols and the other components such as the Gateway Configuration, the Access Network Controller, the Device Manager and the Dispatcher.
- **Gateway Configuration:** This component is duplicated in the virtual and physical part. Every other component can use this component to access the gateway configuration.
- **Connector:** It controls the communication between the physical and virtual part of the gateway.
- **Dispatcher:** The device sends a trigger to the Dispatcher whenever new data are available, being this component in charge of storing the new measurement data from the device into the measurement storage. Any update request or data request from upper layers (MW or API) will be handled by the Dispatcher. It will

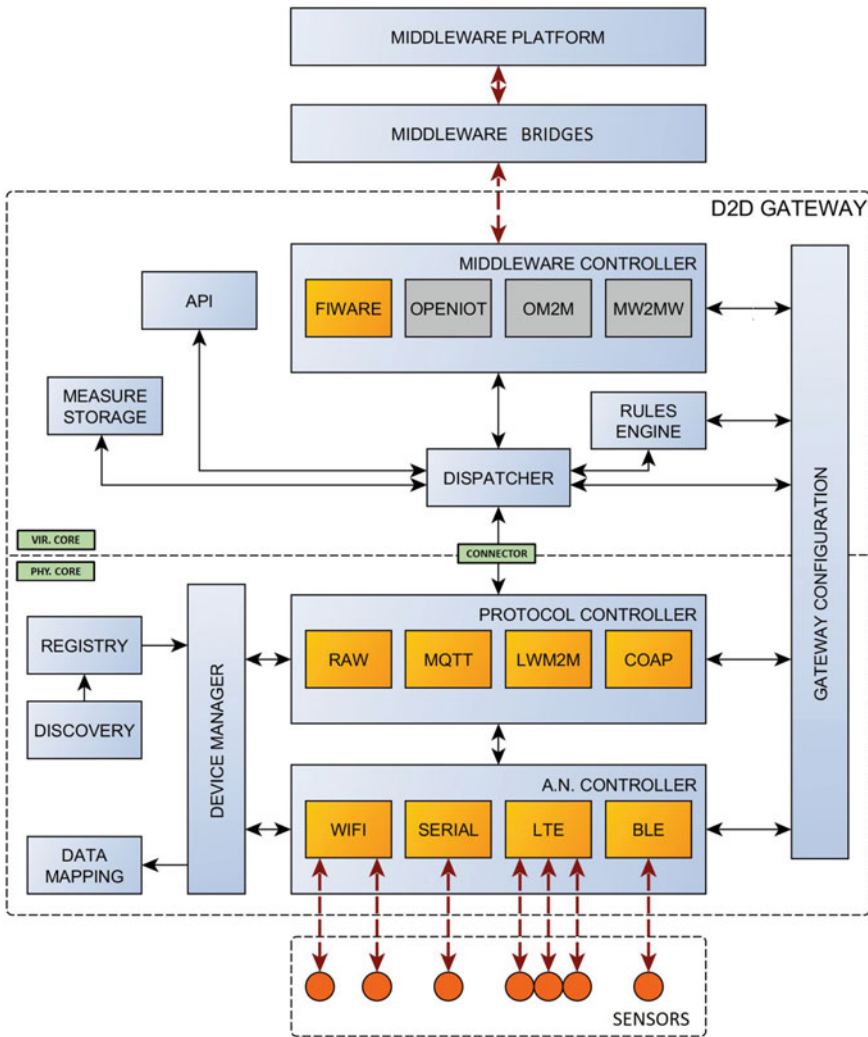


Fig. 1 Gateway architecture

get the latest data sample from the Measurement Storage and will send it to the middleware.

- Measurement Storage:** This component works as a cache in the gateway, storing the information about the devices connected and the last available value, in case of polling of these devices. If a platform requests the value, and the one contained in MS is practically new, or it is the last one obtained in case of disconnection, the value is returned in a faster way.

- **Middleware Module:** This Module is specific to a IoT Middleware platform and handles the communication of the gateway with the platform. It is in charge of registering the sensors and actuators to the middleware platform as well as processing the requests and responses exchanged with it.
- **Middleware Controller:** It wraps the active Middleware Module in order to have a common interface for the gateway. This component creates the connection to the MW platform and handles the messages interchanged between the module and the platform, as well as the messages sent to the Dispatcher.
- **Commons:** Even if it does not appear in the architecture, it is a basic component that includes several classes, methods and tools to be leveraged by the rest of components.

2.3 Implementation and Use Cases

In this section, the main technologies used for implementing the described architecture for interoperability at device layer are presented, and then some use cases in which the proposed solution has been utilized are briefly depicted. In particular, examples of integration at different levels of the device layer are showcased, including integration at device level, at physical gateway level and at virtual gateway level.

2.3.1 Implementation

The physical and virtual gateway implementation share a common base and runtime code. Both are based in an OSGi¹ framework wrapper (the OSGi framework has to be R4 compliant) with a customized bootstrap and initiation routines. This framework first load the third party libraries, then the core components and afterwards the extension modules. Finally, a routine for starting all the modules is launched, and the Physical and Virtual Core take the main thread to control the gateway. In Figs. 2 and 3 a schema and summary of the OSGi Framework, wrapper and components is shown.

This approach follows OSGi recommendations for a clear decoupled and modular system. As can be seen in the previous figures, these extensions can be developed to work in both parts of the gateway. Typically, physical extensions are centered in providing support for other device access network and protocols, creating new device controllers, whereas virtual extensions are centered in creating new middleware controllers to provide support for more IoT platforms. Common extensions can provide utilities to configurate or manage the gateway as a whole.

In the INTER-IoT device-to-device interoperability gateway, there are four different APIs:

¹ <https://www.osgi.org/>.

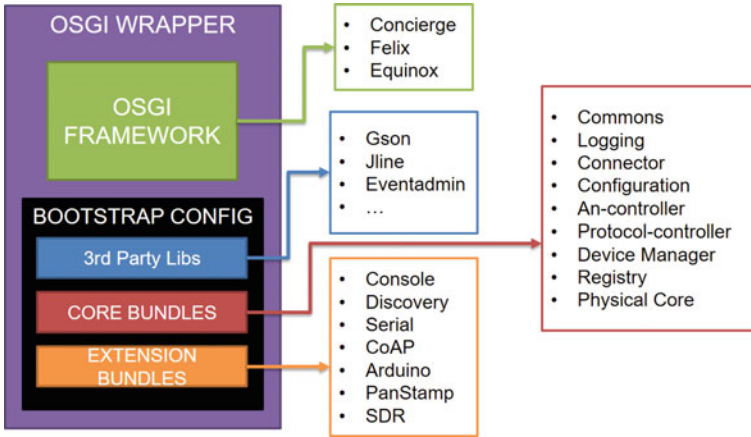


Fig. 2 Physical Gateway components

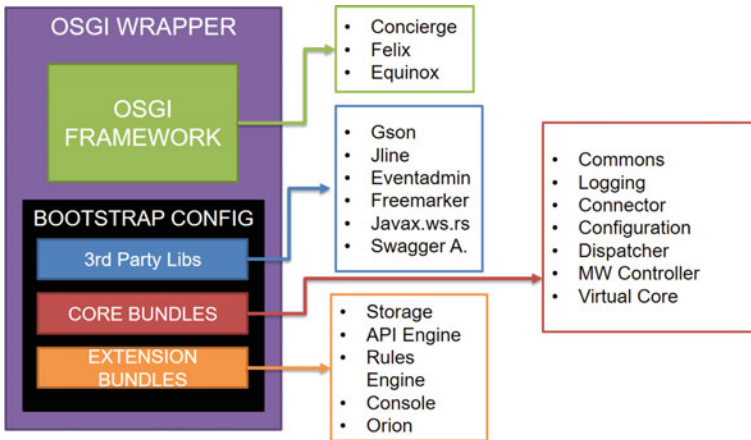


Fig. 3 Virtual Gateway components

- **Gateway CLI:** The gateway console extension provides a Command-Line Interface (CLI) to control the physical or virtual gateway instance.
- **Gateway REST API module:** REST API exposed by the virtual gateway API Engine extension module to interact with the virtual and physical gateway.
- **Physical/Virtual Communication API:** Messages exchanged between the physical and virtual through the connector module.
- **Programmatic API:** Libraries and interfaces needed to develop new extension modules for the gateway.

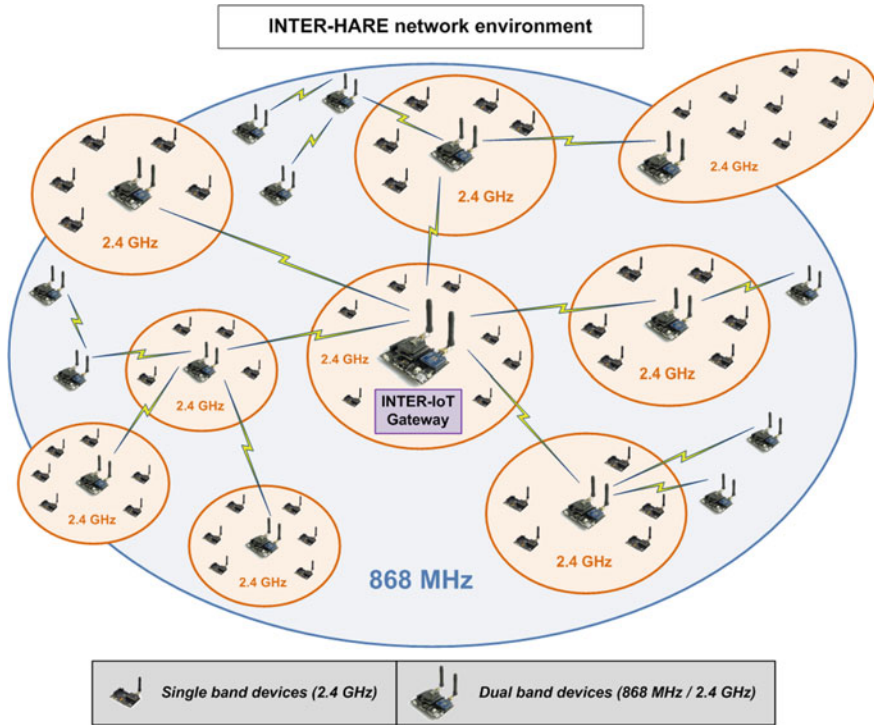


Fig. 4 INTER-Hare network

2.3.2 Integration at Device Level: INTER-HARE

The INTER-HARE project is intended to design a new LPWAN technology flexible enough to transparently encompass both LPWAN devices and multiple so-called low-power local area networks (LPLANs) while ensuring overall system’s reliability. A cluster-tree network is created [15], where the LPWAN acts not only as data collector, but also as backhaul network for several LPLANs, as shown in Fig. 4.

The communication within the LPWAN is based on the HARE protocol stack [16], which ensures transmission reliability, low energy consumption by adopting uplink multi-hop communication, self-organization, and resilience. The INTER-HARE platform is conceived as an innovative evolution of HARE protocol stack and can be considered as a dynamic multiprotocol by means of the integration with the INTER-IoT Gateway. The architecture of the INTER-HARE platform can be split into two networks with different purposes: the transport network and the integration network (as it can be seen in Fig. 5).

The transport network involves all internal infrastructures responsible for gathering and transporting information from the end-devices to the physical gateway. This internal infrastructure is formed by one single HARE protocol Gateway, several

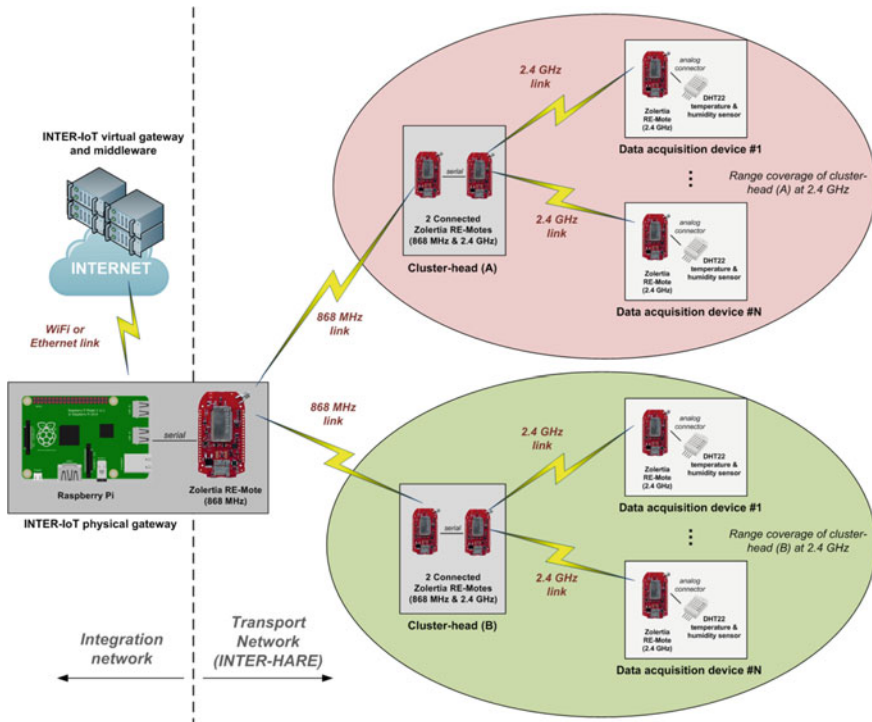


Fig. 5 INTER-Hare architecture

Cluster Heads (CH) and Data Acquisition Devices (DAD). The integration network is formed by the INTER-IoT Gateway, which enables access to the whole IoT stack. Communication between the Physical Gateway and the Hare Protocol Gateway, is done with serial UART communication protocol. The INTER-IoT gateway is therefore considered as the brain of the INTER-HARE platform and the single point of contact between the physical network and the rest of the INTER-IoT system.

2.3.3 Integration at Physical Gateway Level: SensHook

SensHook is a IoT node focused on the prevention and detection of disease-vector mosquitoes. The node is composed by a Smart Mosquito Trap capable of mimicking the human body (scent and respiration) and of automatically counting captured mosquitoes, identify the gender and the species. The information collected by each node is then sent to a server. In this manner, SensHook aims at reducing inspection costs while improving surveillance programs, being the first solution in the world to combine human mimicking with automatic pest information in their value

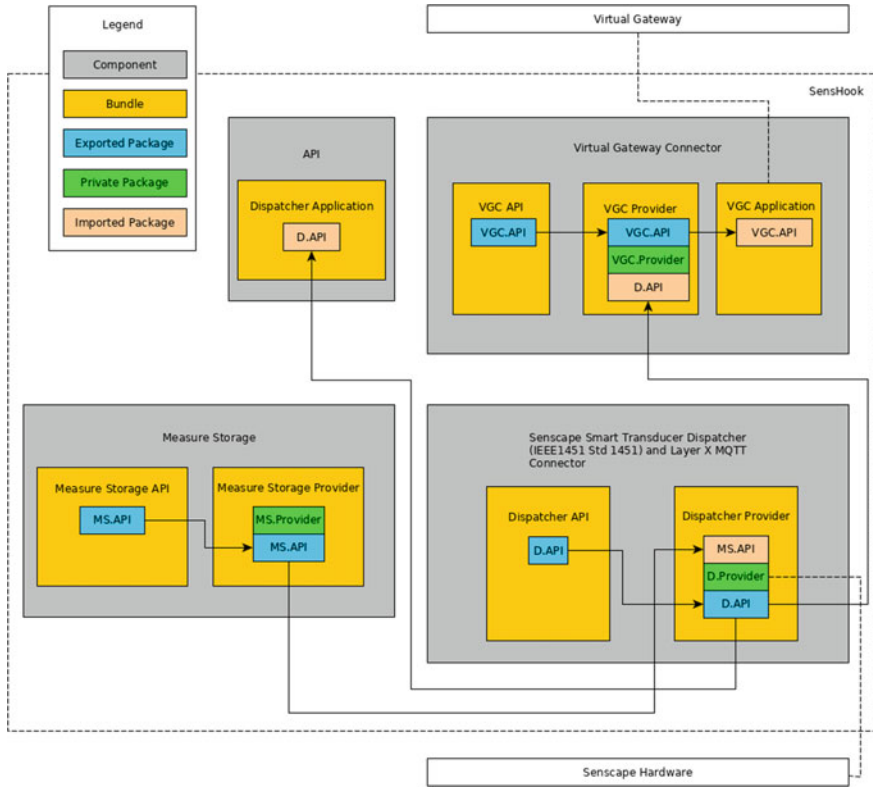


Fig. 6 SensHook architecture

proposition. This will allow a whole new population of consumers to establish surveillance programs that were only accessible to those with significant resources.

In this use case, the integration in this case is performed at physical gateway level. Despite the fact that SensHook provides their own platform for performing low level communication and computing, the capability of sharing the information of its devices with IoT platforms is not available. Hence, aiming at enabling it, a connection is made to the Virtual Gateway, by developing a specific connector integrated in the SensHook platform that understands the Physical-Virtual communication protocol as can be seen in Fig. 6.

2.3.4 Integration at Virtual Gateway Application Level: ACHILLES

ACHILLES is a project that provides an advanced access control and endpoint authentication to devices attached to the INTER-IoT gateway. In general, these devices are usually limited in storage capacity, power, energy and processing capabil-

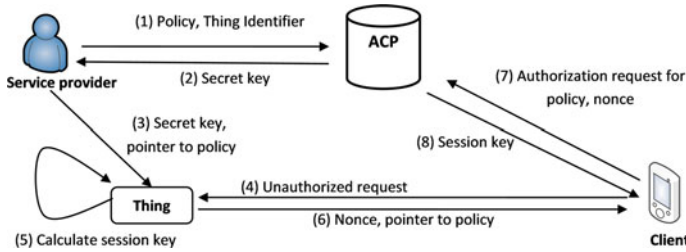


Fig. 7 Achilles architecture

ities, presenting security risks in IoT deployments. Since these devices are not usually able to perform complex cryptographic operations, security management becomes an impossible task from the device perspective. ACHILLES project overcomes these limitations by allowing the delegation of security operations to a third party (ACP, Access Control Provider) which can be implemented by a trusted separate entity as depicted in Fig. 7.

ACHILLES is integrated in the gateway as an extension of the Virtual part. This extension implements the core client functionality and configuration, being able to perform read/write calls to the supported physical devices. The main idea of the ACHILLES concept is that IoT service providers store access control policies in ACPs and in return ACPs generate secret keys which are stored in the device (steps 1–2). These keys are generated, during a setup phase, using a secure hash with the device identifier as input. Additionally, devices are configured with pointers (e.g., a URL that points to an ACP and a particular file) to the access control policies that protect sensitive resources (step 3). Every time a client requests access to a protected resource (step 4), the device uses a secure hash function to generate a session key (step 5). The secret key used by that function is the key generated by the ACP, and the hash inputs are the pointer to the policy that protects the resource and a random nonce. The device transmits the nonce and the pointer to the client (step 6), which in return requests authorization from the appropriate ACP (over a secure channel, step 7). The ACP has all the necessary information required to calculate the session key: if the client is authorized, the ACP calculates the session key and transmits it back to the client (step 8). Providing that the device has not lied about its identity and the messages exchanged between the client and the device have not been modified, the device and the client end up sharing a secret key. This key can be used for securing subsequent communications (e.g., by using DTLS).

3 Network Interoperability

In the traditional OSI reference model for computer networking, the network layer is conventionally located within the third one, directly standing over data link layer (layer 2) and responding to the transport layer (layer 4). However, with the birth

of new radio access technologies and IoT protocols, this model had to adapt to be compliant with the IoT reference layer architecture, hence enabling the embracement of a large heterogeneous range of devices.

INTER-IoT understands the network layer of an IoT deployment as the protocols, systems and devices that work on layers 2, 3, and even 4 in some cases, of the traditional OSI model. IoT products encompass many different data communication scenarios: (i) some of them may involve sensors that send small data packets at low frequency without prioritizing timely delivery; (ii) others may involve storage capabilities to sustain periods when the communication link is down (e.g., Delay Tolerant Networks); (iii) some scenarios may need high bandwidth without having strict latency requirements; (iv) while others may need high quality, high bandwidth, and low latency. Besides, particular characteristics have to be taken in into account, such as the mobility of objects through different access networks, secure seamless mobility and backing of real time data among the network. The operation in highly constrained environments is also an important issue to analyze. Finally, the use of many heterogeneous protocols (6LowPAN, RPL, LoRa, SIGFox, etc.) and mechanisms (tunneling mechanisms over IP, GRE and 6LoWPAN, etc.) on IoT network level are problems that need of a network interoperability solution.

3.1 INTER-Layer Approach to Network Interoperability

The particularity of an IoT deployment network is the treatment of different types of data flows as well as protocols to support communication. The great challenge that interoperability in the network layer must face is caused by the following problems:

- Difficulty to manage large amount of traffic flows generated by smart devices.
- Poor system scalability, which difficulties the integration of new devices.
- Hard interconnection of gateways and platforms via networks used by different providers.
- Several devices with totally different radio network access have to be accessed from a single gateway as an access point.
- Management of device's mobility through different access points.
- Great number of heterogeneous protocols (6LowPAN, RPL, LoRa, SIGFox, etc.) and mechanisms (tunnelling mechanisms over IP, GRE and 6LoWPAN, etc.) at IoT network level.

One of the main approaches to face these problems is the virtualization of the network layer, providing an extra tier of abstraction that facilitates management, scalability, seamless support for smart objects mobility (roaming) and packet routing, hence allowing the design and implementation of fully connected IoT ecosystems. To achieve network-to-network interoperability, the solution proposed by INTER-Layer is based on virtualization and software-defined paradigms, specifically in two approaches: Network Function Virtualization (NFV) and Software Defined Networks (SDN). The following characteristics have been considered:

- Decoupling of data plane from control plane using the well-studied protocol Open-Flow.
- Virtualizing network services at the top of the architecture.
- Implementation of techniques for traffic engineering to handle different flows of data generated by sensors based on their priority.

3.1.1 SDN and NFV

Before getting into the details of the proposed interoperability solution, in this section it is described how these technologies operate and how they have been included in the INTER-IoT ecosystem. The term *virtualization* refers to the technologies that allow the decoupling or abstracting logical resources from the real physical infrastructure. Logical resources are named after the abstract vision that the software has of the physical resources of the system. The creation of these logical resources aims at offering a simpler high-level interface to isolate users and programmers from the details and characteristics of the internal hardware devices as storage, processor, memory or communication elements.

Virtualization can be applied in several domains. For instance, if the storage is virtualized, the real size and distribution of machines' storage is hidden and a logical division of this one is created, which can be leveraged by other elements. Virtualization of resources as processing capacity, as another example, can be useful for aggregating several CPUs to create virtual machines with higher capacity over a combined physical infrastructure. Besides, applying this concept to the components of the network receives the name of Network Virtualization, in which its physical resources (firewalls, routers, switches, load balancers, etc.) are virtualized and assigned to different virtual instances [17]. Network virtualization can stand for either aggregating physical networks into a single logical one, thus resulting in a Virtual LAN (known as external network virtualization), or providing network-like functionality within an operating system (internal network virtualization). In general, hardware and operating system virtualization are applied in the latter, obtaining a virtual network interface to communicate with. In this case, the Internal approach is exploited. The complexity and scale of today's data centres that are based in virtualization of machines makes network virtualization even more complex than traditional ones. Moreover, the hosting of new types of virtual machines as IoT platforms, virtual devices, or containers makes this approach a mandatory need. Hence, thanks to the implementation of virtualization, new architecture approaches can be deployed, as in the case of SDN/NFV.

Software Defined Networking consists in the separation of the network functions in two planes: the control plane and the data (or forwarding) plane. On the one hand, the intelligence and network state, as well as the management and routing algorithms, are centralized, and the underlying network infrastructure is abstracted from applications and services. On the other hand, the elements that compose the network, as switches, routers, etc., become mere forwarders which route the information in an efficient manner, according to flow tables which are filled according to

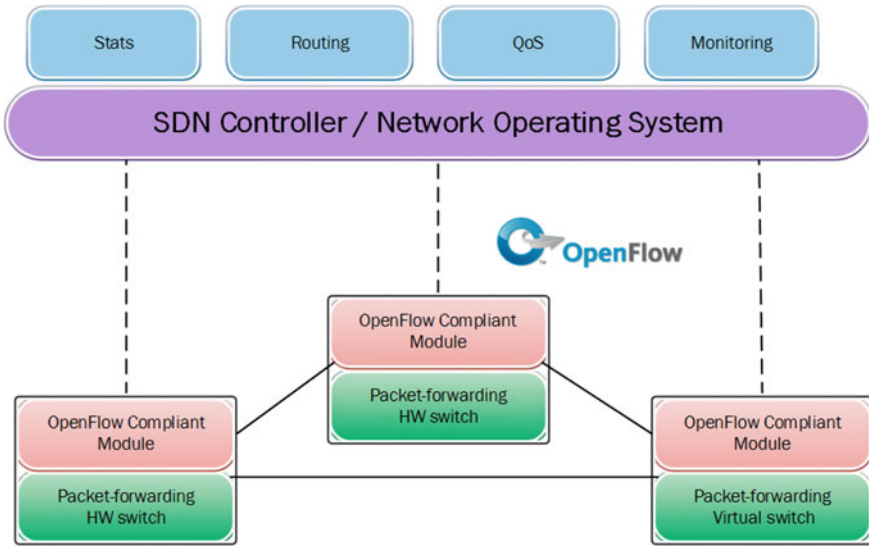


Fig. 8 SDN basic architecture and components

the decisions of the control plane. Thanks to the NFV approach, the aforementioned network elements are virtualized within generic servers instead of making use of dedicated single-purpose equipment, being thus NFV and SDN highly compatible and complementary [18]. The SDN basic architecture and components is shown in Fig. 8.

3.2 Architecture of the Solution and Components

The immense amount of traffic flows generated by smart devices is extremely hard to handle, and thus so is the scalability of IoT systems. Besides, creating the inter-connections between gateways and platforms is not a trivial task. Thus, the network-to-network solution aims at providing seamless support for smart objects mobility and information routing. It will also allow offloading and roaming, which implies the interconnection of gateways and platforms through the network. The approximation that INTER-IoT proposes uses the SDN/NFV paradigm, achieving interoperability through the creation of a virtual network, with the support of the N2N API. The implementation of the N2N solution in INTER-Layer is depicted in Fig. 9.

The data plane (lower components of Fig. 9) is composed of virtual switches. They are connected to each other in a determined topology and all of them securely connected to the controller. The upper part is the control plane, where the controller is located, provided with an OpenFlow connector to parse all packets coming from the network to the different services running on it. The connection with the forwarding

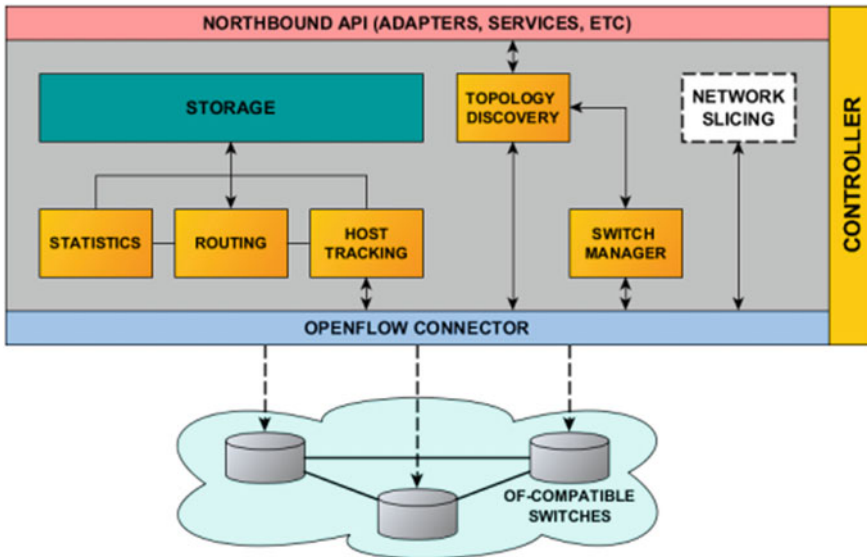


Fig. 9 Network-to-network architecture high level building blocks

devices is implemented by a southbound API, which allows the entrance of OpenFlow packets into the controller and formalizes the way the control and data planes interact. The core of the controller consists of different modules containing all the logic that will dictate how to route the packets as well as to obtain statistics. Specifically, the modules that compose the control plane jointly with the controller are:

- **OpenFlow connector:** It is an OpenFlow understanding plugin that communicates, by OpenFlow protocol, with all the switches that conforms the virtual network. Is the Bridge between the Controller and the nodes of the network.
- **Switch Manager:** The Switch Manager API holds the details of the network elements. When a network element is discovered, its attributes (e.g. what switch/router is, version, capabilities, etc.) are stored in the database by the Switch Manager. Hence, it has all the information about the nodes of the network, the number of switches, their configuration and state, etc.
- **IoT Routing:** In this module, some headers of the packets are introduced to perform a routing algorithm previously configured and resolve the next hop in the network.
- **IoT Host Tracking:** Module in charge of handling the information from a host, including the address, the position in the network, etc. It tracks the location of the host relatively to the SDN network topology.
- **Statistics:** This module storage and provides information about the number of packets analyzed through the Controller. It can return the number of packets attending to some filters.

- **Storage:** In this module, the information about statistics, topologies, direction, and other data related with the network is stored and updated for other modules to access them.
- **Topology Discovery:** It contains a set of services that allow conveying topology information. It keeps track of the nodes in the network along with their links, and creates a graph representing the state of the network with additional information about the state of the links.
- **Northbound API:** It has been created and exposed so upper applications and services can configure the controller or gather data from it.

3.3 *Implementation and Use Cases*

In this section, the main technologies used for implementing the described architecture for network interoperability are presented, and then some use cases in which the proposed solution has been utilized are briefly depicted.

3.3.1 **Implementation**

In order to implement the aforementioned modules, the most suitable available technologies have been implemented for creating a fully virtualized network with QoS capabilities that enables connectivity between the components that traditionally conform an IoT deployment. On the one hand, the virtual switches of the data plane have been implemented through Open VSwitch.² They are connected to each other in a determined topology, being all of them securely connected to the controller through TCP/SSL, leveraging the OpenFlow northbound protocol to update the flow tables and the OVSDB management protocol to retrieve information about their status and other statistics. On the other hand, the control plane contains the controller (RYU³), which provides OpenFlow and OVSDB connectors to parse all data packets coming from the network to the different services running on top it.

OpenVSwitch as a Virtual Switch

Open vSwitch (OpenVSwitch) is a production quality, multilayer virtual switch designed to enable massive network automation through programmatic extension, supporting standard management interfaces and protocols. Among its features one can find:

- VLAN 8021.Q support with trunk and access ports.
- Traffic flow-based statistics (NetFlow and sFlow).
- Traffic mirroring for monitoring (SPAN and RSPAN, among others).
- Link aggregation and bonding with LACP.

² <https://www.openvswitch.org/>.

³ <https://ryu-sdn.org/>.

- Routing with Spanning Tree (STP).
- Quality of service management.
- Traffic queuing and shaping.
- Tunnelling (GRE, VXLAN, etc.).
- Security: VLAN isolation and traffic filtering.
- Automated Control: OpenFlow/OVSDB management protocol among others.

OpenVSwitch provides a more complex design than simple “bridges”, being these the basic components to be used but, whereas bridges are only executed in host kernel space, the virtual switch makes use of both kernel and user spaces, which allows creating more complex rules of packet processing. The main components that compose a virtual switch are: (i) **ovs-vsitchd**, which is a daemon that implements the switch, jointly with a compilation of the Linux kernel module for flow-based switching; (ii) **ovsdb-server**, a lightweight database server to store and obtain switch configuration; (iii) **ovs-dpctl**, which is a tool for configuring the switch kernel module; (iv) **ovs-brcomptd**, which is a daemon that allows ovs-vsitchd acting as a substitute of Linux bridge; (v) **ovs-vsctl**, a command for queuing and updating the configuration of daemons; (vi) **ovs-appctl**: which is a utility that sends commands to the switch daemons that are running; and (vii) **ovs-ofctl**, a utility that implements the OpenFlow protocol to communicate with the controller.

The programmability and virtualization capabilities of OpenVSwitch have motivated its selection to deploy and manage the INTER-Layer virtual network solution. Besides, this switch supports different versions of the OpenFlow protocol, so it can be programmed to make specific actions with specific data flows. Hence, after the processing and decision-making that takes place in the specific modules of the controller, the adequate flow entry is inserted in the tables of the switches so that when data packet arrive it is already prepared to execute the necessary actions (forwarding, dropping, etc.).

OpenFlow as Southbound Communication Protocol

OpenFlow was the first SDN standard defined and vital element of an open SDN architecture. It is a communication protocol that gives access to the data plane as well as to the remote programming of network switches and routers over the network. This protocol decouples the intelligence required to route a packet from the act of forwarding it through the correct interface of the router, switch or network component, thus enabling the remote programming of the forwarding plane. This is achieved by inserting flow tables, designed by the protocol, within the switches managed by the controller.

The flow entries that compose the flow table are inserted and managed in the virtual switches by this protocol. They require three fields: (i) Match Fields, which defines a set of ingress ports, packet header fields and other metadata; (ii) Instructions, which is the action that the virtual switch has to make when a match is found (ports and fields of the data matches with the ones defined on the first field), and (iii) Counters, in charge of updating the number of packets matched against the Match Field. The OpenFlow protocol also allows representing additional methods of forwarding using group

entries to classify and manage groups of flows. These entries are: Group Identifier, Group Type, Counters and Action Buckets. With the aid of this protocol and the programmability of the switches, different policies can be applied to manage the flows coming from the devices through the gateway. Additionally, with the information provided by the headers of the protocol, informative statistics can be obtained so the controller has a better overview of the state of the network and the flows being carried out [19]. Besides, the Quality of service possibilities implemented by OpenFlow includes:

- **Queues:** Associated to a port, define a priority treatment depending on the configuration, and could define the rate of the packets.
- **Rules:** Implemented in the queues, define the aforementioned treatment.
- **Meters:** Switch element which measures and controls the ingress rate of packets, i.e., the rate of packets prior to the output.

This protocol has several stable releases, starting from 0.8 and being versions 1.1 and 1.3 the most used ones. There are not many differences among version except for some QoS aspects like:

- *OF1.0:* In this version, an OpenFlow switch can have one or more queues for its ports. It is also possible to read/write headers for VLAN priority and IP type of service.
- *OF1.1:* This version improves the matching and tagging of VLAN and MPLS labels and traffic classes.
- *OF1.2:* Supports querying all queues of a switch, and introduces the OF-CONFIG protocol 5 to reconfigure queues within the switch. Max-rate property can be set to the queue. Flows can also be mapped to queues attached to ports.
- *OF1.3:* This version introduces meters.

In INTER-Layer, the controller chosen to communicate through this protocol will support all versions in order to connect with legacy switches that have implemented one of them. The network layer provides a QoS API in order to satisfy the potential QoS requirements of the deployment. When using the QoS API of Inter-IoT, the developer can add/delete/monitor rules, queues and meters. Rules determine whether the specified traffic is assigned to a certain queue or meter. Queues are designed to provide a guarantee on the rate of flow of packets placed in the queue. Different queues at different rates can be used to prioritize specific traffic. And meters complement the queue framework already in place by allowing for the rate-monitoring of traffic prior to output.

OVSDB as a Southbound Protocol to Manage OVS Database

The state of OpenVSwitch is stored in a database server. The Open vSwitch Database management protocol (OVSDB) is used to manage this database, thus leveraged for controlling the cluster database and determine the configuration of the virtual switch including its ports, bridges, interfaces and other important switch information. The OVSDB Protocol uses the JavaScript Object Notation (RFC 4627 [20]) for its schema and wire protocol format, and JSON-RPC 1.0 for its wire protocol.

The differences between OF-CONFIG and OVSDB protocols are several. The most important one is related to the fact that OVSDB is focused on the configuration of virtual switches implemented with OpenVSwitch while OF-CONFIG is focused on the configuration of physical switches. Still, vendors are also tending to implement OVSDB within their physical switches. These protocols are quite different regarding encoding, features, commands, etc. depending on the characteristic to be configured.

Ryu as a Base Controller

The component-based SDN framework Ryu has been chosen to handle the control plane and manage the virtual switches that compose the network-to-network solution. Ryu is simple, modular and highly designed to increase the agility of the network through its management and versatility. It is composed by a main component, Ryu-manager, which is in charge of (i) providing the environment where the different modules and applications will run, and (ii) the communication between the different modules. Besides, some modules have been implemented to extend the capabilities of the controller and adapt it to the particular needs of the network interoperability solution. Some of these modules are: (i) the Topology Discovery, which is in charge of obtaining the network information to create a graph that represents the current state of both the network and its components; (ii) a statistics module for accounting the number of packets processed, dropped or queued; (iii) the IoT Routing and Host Tracking modules, which goals are to create and manage the routes that each packet has to take to reach their destination; and (iv) a set of modules related with QoS and Security to prioritize some traffic flows and isolate them from others in order to create network slicing over the virtual infrastructure. Since the controller is entirely developed in Python 2.7, the modules created at the top of it to conform the network solution have been also developed in this programming language. Finally, a Northbound APIs REST-based have been developed for providing access to the software components that compose the controller and facilitate the deployment of new applications by future developers. This API is aggregated to other layers' APIs and published through INTER-API, which is presented in the following chapter.

3.3.2 Use Case: Traffic Priority in E-Health Environment

This use case implements the network-to-network solution with virtualized functions and central management of the cloud in an e-health environment. The virtual network is managed from a central monitoring application, using the API to request topologies, statistics, historical, etc. Additionally, the implementation of the SDN paradigm allows prioritizing data flows using traffic engineering and QoS. A real proof of concept was designed in order to provide an example of the usability of the solution. In this use case, the scenario presented a Central Hospital with different care houses in charge, located around the city. In these care houses, there were nurses that take care of the patients, measuring different health values (heart rate, temperature, sugar in blood, etc.) with the devices located at those houses. However, this information

never leaved the care houses, and thus the doctors had to visit continuously each one of them to gather the information, with the consequent cost that it entails.

The proposed architecture was based on the implementation of the network-to-network solution together with the device-to-device solution. Each care house was provisioned with a physical gateway that could obtain the information measured by the health tools (pulse meter, thermometer, glucose monitor and others). The physical gateway is directly connected and synchronized with its virtual counterpart located in a private cloud at the Central Hospital. This cloud was designed by means of the virtualization and SDN solution proposed by the INTER-Layer network approach. This way, the different virtualized gateways, together with other resources as IoT platforms or applications, were able to communicate through the virtual infrastructure that can be automatically re-configured.

Once the information arrives at the Central Hospital border router, which is connected to the SDN network, this information is automatically routed to the proper virtual gateway instantiated in the private cloud. Once the information arrives at this point, it can be filtered, aggregated or dropped off, and afterwards, all data could be forwarded to another IoT/Big Data Cloud Platform for future processing. One of the advantages of the implementation of SDN techniques, in this case, is the scalability that brings to the network. For instance, if a new virtual resource has to be deployed in the cloud infrastructure, it can be done in a seamless manner and its connection and configuration is made quite straightforward. Moreover, a different doctor could have access to a subset of care houses, hence to a subset of virtual resources. The possibility to define network slices within the SDN networks allows the separation of sets of virtual resources in order to define roles of data access, thus bringing privacy to patients and doctors. Finally, as the controller provided QoS capabilities, the prioritization of specific types of traffic brings a myriad of advantages to this scenario. For instance, data from a care house with severely ill patients or from a specific type of health devices can be prioritized in order to not get lost, for lowering the latency to the destination, or to trigger an alarm to notify the expert in charge.

4 Middleware Interoperability

Middleware refers to the software and hardware infrastructure that enables communication between different system components, usually in either request/response fashion or a sustained connection communication for data streaming.

Middleware thus abstracts several aspects of an end-to-end communication including the service name, address and location, the message transport protocol, service instance, interoperability features, etc. For example, a client can issue a request to a service without knowing which instance of that service will communicate with, thus hiding some of the complexities of service scalability. Middleware is also a convenient layer for placement of additional system-wide meta-services such as security, anonymization, auditing and monitoring.

4.1 INTER-Layer Approach to Middleware Interoperability

In the IoT domain, there is a need for dedicated and powerful middleware technologies to take the critical role of interconnecting the heterogeneous ecosystem of applications communicating over several interfaces using and operating on diversified technologies. Interoperability, context awareness, device discovery and management, data collection/storage/processing/visualization, scalability, privacy, and security are among the most significant aspects that have to be addressed by such solutions. Development of middlewares in the IoT domain is an active area of scientific and industrial research, and a number of interesting solutions have been developed so far [21, 22] (Fig. 10).

Due to the intrinsic difficulty to define and enforce a common standard among all these complex scenarios, IoT platforms have to provide an abstraction and adaptation layer to applications from the things and offer multiple services by means of easy-to-use, yet powerful APIs. However, there is no clear division line between

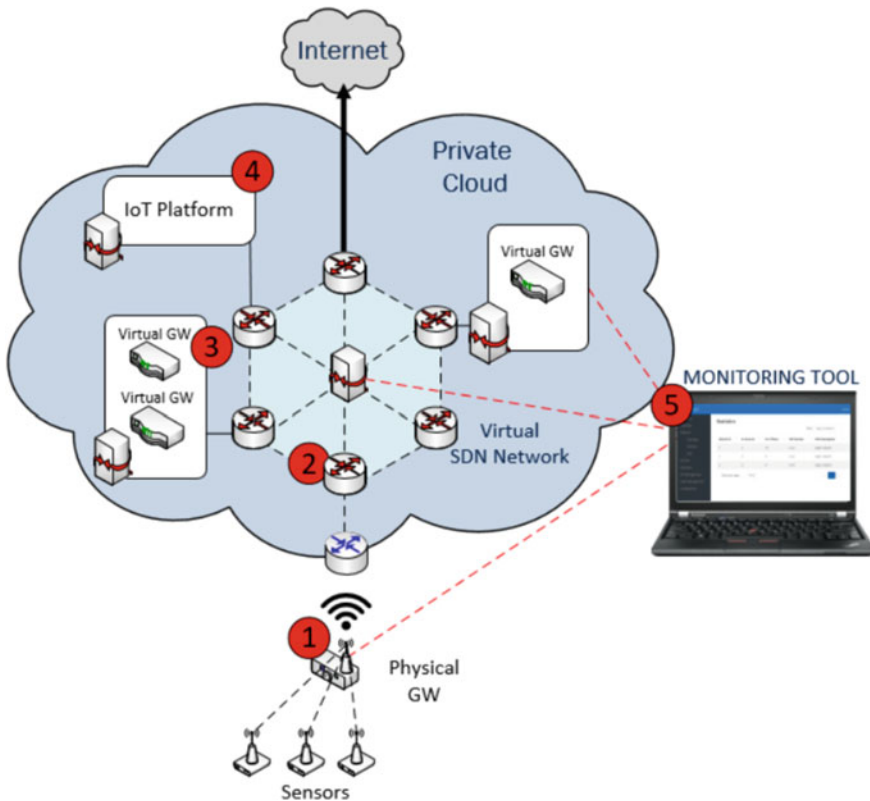


Fig. 10 Use case on Health Vertical

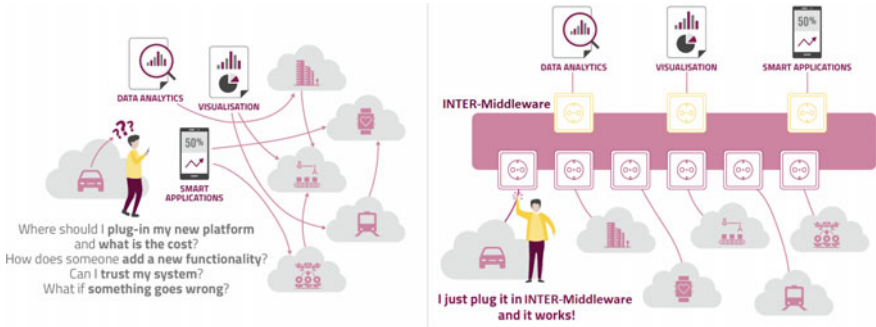


Fig. 11 INTER-middleware complex usage scenarios made simple

a middleware solution and IoT platforms. Most IoT platforms provide some middleware functionalities, although their focus is on providing efficient IoT platform services and not on solving interoperability issues [23].

Figure 11 shows a scenario where a system composed of ad hoc solutions has been created. This approach exponentially increases the complexity (and thus development and operating costs) of the system with the addition of each new application or platform. As the system grows, it creates a complex network of different communication channels between applications and platforms. The main challenge is to interconnect systems that were never meant to work together, in a user-friendly, cost-effective, transparent, and secure way. This raises many issues, introduces technical, legal, privacy and security risks, and often places an insurmountable hurdle in front of delivery of such innovative systems. As a result, we can claim that developing value-added services on top of IoT platforms is expensive and time-consuming.

To address this issue, a middleware-to-middleware interoperability solution called INTER-Middleware has been created. At the time of writing, INTER-Middleware is in the incubation phase, where a robust product is being created from project results that have been validated in e-health and port logistics. We present in this chapter an IoT-focused middleware solution dedicated to the provision of the most critical interoperability services in the IoT domain.

4.2 Architecture of the Solution and Components

INTER-Middleware architecture (Fig. 11) provides core functionalities related to facilitation of interoperability among IoT middleware platforms as well as provision of a common abstraction layer to provide access to IoT platform features and information. The architecture acts as an abstraction layer unifying the view on all interconnected platforms, devices and services. Moreover, its scalability permits the easy addition of new IoT platforms. INTER-Middleware architecture is composed by five main components (Fig. 12):

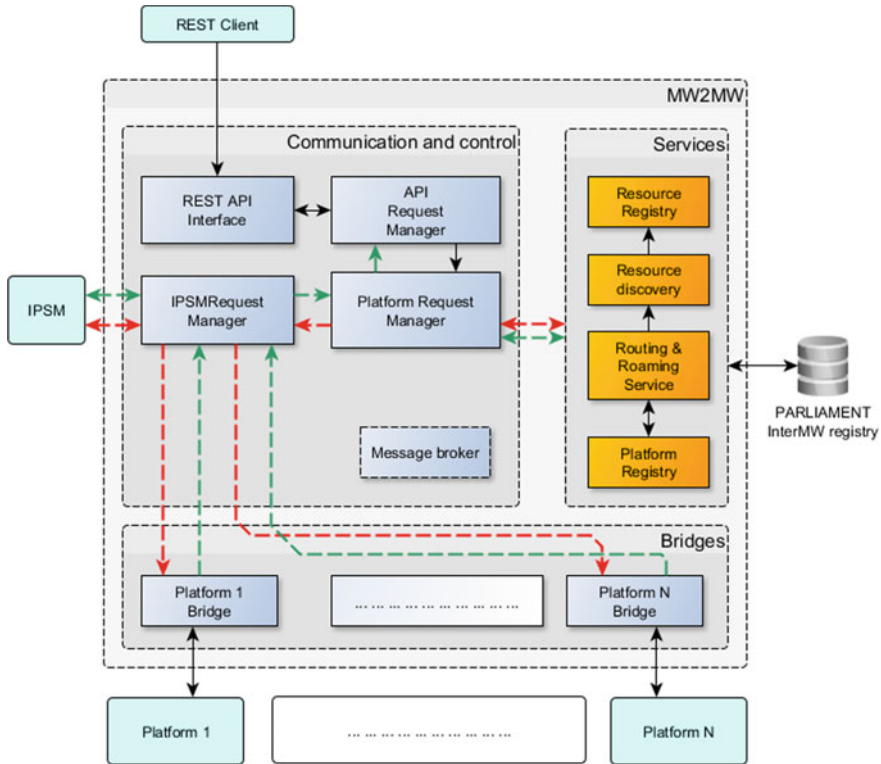


Fig. 12 INTER-middleware architecture

- **Ontology:** Common ontology to represent all messages routed through the system.
- **Bridges:** Act as a middleman between INTER-Middleware and IoT platforms.
- **Communication and control:** Orchestrates all the communications that take place between the different components of the architecture
- **Services:** Common services offered by INTER-Middleware to facilitate interoperability between platforms.
- **REST Interface:** Extends the usability of the abstraction layer by exposing this functionality through a widely used technology.

4.3 Implementation and Use Cases

4.3.1 Ontology

Data model, used in INTER-Middleware, is based on the ontological reference model of meta-data developed in INTER-IoT. It includes core concepts, shared between

IoT platforms, that have been identified and standardized in ontologies, such as SSN (Semantic Sensor Network ontology). Using one common model for all internal INTER-Middleware components improves the efficiency of internal data transfer, as well as allows components to make assumptions about structure and content of data, so that rich functionalities specific to IoT domain can be implemented and offered in one common data model. INTER-Middleware uses the common INTER-IoT ontology (GOIoTP) to represent all messages routed through the system. It is implemented through JSON-LD messages and is in the core of INTER-Middleware, thus tightly coupled with the common abstraction layer that unifies the view on all interconnected platforms, devices and services. It does not matter what device belongs to what platform, or what service is in which platform. There is, however, no requirement of compliance with the INTER-Middleware data model placed upon IoT platforms that use it. Data models of platforms participating in communication through INTER-Middleware are mapped to ontologies and semantically translated in IPSM. As a result, the commonalities between data models of IoT platforms can be expressed through a common data model, despite the possibility of having different semantics.

4.3.2 Bridges

Interoperability at the middleware layer is achieved through the establishment of an abstraction layer and subsequent integration of all IoT platforms. INTER-Middleware open architecture is extended through the development of IoT platform bridges that provide specific functionalities to connect INTER-Middleware with an IoT platform. This way there is no need interconnect all platforms among themselves, but rather connect them to the abstraction layer and provide a mechanism for their communication within this layer. It supports actuation and subscription to observations as core IoT platform functionalities. Virtual devices management, which is basically mirroring devices across IoT platforms, is implemented for those platforms that support this functionality.

INTER-Middleware provides a common Java interface that defines bridge features that have to be implemented: subscriptions, actions, virtual devices management and discovery. One important step in bridges development is the implementation of a syntactic translator to/from platform-specific format and JSON-LD. Definition of rules for semantic alignments is still necessary, but not at the bridge level. That part of the process is fully implemented in IPSM. The integration with IPSM is achieved through the IPSM Request Manager component that orchestrates the communication between IPSM and INTER-Middleware components (Bridges, Platform Request Manager).

4.3.3 Communication and Control

Data flow is managed through the introduction of *conversations*. A group of messages belongs to the same conversation if they share the same unique conversation identifier. For example, in a single conversation we would typically have first a message which subscribes to a particular group of sensors, and then messages with sensor readings, going upstream from the sensors to the application. Subscriptions in INTER-Middleware are also tracked by the unique conversation identifier. Technically, data flows are implemented through a message broker. This allows complete decoupling between components as well as isolation of the communication responsibility in a single element, which in turn makes profiling, scaling and adaptation to enterprise infrastructures easier. An abstraction mechanism enables interchangeability of the message broker implementation [24].

4.3.4 Services

The INTER-Middleware solution maintains a registry of all devices present in connected IoT platforms and provides meta-information about those devices. The Parliament^{TM4} triple store database provides persistence and advanced querying mechanisms for the Services subsystem. All registry-related requirements that need persistence or querying support, such as Platform Registry, Resource Registry and Subscriptions Registry, are implemented through this database. This allows the implementation of an efficient querying mechanism and seamless access to device information across IoT platforms. Maintenance of the device registry is not a trivial task, as there are several approaches utilized by IoT platforms to provide meta-data about attached devices. INTER-Middleware implements discovery strategies that can be used to populate the registry: full-query at regular time intervals, difference query at regular time intervals or, with more advanced IoT platforms, registry updates with callbacks.

4.3.5 REST Interface

Applications communicate with INTER-Middleware through a REST API. It further extends the usability of the abstraction layer by exposing most of the functionalities through a widely used technology and making them available to application layer components. Requests and results may be provided in either a simple JSON format, that fulfils most of the basic user requirements, or in the more complex JSON-LD format that also offers a richer set of functions and full semantic interoperability.

Security at application level is provided through the integration with the REST API Manager and Identity Manager. Platform security, on the other hand, is a respon-

⁴ <http://parliament.semwebcentral.org/>.

sibility of bridge developers. In principle, authentication information can be passed through platform registration messages.

4.3.6 E-Health Use Case

In the area of health, INTER-Middleware has been used to develop e-Health application whose objective is the prevention of obesity-associated diseases through patient monitoring. The application integrates data collected from two different data sources, universAAL platform and Body Cloud platform. Both are health-focused IoT platforms that use Bluetooth technology to collect sensor measurements. However, they are not interoperable from a technological point of view. Instead of consuming data directly from the application through their APIs (leaving the resolution of interoperability problems as a task to be solved within the application), a new element is added to the architecture of the solution, INTER-Middleware, to be responsible for providing data interoperability between the platforms involved. To this end, it has been necessary to develop a platform bridges for each platform and connect them to INTER-Middleware.

Thus, the application consumes the data from the API provided by INTER-Middleware. In this way, if in the future it is desired to incorporate new platforms or devices associated to the platforms to the solution, these changes will be made in the interoperability layer, being transparent for the health application. An exhaustive description of the interoperability application in e-Health can be found in the Chap. 8.

4.3.7 Port Logistics Use Cases

INTER-Middleware has been validated in a port environment through three use cases whose main purpose is to improve the efficiency of resources in the transport chain of a port system through the monitoring and automation of processes involving different actors. The main actors are:

- Port authority: organism that manages the collection of different terminals, facilities and auxiliary systems that enable the activity of the port itself.
- Container terminal: installation or set of port installations constituting the interface between the mode of maritime transport and other modes.
- Haulier company: company that owns the fleet of trucks that access the port daily.

On the other hand, 3 use cases have been defined where each of them is focused on solving a different problematic:

- IoT access control, traffic and operational assistance.
- Dynamic lighting.
- Wind gusts detection.

The main challenge presented by this scenario is precisely the difficulty of interacting between systems that have not been designed to work together. The port authority

IoT platform is based on WSO2 whereas the container terminal uses its own IoT platform (SEAMS) and the haulier company has an Azure IoT platform in the cloud.

In order to establish a common base on which to build the new solutions associated with the different use cases in a user-friendly, secure and scalable manner, it was decided to use the INTER-Middleware interoperability solution. For this purpose, the systems are integrated with INTER-IoT through the middleware layer and the API layer. To this end, each system implements a specific platform bridge. An exhaustive description of these cases can be found in the INTER-LogP chapter.

5 Application and Services Interoperability

There are multiple types of services in IoT ecosystems, such as Complex Event Processing, Historical Database, Big Data Processing, Visualization, Analytics, etc. IoT platforms do not have the capability to interact between each other at application and service level. This lack of interoperability is mainly produced by the heterogeneity of the services, the different domains involved, the lack of standardization of the technologies and the large amount of protocols involved, which in the end prevents the emergence of vibrant IoT ecosystems [25]. There are main aspects to consider in order to achieve interoperability at this layer:

- The native access to the IoT Platform services. It should be considered as a method to access IoT platforms applications and services. Most IoT platforms provide a public API to access their services, APIs that are usually based on RESTful principles and allow common operations such as PUT, GET, PUSH or DELETE. However, there are other IoT Platforms that do not include a REST API or SOAP for easing the development of Web services, but provide different ways to interact with them.
- The use of wrappers. The term wrapper in this context refers to a specific program able to extract data from Internet sites or services and convert the information into a structured format.
- The creation of enablers to the applications and services. They guarantee, organize and simplify access to the IoT Platform services.
- The development of application, data and device catalogues dedicated to the IoT services. They are generally missing in the market. A solution based on a Service Catalogue will be able to register applications to make them discoverable. Furthermore, it will offer a description or detailed information about services/applications.
- The virtualization of services and applications. There are some benefits like simplifying the monitoring of the infrastructure, network issues and security incidents. Furthermore, it provides flexible mechanisms like the creation of additional instances of the services whenever needed. This allow to handle the additional load while maintaining the quality of the service.

Service composition solutions facilitate achieving interoperability between applications and services [26]. These solutions encompass all those processes that provide

added-value services, so-called composite services, from existing services in the IoT platforms. Service composition can be understood as allowing routes for the data to be treated before reaching an application or end-user, or agents requiring information or processes from other such agents. Such compositions can help provide more valuable information and actions than plain raw data, tailored to a particular receiver or purpose. The composition of such services can be as simple as one service making use of a second service, to very complex and flexible schemes of interconnection that need a coordinator to manage the mesh of requests and responses. There are several techniques of service composition like mash-up, orchestration, choreography of flow based programming and tools to facilitate its implementation. In the solution that will be described in the following subsections, the flow based programming approach is the selected technique.

5.1 INTER-Layer Approach at Service and Application Layer

The main objective at this layer is to guarantee a solution that is capable to offer a layer of abstraction to achieve interoperability between the applications and services of IoT platforms. In order to provide benefits like access, use, import, export, catalog, discovery and combination of heterogeneous services between different IoT platforms (Fig. 13).

INTER-Layer approach provides a detailed plan about how to perform access to IoT platform services and implements a complete architecture to interact with these services and to create and manage new composed applications. The technique selected to create interoperability between services is Flow Based Programming paradigm. This paradigm defines applications as black-box process, which exchange data through predefined connections with message passing. These black-box processes can be connected to create different solutions without the need of being modified internally [27].

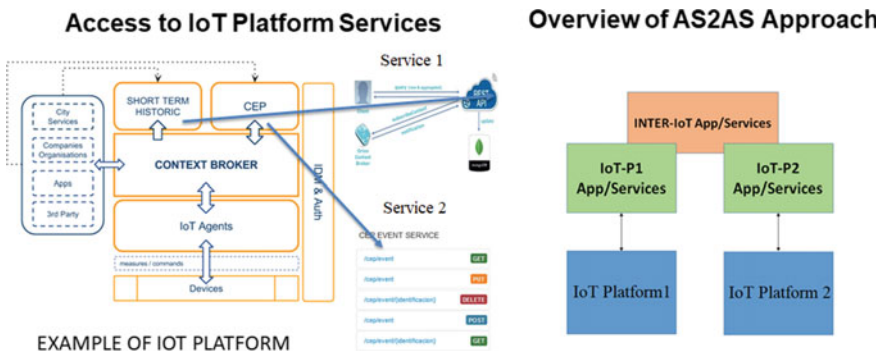


Fig. 13 AS2AS basic overview

The main functionalities offered at AS2AS level are based on access to APIs or interfaces provided by IoT platforms, register those services/applications with their description or detailed information to make them accessible, offer requests to the catalogue to obtain the necessary services from the IoT Platforms, providing an abstraction layer of interoperability that facilitates the common access to these services and helping developers to make them interoperate with others of the catalog.

5.2 Architecture of the Solution and Components

The following architecture [28] is designed to perform the functionalities and goals that have been listed in the previous subsection (Fig. 14).

The components and the relation that exists between these components are the following:

- Service Catalogue and Service Discovery are in charge of storing and managing the information and description about the services available on IoT Platforms. To interact with these components, users can make use of the graphical environment (GUI): Modeller and Register Client.

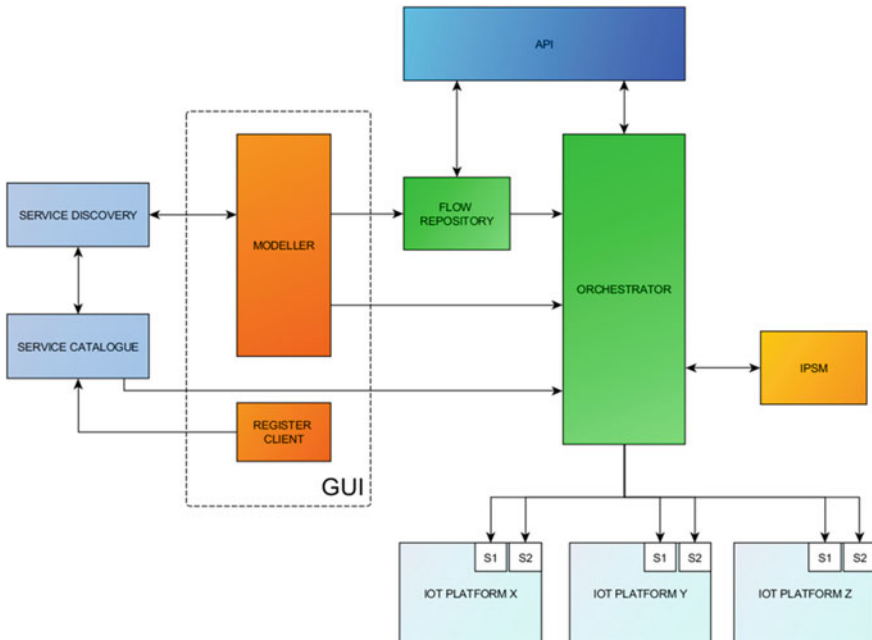


Fig. 14 AS2AS basic architecture and components

- The Register Client provides a tool to register new native IoT platform services and new composite services (also known as flows). During the registration of a service, it is possible to add a description about its features. Once the registration of the service occurs, it is stored in the Service Catalogue.
- The Modeller is a graphical environment that has access to the services that have been registered. It uses the Service Discovery module, through which it calls the Service Catalogue. In addition, it can access to util internal functions to execute a particular process (for example, functions to perform transformations in the data resulting from the execution of a service, to display information, to determine a timeout between calls, to repeat a call to a service a number of times, etc.) that facilitate the interaction with the available services. Using this tool, the AS2AS users can design a solution based on the composition of services. The visual editor lets user drag and drop the services (visually represented as nodes) onto the design surface and then join them together by dragging lines between them. Once the design made by the Modeller is validated, the generated flow is stored in the Flow Repository. This component manages the information of all flows created.
- The Orchestrator is the central engine of the interoperability solution. It is responsible for loading the flows created with the Modeller and stored in the Flow Repository. Once the design is loaded, it makes the necessary calls to the service APIs of the IoT Platforms Services and executes its internal functions, in the order indicated in the model to run the service composition. It collaborates with the semantics module, which is responsible for performing semantic translation of data exchanged.

The orchestrator and modeller are based in Flow Based Programming paradigm. For that reason, it is necessary to take into account that in this technology the main element is the *node*. These nodes provide a mechanism to access and interact with the IoT services. A node needs input parameters and provides output information. It executes a series of internal processes in the application that is calling. The interaction between the different nodes will be defined by an execution flow, which defines and manages this interoperability process between services.

The Catalog, Register and Discovery components work with the nodes and its properties. The Modeller is responsible to create and modify the flows, which are composed by the interconnection of several nodes. The purpose of the Modeller is to define the service composition. These flows are stored and loaded in the Flows Repository. Finally, they are executed by the Orchestrator, which is the one who starts the service composition operation.

Finally, the API is responsible to manage the Orchestrator and the flows stored in the Flow Repository. It offer the functionalities of a process manager, allowing users to start/stop a flow of execution, view its status, load a flow of interoperability in the orchestrator or access to specific services.

5.3 *Implementation and Use Cases*

The core solution is based in Flow-Based Programming using as core a tool, Node-RED,⁵ that implements this paradigm according to the AS2AS INTER-Layer interoperability requirements. This tool interacts with the components designed and developed to offer the interoperability solution. It transfers the advantages of this service composition paradigm to IoT. The solution enables a number of IoT services to be available in a development environment. Access to IoT services has been achieved by accessing its REST APIs and wrapping them through a node with a series of functionalities for the user. For those services that do not have REST API, other alternatives have been looked (e.g. SOAP web services).

5.3.1 **Node-RED Integration in INTER-IoT**

Node-RED offers a visual tool for wiring together hardware devices, APIs, IoT Native Services and online services [29]. From the point of view of INTER-Layer, Node-RED offers tools for creating new nodes for IoT services, as well as a legacy and huge core set of useful nodes. These nodes can be stored in a catalogue. Users can search available nodes in the catalogue or in the npm repository. New nodes can be registered and installed, and existing nodes can be enabled or disabled. In order to design and orchestrate interoperability, Node-RED provides a browser-based flow editor that makes it easy to wire together flows using a wide range of nodes in the palette. This flows can be deployed in runtime with a single-click. In addition, it allows users to save useful functions, templates or flows for re-use [30, 31]. Finally, it offers an API to remotely administer the runtime.

5.3.2 **Nodes**

Paying attention to technical issues, a node consists in a JavaScript file that runs in the Node-RED service, and an HTML file consisting in a description of the node. The description appears in the node panel with a category, colour, name and icon, code to configure the node, and help text. Nodes can have at most one input, and zero or more outputs. During the initialization process, the node is loaded into the Node RED service. When the browser accesses the Node RED editor, the code for the installed nodes is loaded into the editor page. Node RED loads both HTML for the editor and JavaScript for the server from the node packages.

⁵ <https://nodered.org/>.

5.3.3 Flows

The flows are a collection of nodes wired together to exchange messages, the data contained in the flow is stored in a file in JSON format. It consists of a list of JavaScript objects that describe the nodes and their configurations, as well as a list of downstream nodes they are connected to, the wires. Wires define the connections between node input and output endpoints in a flow. The messages passed between nodes in Node-RED are, by convention, JavaScript Objects called messages. Messages are the primary data structure used in Node-RED and are, in most cases, the only data that a node has to work with when it is activated. This ensures that a Node-RED flow is conceptually clean and stateless.

5.3.4 Implementation

Regarding a mapping between the architecture and the implementation, it is necessary to consider how to access to a complete instance of the interoperability solution. For that reason, during the implementation it has been taken into account that the flow designed by the modeller and executed by the orchestrator can be accessed by one or more users, at the same time and with different permissions, through the APIs. Still, only one flow can be executed in each instance of the solution. Therefore to have several flows of the interoperability solution running at the same time, it is necessary to work with different instances of the solution.

The core solution could be virtualized inside a docker container image allowing deploying the solution in the same way regardless of the environment. It allows to offer different instances of the interoperability solution located at the same host. Each instance of the server have different nodes, its own internal folders and files with its running configuration. It should be highlighted that instances access the same service catalogue and flow repository.

Different graphical interfaces and web services have been developed to interact with the components at various levels. The first level is from the point of view of the management of instances, allowing users to generate and manage different instances of the solution dynamically. The second level is to communicate with each specific instance, to design flows and load configurations or services. The third level is to interact with the catalogues. The framework designed in INTER-IoT starts integrating in its graphic environment the complete management of these components of the interoperability solution, including the management of the different users and security.

5.3.5 Node Design Methodology

Regarding the validation of the correct operation of the interoperability solution and the correct design of the nodes. The first steps after the creation of nodes to access to the desired IoT services involve the design and implementation of interoperability

use cases. These use cases consist in a flow of execution to develop a new composite application with a specific purpose. To homogenize the process, it has been defined a procedure to integrate services and applications in the interoperability solution. If a developer follows these steps, the result is the creation of an accessible and totally functional node. As a summary of these procedure:

- Firstly, it is necessary to perform a complete analysis of the service, to obtain access to an instance of the platform with the service running, to perform test with data, to analyze the functionalities offered by the service, to study the provided methods to access to the service, to document the functionalities and to analyze the messages or actions that return the execution of each functionality of the service.
- In second place, the node has to be implemented: to group the functionality of the service, to identify the parameters needed to access the service, to create configuration nodes, to create the interface that collects the parameters that will consume the service, to develop the code that will execute the functionalities, and to define the messages that the node sends and receives.
- Then, the correct actuation of the node has to be tested considering real data, fixing the bugs and catching errors.
- Finally, the deployment of the service with real data and the characteristics of the node have to be documented.

5.3.6 Interoperability Use Cases

These nodes are used in the interoperability flows. They implement uses cases consisting of a flow of execution to perform a new composite application with a specific purpose. The flow and nodes involved are inside a instance of the interoperability solution deployed as a docker container.

There are implemented several use cases, so in order to give a practical approach to the information of this chapter, some of the most outstanding ones are going to be briefly described. For instance, considering a use case of a Port Environment, a CEP can be connected to trigger actions when the trucks monitored by different and heterogeneous road haulier companies platforms are physically close to a specific location, to perform actions in a platform from the port authority domain, like queries or to store historical information about this truck and show it in a dashboard. This implementation facilitates the improvement of logistic processes, shows alerts and allow to consider several application domains (road haulier companies, port and terminal) working together in a single composite application.

Another use case is related to Active Assisted Living Environment for accessing to historical information from different IoT platforms with heterogeneous formats, to perform syntactic and semantic translation in a common data model and to store data in a centralized database. All this heterogeneous information is stored in a common format and there exists the possibility of using different kinds of databases. Hence, external applications can directly access and use all these data from various sources

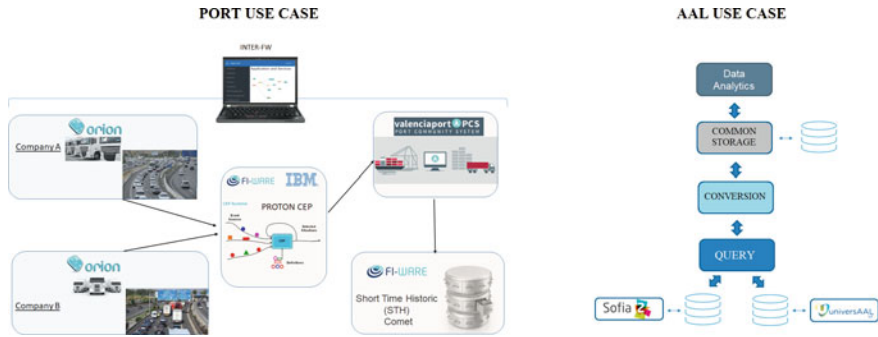


Fig. 15 AS2AS example of use cases

in their applications without making different connectors, providing extra value to this data (Fig. 15).

Finally, the following are some examples of generic use cases of this level: accessing to different services from different IoT platforms and use a dashboard to show the information in a mashup composition way, to use external services to store information in an application that provides an extra security layer, to connect and interoperate local IoT platforms services with services provided by main providers of web applications (mail, social networks, cloud services, etc.) or performing conversion and translation between services that use different formats or protocols.

5.3.7 Results and Discussion

Different solutions and tools have been developed that are available in the public GIT repository of the INTER-IoT project, together with the a guide that explains the technical information, how to deploy the components and how to develop new elements to extend the solution. In addition, the different components of Application and Services Interoperability solution are integrated inside the INTER-IoT Framework, which offers a visual interface to access and interact easily with the components in a way that integrates the common security and privacy functionalities of INTER-IoT. The main accessible components are:

- Instance Manager. It performs the deployment and management of the solution instances. It facilitates both extensibility and usability of the solution in a local or cloud deployment.
- Service registry. It offers the registration of new platforms and services in the interoperability solution.
- Flow registry. It provides access to the new services created and its information, the easy execution and the possibility of reuse the flows.

An API is provided to manage, deploy, access and modify different instances of the interoperability core solution without the use of v Framework. In addition, an

automatic Docker instantiation of the solution has been developed, since the use of containers allows a centralized management of different instances of the solution that can work concurrently in a scalable way and can be located in different servers.

As a result of this work there are available around 20 nodes and 10 interoperability flows available in the INTER-IoT git repository. They are related directly with elements of the INTER-IoT project. These nodes cover different aspects, like access to IoT Platform services or translation of formats of messages exchanged. All the nodes and flows available are compatible with the Node-RED solution and can be reused without effort.

6 Conclusions

This chapter describes the approach followed by INTER-IoT to solve the interoperability problem at each one of the IoT layers. These solutions have been proposed as an attempt to resolve one of the main remaining challenges that blocks the growth of IoT systems in real environments.

The solutions proposed by INTER-Layer are based in the creation of adaptors, gateways and higher level components, like the middleware, which provides a new layer of abstraction that allows the communication of the different components through it. Each one of the solutions can be implemented independently, following its own architecture aiming at solving a concrete interoperability problem in a specific layer. However, some of them are usually implemented altogether. For instance, the Device-to-Device solution and Network-to-Network solution can conform an IoT deployment solution for lower levels, although they could perfectly work separately. The only exception is the Middleware-to-Middleware solution, as it requires the IPSM component of the Semantics module to allow interoperability at Platform and Semantic levels as a whole.

A summary of the main and derived products extracted from each layer can be observed in the following table:

Layer	Core solution	Derived products
D2D	Physical and virtual gateway	Installer tool, automatic deployment tool, gateway extensions and SDK
N2N	SDN IoT controller	CLI, QoS application, dashboard and utilities
MW2MW	INTER-MW	IoT bridges, Docker-compose, examples and demo dashboard
AS2AS	INTER-AS	Instance manager, INTER-IoT services, flows and nodes

Moreover, each solution exposes a standardized API that facilitates the extensibility of the system and the creation of new application tools at the top of INTER-Layer.

An example of this is INTER-FW, a management utility used to monitor the status of the layered solutions and to create new instances of each one. This utility makes use of the API and is installed combined with one or more INTER-Layer solutions. More information about this tool is described in this chapter.

References

1. Gravina, R., Palau, C.E., Manso, M., Liotta, A., Fortino, G. (eds.): *Integration, Interconnection, and Interoperability of IoT Systems*. Internet of Things. Springer International Publishing (2018)
2. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: Towards multi-layer interoperability of heterogeneous IoT platforms: the inter-IoT approach. *Internet of Things* 199–232 (2018)
3. Fortino, G., Palau, C.E., Guerrieri, A., Cuppens, N., Cuppens, F., Chaouchi, H., Gabillon, A. (eds.): *Interoperability, Safety and Security in IoT—Third International Conference, InterIoT 2017, and Fourth International Conference, SaSeIoT 2017, Valencia, Spain, November 6–7, 2017, Proceedings*. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering, vol. 242. Springer (2018)
4. Fortino, G., Garro, A., Russo, W.: Achieving mobile agent systems interoperability through software layering. *Inf. Softw. Technol.* **50**(4), 322–341 (2008)
5. Vermesan, O., Friess, P. (eds.): *Digitising the Industry Internet of Things Connecting the Physical, Digital and Virtual Worlds*. Riverpublishers (2016)
6. Broring, A., Zappa, A., Vermesan, O., Främling, K., Zaslavsky, A., Gonzalez-Usach, R., Szmeja, P., Palau, C., Jacoby, M., Zarko, I.P., Sour-sos, S., Schmitt, C., Plociennik, M., Krco, S., Georgoulas, S., Larizgoitia, I., Gligoric, N., García-Castro, R., Serena, F., Orav, V.: *Advancing IoT Platform Interoperability*. River Publishers, The Netherlands (2018)
7. ITU-T. Y.2060: Overview of the Internet of things. Technical report, The International Telecommunication Union SG13 (2012)
8. Internet Engineering Task Force (IETF). RFC 7228: Terminology for Constrained-Node Networks (2014)
9. Aloï, G., Caliciuri, G., Fortino, G., Gravina, R., Pace, P., Russo, W., Savaglio, C.: Enabling IoT interoperability through opportunistic smartphone-based mobile gateways. *J. Netw. Comput. Appl.* **81**, 74–84 (2017)
10. Tschofenig, H., Arkko, J., Thaler, D., McPherson, D.: Architectural considerations in smart object networking. Technical report, Internet Architecture Board (2015)
11. Ramalho, F., Neto, A.: Virtualization at the network edge: a performance comparison. In: *WoW-MoM 2016—17th International Symposium on a World of Wireless, Mobile and Multimedia Networks*. Institute of Electrical and Electronics Engineers Inc., July 2016
12. Yacchirema, D.C., Esteve, M., Palau, C.E.: Design and implementation of a gateway for pervasive smart environments. In: *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*, pp. 4454–4459, October 2016
13. Zambrano, A., Perez, I., Palau, C., Esteve, M.: Quake detection system using smartphone-based wireless sensor network for early warning. In: *2014 IEEE International Conference on Pervasive Computing and Communication Workshops (PERCOM WORKSHOPS)*, pp. 297–302 (2014)
14. Zambrano, A., Perez, I., Palau, C., Esteve, M.: D3.2. Methods for Interoperability and Integration v.2. INTER-IoT H2020 project, October 2017. <https://inter-iot.eu/deliverables>
15. Varga, L.-O.: Multi-hop energy harvesting wireless sensor networks: routing and low duty-cycle link layer. Ph.D. thesis, Grenoble, December 2015
16. Vázquez, T.A., Barrachina-Muñoz, S., Bellalta, B., Bel, A.: HARE: supporting efficient uplink multi-hop communications in self-organizing LPWANs. *Sensors* **18**(2), 115 (2018)

17. Omnes, N., Bouillon, M., Fromentoux, G., Grand, O.L.: A programmable and virtualized network it infrastructure for the Internet of Things: how can NFV SDN help for facing the upcoming challenges. In: 2015 18th International Conference on Intelligence in Next Generation Networks, pp. 64–69 (2015)
18. Kreutz, D., Ramos, F.M.V., Veríssimo, P.E., Rothenberg, C.E., Azodolmolky, S., Uhlig, S.: Software-defined networking: a comprehensive survey. *Proc. IEEE* **103**(1), 14–76 (2015)
19. McKeown, N., Anderson, T., Balakrishnan, H., Parulkar, G., Peterson, L., Rexford, J., Shenker, S., Turner, J.: Openflow: enabling innovation in campus networks. *SIGCOMM Comput. Commun. Rev.* **38**(2), 69–74 (2008)
20. Internet Engineering Task Force (IETF). RFC 4627: The application/JSON Media Type for JavaScript Object Notation (JSON). Technical report, IETF Network Working Group (2006)
21. Zdravković, M., Trajanovic, M., Sarraipa, J., Jardim-Gonçalves, R., Lezoche, M., Aubry, A., Panetto, H.: Survey of Internet of Things platforms, February 2016
22. Partha Pratim Ray: A survey of IoT cloud platforms. *Future Comput. Inf. J.* **1**(1), 35–46 (2016)
23. Pileggi, S.F., Palau, C.E., Esteve, M.: Building semantic sensor web: knowledge and interoperability. In: Proceedings of the International Workshop on Semantic Sensor Web: SSW, (IC3K 2010), vol. 1, pp. 15–22. INSTICC, SciTePress (2010)
24. Giménez, P., Molina, B., Palau, C.E., Esteve, M.: SWE simulation and testing for the IoT. In: IEEE International Conference on Systems, Man, and Cybernetics, pp. 356–361 (2013)
25. Bröring, A., Schmid, S., Schindhelm, C.K., Khelil, A., Käbisch, S., Kramer, D., Le Phuoc, D., Mitic, J., Anicic, D., Teniente, E.: Enabling IoT ecosystems through platform interoperability. *IEEE Softw.* **34**(1), 54–61 (2017)
26. Lemos, A.L., Daniel, F., Benatallah, B.: Web service composition: a survey of techniques and tools. *ACM Comput. Surv.* **48**(3) (2015)
27. Guth, J., Breitenbücher, U., Falkenthal, M., Leymann, F., Reinfurt, L.: Comparison of IoT platform architectures: a field study based on a reference architecture. In: 2016 Cloudification of the Internet of Things (CIoT), pp. 1–6 (2016)
28. Belsa, A., Sarabia-Jacome, D., Palau, C.E., Esteve, M.: Flow-based programming interoperability solution for IoT platform applications. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 304–309 (2018)
29. Blackstock, M., Lea, R.: Toward a distributed data flow platform for the web of things (distributed node-red), pp. 34–39, October 2014
30. Blackstock, M., Lea, R.: Fred: a hosted data flow platform for the IoT, pp. 1–5, December 2016
31. Kleinfeld, R., Steglich, S., Radziwonowicz, L., Doukas, C.: glue.things: a mashup platform for wiring the Internet of Things with the Internet of Services, October 2014

Semantic Interoperability



Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski,
Bartłomiej Solarz-Niesłuchowski, Paweł Szmeja, and Katarzyna Wasielewska

Abstract Interoperability, on the semantic level, deals with shared understanding of data, between IoT artifacts. Positioned on top of the *syntactic* layer, semantic interoperability facilitates solutions to problems that arise after the data is in a common format, with syntax understood by all participants. Provisioning of compatibility between not directly compatible data structures, representations, conventions and standards, falls strictly under the responsibility of semantic methods. This chapter introduces the INTER-IoT perspective on data semantics and summarizes its achievements in the field of semantic interoperability. Being a generic and far-reaching solution, the syntactic compatibility challenge is also mentioned, as it is a necessary precondition to comprehensive data interoperability suite.

1 Introduction

While the outlook for the Internet of Things (IoT) remains positive, its underlying vision, of an all-encompassing ecosystem, remains unfulfilled. The pervasiveness of so-called *silos* is, in no small part, due to differences in requirements on data semantics introduced, independently, by joining systems. Whether caused by varying

M. Ganzha (✉) · M. Paprzycki · B. Solarz-Niesłuchowski · P. Szmeja · K. Wasielewska
Systems Research Institute Polish Academy of Science, Warsaw, Poland
e-mail: maria.ganzha@ibspan.waw.pl

M. Paprzycki
e-mail: marcin.paprzycki@ibspan.waw.pl

B. Solarz-Niesłuchowski
e-mail: bartlomiej.solarz@ibspan.waw.pl

P. Szmeja
e-mail: pawel.szmeja@ibspan.waw.pl

K. Wasielewska
e-mail: katarzyna.wasielewska@ibspan.waw.pl

W. Pawłowski
Faculty of Mathematics, Physics and Informatics, University of Gdańsk, Gdańsk, Poland
e-mail: wieslaw.pawlowski@ug.edu.pl

© Springer Nature Switzerland AG 2021

C. E. Palau (eds.), *Interoperability of Heterogeneous IoT Platforms*, Internet of Things,
https://doi.org/10.1007/978-3-030-82446-4_5

domain requirements and perspectives, legacy software, lack of foresight, or deliberate vendor-locking, IoT artifacts usually “speak different languages”, understood exclusively “within their cliques”. When large(r)-scale IoT ecosystems are developed, different silos that are (physically and/or virtually) close to each other, often require meaningful communication. Here, semantic interoperability layer offers possibility of shared understanding of data. Application of a semantic solution allows implementation of valid, purposeful, and useful conversation, while protecting internal use of varying data syntax, models, and semantics [52].

Here, note that it is very rare that *explicit* semantics, using formal representation of the domain, in the form of an *ontology*, is available. Often, the meaning of data is *implicit* and, for instance, buried in system documentation. Fortunately, even in such case it is possible to “make implicit semantics explicit” (i.e. establish an ontology that represents it) and instantiate appropriate translators between the internal data model and its ontology-based, semantically-enriched counterpart. Readers interested in how this preliminary step towards semantic interoperability can be performed are welcome to consult [37, 39], and references provided there.

In the layered model of interoperability [24] semantics is positioned above *syntactic interoperability* [39], which ensures common understanding of the *structure* of data. Since the objective of this chapter is to present the INTER-IoT approach to *semantic* interoperability, and because syntactic interoperability is its prerequisite, in what follows, we shall assume that: (i) (one way or another) IoT artifacts have explicit semantics, represented as an ontology; (ii) artifacts participating in the ecosystem, have already achieved the syntactic interoperability [53].

1.1 Towards Semantic Interoperability

To provide the context for this chapter, let us assume that a number of IoT artifacts has to be “associated” to instantiate a new IoT ecosystem. Naturally, the case when an existing IoT ecosystem is to be enriched, by adding new artifacts (e.g. sensors/platforms/services) to deliver new functionalities, is also covered.

In general, there are multiple ways of achieving semantic interoperability within the ecosystem. The simplest is to implement the same semantics (use the same ontology) across the ecosystem. However, it would mean that each artifact would have to comply to the common ontology, and possibly change its preexisting semantics to fit the unified vocabulary and schema. In a structure, where no party is privileged, and no one can enforce such far-reaching changes, such approach is not feasible [49].

Semi-interoperability may be achieved when many artifacts write data to the same storage that has a specific semantics. In this case, they all need to translate the data one-way, on their own, from their own semantics into the selected one. This approach, however, is mainly applicable to data-fusion and does not support inter-artifact communication. Furthermore, the question “which ontology should be selected for the common repository” remains unanswered.

Partial interoperability is (at least theoretically) achievable when artifacts share an upper ontology as a core “module” of their ontologies. This level of interoper-

ability, however, is hardly applicable to IoT ecosystems, mainly because of their vast (internal) heterogeneity. Simply said, in order to be fully functional, highly specialized applications in IoT require higher level of interoperability than what, in most realistic cases, an upper ontology can offer. Here, it should be also noted that the heterogeneity of semantic representations occurring in IoT-related domains (see, for instance, [35]) further diminishes real-world applicability of this approach.

Even though, it is not strictly necessary from the technical point of view, the INTER-IoT approach assumes that the joining process should be “non-invasive” for the participating artifacts. In particular, it means that no *internal* adjustments should be required in the process of forming/joining the ecosystem. To achieve this goal, INTER-IoT uses *communication channels* which facilitate flow of information between artifacts. Since the artifacts may “speak different languages” it is natural to expect that the communication channels not only form communication “media” but perform *semantic translation* as well.

1.2 IoT Case Study

To illustrate the proposed approach, let us consider a logistic and delivery system, operating in an IoT-enabled city (see, Fig. 1). There are several delivery companies (*iotDelivery*), operating in the area, that deliver/pick up goods to/from specific locations. Their goal is to dynamically optimize routes and operation time of their trucks. Therefore, each monitors truck positions and routes and chooses a specific vehicle for selected job. The choice is based on: (i) current traffic information, (ii) availability of parking slots near the delivery/pick-up point. Traffic information is provided by *dTraffic* company. This company analyses and publishes information gathered by drones monitoring city traffic. Note that drones can come from different vendors and, consequently, support different communication standards. Each drone should, periodically, send two types of messages, containing: (i) its position and battery level, and (2) traffic congestion information. *dTraffic* gathers and processes this information, to publish traffic reports. Obviously, *dTraffic* needs to send messages to control drones,

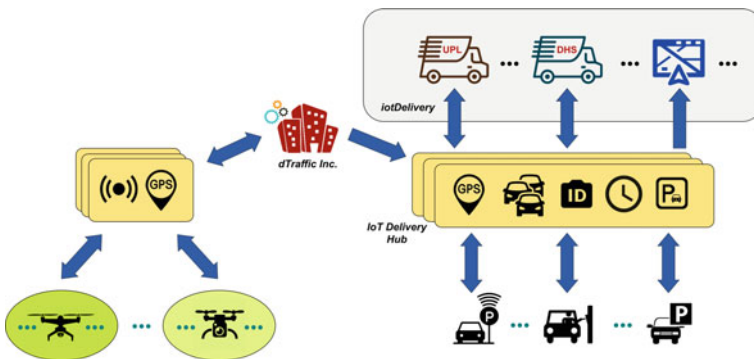


Fig. 1 Smart delivery and logistics interoperability scenario

e.g. request to return to base for charging (including coordinates of selected base station). Since platform used by *dTraffic* has to operate 24/7, recycling/connecting new devices must be seamless. Finally, parking lots companies (*iParking*) manage parking spaces in specific regions. *iParking*'s publish information about availability of free slots, and receive reservation requests from *iotDelivery* companies. Reservation messages contain, car ID/RFID and arrival time. Upon truck arrival, *iParking* provides slot number. Note that each *iParking* instance can be managed by different entities and use different communication standards.

Here, interoperability is needed for: (i) *dTraffic* company communicating with its drones—drones should be able to send messages in their native format, while *dTraffic* should communicate with them in a common selected format, not having to directly use/support all vendor-specific standards; (ii) *iotDelivery* cooperating with different trucks—should have simplified communication, as in the case of *dTraffic*; (iii) *iotDelivery* gathering information from *dTraffic*, trucks, *iParking*'s, and sending reservations to *iParking*'s. In first two cases, the interoperability mechanism should seamlessly handle addition of new drone/truck vendors. The last example requires creation of an *IoT Delivery Hub*, where information between parking lots, trucks, delivery companies and traffic monitoring can be exchanged and understood.

The *IoT Delivery Hub* is a place where information is exchanged between stakeholders: trucks, *iParking* platforms, *iotDelivery* platforms, and *dTraffic* platform. Interoperability should enable any *iotDelivery* platform to receive information from different parking lots with parking spaces availability in a way that is independent from the original semantics that these platforms use internally. Moreover, when *iot-Delivery* platform wants to make a reservation at a parking lot it should be able to send it in its natively supported semantics. Additionally, *IoT Delivery Hub* consumes data from *dTraffic* platform with current city congestion status, and translates data exchanged between trucks and *iotDelivery* platforms. Note that real-life scenario can be even more complex because trucks may cooperate with different delivery companies.

When designing the shared data model, in the *IoT Delivery Hub*, the following concepts should be taken into consideration:

- *Geolocation*—for expressing trucks location, parking lot and parking place location, and to identify areas with different traffic levels,
- *Time*—for defining reservation time slots,
- *Traffic*—for describing traffic congestion level in a given area,
- *Logistics*—for trucks identification, description of delivery orders,
- *Parking*—for describing parking availability.

Clearly, most of data publishing/exchange happening within the use-case, has a *streaming nature*. Therefore, the interoperability solution should enable translation of *streams* of messages exchanged between artifacts as well.

With the scenario in mind, we proceed as follows. First, we outline the semantic interoperability problem, and possible approaches towards solving it. In Sect. 2, we consider standards and ontologies that are available on the market, including the domain of our example scenario. Next, in Sect. 4, we discuss how to approach semantic translation, and what is the role of ontology *alignments* in the process. Then,

we describe a specific format for persisting mappings/alignments between ontologies, and discuss the Inter-Platform Semantic Mediator (IPSM) – tool for performing alignment-based semantic translation. Finally, in Sect. 7, we return to the example scenario, to describe its realization using INTER-IoT approach.

2 Ontologies in the Internet of Things

An *ontology* [45], in a broad sense, is a way of representing and describing knowledge, and in more “applied” terms—is a way of representing data together with *metadata*. It can contain information about both concrete instances of data (individuals) and structural information about data, usually confined to a domain of interest. In this chapter, systems are *semantically interoperable* if a sent message can be (in a practical, not theoretical, way) expressed in terms of the ontology of the receiving system.

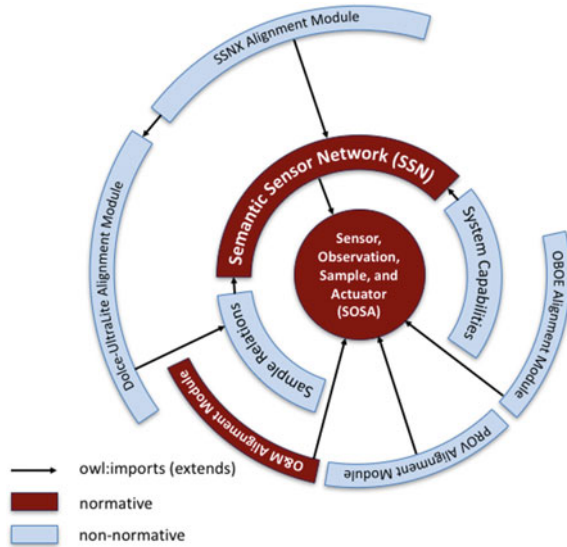
A formal way of expressing semantics is the Web Ontology Language (OWL).¹ OWL ontologies are often modular. Here, *Horizontal* modules are subsets of ontologies, often very loosely (or not at all) connected. They describe different areas of the domain of interest, that reside on roughly the same level of abstraction. For instance, in the scope of logistics, a geolocation module is independent from a module describing the cargo. While both modules will likely be used to describe an instance of a physical container, there is no formal connection between the location of a container and its type, weight, or color. Moreover, a geolocation module is not tied to transportation or logistics, as it may be used for other physical entities, and even in entirely different domains. In summary, horizontal modules are not dependent on each other, separable, and sometimes completely orthogonal.

Vertical modules, on the other hand, build “on top of each other” and form a strict top-down hierarchy. The “level of a module” corresponds to its relative level of abstraction. Here, most general ontologies are “on top”, and most specific ones “at the bottom”. Overall, vertical modularity, based on levels of abstraction, is a backbone of semantic interoperability. High-level (“top”) ontologies (or modules) contain general terms that are being “specified” by lower ontologies. An example, central to the IoT, are “device” ontologies that describe, in general terms, IoT devices (sensors, actuators, smart and mobile devices, etc.). These ontologies are then extended by domain-specific (or application-specific) ontologies. In the eHealth domain, for instance, lower ontologies may add types of specific devices, e.g. patient monitoring tools, automated medicine applicators, etc. Transportation devices, on the other hand, may include GPS or speed sensors, port crane actuators, etc. A canonical model of semantic interoperability assumes that the same high-level device ontologies are used in both cases. Hence, without additional effort, there is a set of terms understood across domains. For a more detailed explanation of ontology modularity, see [47].

To bring semantic methods to IoT, availability mature/standardized, IoT-specific and domain-specific, ontologies is crucial. Here, note that many ontologies were developed within research projects and remain as prototypes, often incomplete, or abandoned (upon project end). We will thus briefly describe the notable exceptions.

¹ <https://www.w3.org/TR/owl2-overview/>.

Fig. 2 SOSA/SSN modular structure (source W3C)



2.1 IoT Core Ontologies

Proper semantic treatment of sensors, sensor networks, actuators, and their operations, i.e. observations and actuations, is of fundamental importance for the IoT applications. Many ontologies for describing these concepts/entities have been proposed. Some of them evolved over time to accommodate changing scope, target audience, and technological landscape. Usually, for obvious reasons, the IoT-dedicated ontologies are also combined with/utilize other, high-level ontologies (ontological “modules”), such as ontologies of geolocation (e.g., LinkedGeoData [26], GeoSPARQL [9], or WGS84 [1]), units of measure (e.g., QU,² OM,³ or SWEET units⁴), time (e.g., Time OWL⁵), or provenance (e.g. PROV-O [18]).

The W3C SSN [25, 29, 51] is an ontology or, actually, a suite of ontologies for describing sensors, their accuracy and capabilities, observations, methods used for sensing, and sensor deployment. The original version of W3C SSN turned out too “heavy-weight” for many smart devices, and did not cover concepts, which became important over time, such as actuators and actuation, or samples, samplers, and sampling. Therefore, recently, a new version of the SSN ontology was proposed. The W3C SOSA/SSN⁶ is a modular ontology (see Fig. 2), providing the required extensions. After the redesign, (the new version of) SSN became an extension of the kernel module, called SOSA (*Sensor, Observation, Sample, Actuator*).

² <https://www.w3.org/2005/Incubator/ssn/ssnx/qu/>.

³ <https://github.com/HajoRijgersberg/OM>.

⁴ <https://github.com/ESIPFed/sweet>.

⁵ <https://www.w3.org/TR/owl-time/>.

⁶ <https://www.w3.org/TR/vocab-ssn/>.

IoT-Lite [28], is a light-weight ontology for representing IoT specific concepts. It specializes the SSN Device concept, and adds representation for objects, services, and actuators. It deal with geolocation, by using WGS84 ontology. The IoT-Lite focus on key IoT concepts directly supports semantic interoperability between IoT platforms.

Another notable ontology for IoT is the ETSI SAREF.⁷ It models the domain of smart appliances and household devices, although there are many extensions that bring this ontology into the domains of energy, environment, and many others.

Many IoT platforms and middleware software solutions are also “riding the semantic wave” and introducing support for ontologies by either (i) adopting existing standards (e.g. FIWARE support for schema.org ontologies in its data models⁸), or (ii) authoring endemic ontologies (like oneM2M Base Ontology⁹). Readers interested in an overview and a more in-depth discussion of the IoT related ontologies can consult [36].

2.2 Domain-Specific Ontologies

Virtually every IoT-based application/system needs to consider domain-specific ontologies. They accompany IoT ontologies to form a complete view of modeled and exchanged information. We will now briefly discuss some of them, representing several application domains. First, let us focus on transportation and logistics, as related to our sample use case. It was also central to one of pilot deployments of INTER-IoT. Here, ontologies span business perspectives of freight and production companies, transportation hubs (e.g. airports, train stations, ports), mass transit, transport infrastructure, personal and business travel, etc. Interestingly, in logistics in particular, many ontologies cover specific (and narrow) areas and very rarely describe a broad view of the domain. Furthermore, in many cases, work on logistics ontologies ended at the design phase (see, [46, 48]).

Here, one of the most-known ontologies is OTN (Ontology of Transportation Networks; [17, 42])—a top-level ontology, produced in the REWERSE [20] project. It models general concepts of transportation, traffic networks and locomotion, as well as describes aspects of transportation relevant to, for instance, smart city or smart highway systems. The OTN is a realization and extension of the GDF [8]—Geographic Data Format (ISO specification)—as a formal OWL ontology. It was used in [43], to facilitate ontology-driven interoperability between urban models.

Let us also mention a Logistic Grid Ontology [15] that represents a service-oriented approach to logistics, realizing the idea of combining semantic technologies and cloud services, in order to enable semantically-driven description and application of logistic processes (see, [44]). It was developed within the LOGICAL project [11], aim of which was to “enhance the interoperability of logistics businesses.” A cloud service [12] utilizing the Logistic Grid Ontology was one of the results of this project.

⁷ <http://ontology.tno.nl/saref/>.

⁸ <https://fiware-datamodels.readthedocs.io/en/latest/guidelines/>.

⁹ <http://www.onem2m.org/technical/onem2m-ontologies>.

Finally, LogiCO (and its extension LogiServ) ontologies [13, 14] contain core concepts and properties from the domain. They model business activities (such as Transport, Transshipment, Load, Discharge, Storage, Consolidation, Deconsolidation, etc.) and their properties as the basis for specifying logistic services.

There are also ontologies, which cover other concepts related to transportation and logistics. The *Transport Disruption Ontology* [22], for example, is devoted to events, which can have a disruptive impact on travel planning. It is based on analysis of published disruption information, described in the DATEX II [4] specification.

We will now give a brief overview of other domains with knowledge modeled as ontologies. Some of them are related to INTER-IoT pilot use cases e.g. geospatial data, whereas some were used in INTER-IoT open call projects e.g. weather data.

In the medical domain, biological and biomedical ontologies are within the OBO Foundry [16] and the BioPortal [2]. These are mostly very specialized ontologies or vocabularies, e.g. ICD10, ICD11, SNOMED CT. Standardization bodies, such as HL7, support ontological representation of their standards, e.g. FHIR (Fast Healthcare Interoperability Resources) [6]. In INTER-IoT mHealth pilot we used a set of UniversAAL ontologies [23] as domain ontologies (with extensions). They were originally designed for UniversAAL platform and allow to describe measurements.

Ontologies from weather and meteorology domain include: (i) Linked Earth Ontology [10] that provides a common vocabulary for annotating paleoclimatology data; (ii) SEAS WeatherOntology [21] that allows to describe weather conditions, e.g. temperature, weather, sunrise, sunset; (iii) Climate and Forecast (CP) features [3] that offers representation of generic features defined by Climate and Forecast (CF) standard names vocabulary.

In IoT deployments, concepts related to geolocation and positioning are frequently required. Here, the most advanced ontology and geographic query language seems to be GeoSPARQL [9]. They allow to describe complex geospatial objects and query them using a set of built in functions. Other, simpler but still suitable for many use cases ontologies include W3C Basic Geo Vocabulary [1] and GeorSS.¹⁰

Overall, ontologies have been designed for various domains—either as simple vocabularies, or as more expressive conceptual models. The decision, which one to use should be based on the requirements analysis and detailed information concerning all data models that are involved. However, it is clear that, for interoperability, abundance of ontology choices may also turn out to be a challenge rather than help.

INTER-IoT, as a generic interoperability solution for IoT, in its scope, covers IoT entities that go far beyond a typical IoT platform, or service. Although, from a semantic perspective, support for sensors and actuators, deployments, physical locations, simple measurements, etc., is common, it is very rare that a given platform considers a broader ecosystem. Hence, the interoperability problem necessarily includes not just multitude of devices, but also multiple platforms, services, middlewares, users, etc. Each of them needs to be treated with the same attention as devices. Hence, we have developed the *GOIoTP* ontology, described in the next section.

¹⁰ <http://www.georss.org/>.

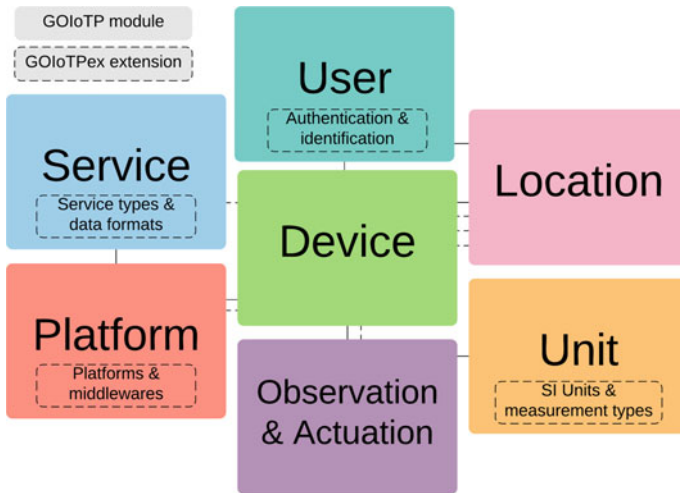


Fig. 3 GOIoT ontology—main modules

3 Generic Ontology for IoT Platforms

Among the semantic products of INTER-IoT one can find two closely related ontologies: (i) Generic Ontology for IoT Platforms (*GOIoT*),¹¹ and its extension (ii) *GOIoTPex*.¹² The *GOIoT* is a modular core ontology for general use in IoT projects. It offers an expanded perspective on IoT, including both top-level concepts related to platforms and users, as well as “devices” that the ecosystem consists of. Therefore, *GOIoT*, provides common and consistent shared semantics (see, Sects. 1.1 and 2), as well as typical core IoT ontology (Sect. 2.1). *GOIoT* was built around SOSA/SSN and offers seven horizontal modules (with additional provenance extension points), each focused on a distinct area of an IoT ecosystem (see, Fig. 3). Other notable imports include GeoSPARQL and NASA SWEET units ontologies.

Along a rich *Device* module (as in other IoT ontologies), *GOIoT*, following SOSA/SSN, proposes a separate module for observations and actuations. This decoupling allows better separation of physical or virtual devices, described by selected properties, and device operation (including obtained or produced data).

Module dedicated to *Platforms* goes beyond the usual scope of platform-specific IoT ontologies, and treats a platform as an entity in an ecosystem, in which it needs to cooperate with other, equally important, platforms. This is critical for interoperability scenarios, in which a platform can join or leave a federation/group, delegate the management of some of its roaming devices to other platforms, while no artifact is in a privileged position. Note that the importance of this approach will increase as new IoT ecosystems will federate those already deployed using existing platforms.

¹¹ <http://inter-iot.eu/GOIoT#>.

¹² <http://inter-iot.eu/GOIoTPex#>.

The *User* and *Service* modules are largely independent from others. Whereas, usually in IoT ontologies, a service is attached to a platform, *GOIoTP* opts for decoupling them. This allows treatment of services as separate entities, that may, or may not be offered by some platform, or device. Similarly, a user in *GOIoTP* is a separate “sovereign” that may be dynamically associated with a service, device, platform, location, etc. Specifically, user is treated as a physical (e.g. a human) or virtual agent that may have some presence and history with an IoT artifact (e.g. an account and a set of roles registered with a platform). The user description is independent from technical details or requirements of any platform, device, or service.

In addition to generic descriptions of physical locations and their interconnections available in the *Location* module, the geographic information can be annotated with GeoSPARQL descriptions. In this way *GOIoTP* supports detailed descriptions of areas, and geographical functions, such as area intersections, collisions, point inclusion, distance calculation, etc.

In accordance with the ontology engineering best practices, *GOIoTP* has large number of generic descriptions, also called “stubs” that are lightweight, and are designed to be extended with more concrete definitions.

GOIoTPex is an official extension of *GOIoTP* that proposes a number of realizations and concretizations for top-level abstractions, defined in modules vertical with respect to *GOIoTP*, and horizontal to each other. For instance, where *GOIoTP*, across 2 modules, defines general framework for units of measurement, and connects it to observations and actuations, *GOIoTPex* complements it with instances and classes related to SI units. By the virtue of vertical modularization, this has no bearing on *GOIoTP*, which remains compatible with the imperial unit system. In this way, *GOIoTPex* brings semantics closer to concrete implementations of individual IoT systems, while leaving the option to fall-back to the more general *GOIoTP*, and preserving the vertical and horizontal modularity. This is important both for semantic interoperability, and for practicality of use. Both ontologies were successfully used in INTER-IoT deployments as Central Ontologies (see, Sect. 6).

For more information about history and design of *GOIoTP*, please refer to INTER-IoT deliverables (specifically D4.2 and D4.1). Ontology files, detailed axiom descriptions, outline of modules, updates and more can be found online.¹³

4 From Alignments to Translation

Since the conceptualization of a domain directly corresponds to its *ontology* (see, [40]), to bridge the “semantic gap” between the artifacts, we have decided to use *alignments* between the corresponding ontologies. An *ontology alignment* [30] is a set of correspondences between two or more ontologies. These correspondences may be simple (between atomic entities) or complex (between groups of entities and sub-structures), but always relate entities from different ontologies and express their

¹³ <https://inter-iot.github.io/ontology/>.

similarity. Here, *semantic translation* is a process of changing the underlying semantics of a piece of knowledge. Given information described semantically, in terms of a source ontology, it is transformed into information interpretable (understandable) in the scope of target semantics.

We assume that new instances of data may be generated dynamically within an ecosystem. Therefore, we aim to utilize alignments between structural information, rather than individuals, and assume that all exchanged messages are ontologically demarcated and schema-compliant. In the translation process, appropriate alignments are “applied” to messages traveling through *communication channels*. Specifically, application of an alignment substitutes parts of information from incoming messages with their translations (parts mapped by the alignment). Such, updated/translated, messages are available to recipient(s) “at the end of the channel”. Here, let us assume that two (or more) platforms operate in the same domain, e.g. logistics, but use different semantics. For instance, one platform uses term *truck* with an attribute *capacity*, while the other uses term *lorry* with an attribute *volume*. Obviously, these terms have the same meaning and can be treated as equivalent. This exemplifies the simplest case of an alignment—equivalence between two atomic terms. Observe also that capacity and volume may be represented in different units, e.g. one uses the metric system (cubic meters), while the other comes from US and uses cubic inches, which introduces a complication into the, otherwise simple, example.

The simplest way in which ontology alignment can be found/defined is to print ontologies, place them next to each other and establish how their concepts relate. However, potential size and complexity of ontologies immediately undermines this approach. In [38], we have summarized the state-of-the-art in the area of tools for ontology alignment/merging/translating, etc. We have identified several tools that can support semantic engineers in preparing alignments. The most interesting among them were LogMap [41],¹⁴ COMA [27]¹⁵ and Agreement Maker [31].¹⁶

Let us now consider the translation process. In principle, a *semantic-enabled* communication channel is a medium that, when properly configured, can accept messages with data annotated by entities from one (input/source) ontology and produce semantically equivalent messages annotated with another (output/target) ontology. In this way, the translation is entirely *external* to the participating artifacts. This simple idea (see, Fig. 4) turns out to be fairly complex to realize in practice, and requires harmonization of both syntax and semantics of information, besides actually using the communication infrastructure. Note that the actual number and topology of communication channels depends on the information flow in the ecosystem, and can change dynamically over time (in response to the needs of the applications using them). Additional challenge is to ensure that the communication architecture is capable of handling the volume of messages that can be exchanged between IoT artifacts. We shall come back to this problem in Sect. 6, but first, we need to describe a format that INTER-IoT solution uses for expressing alignments.

¹⁴ <https://www.cs.ox.ac.uk/isg/tools/LogMap/>.

¹⁵ <http://dbs.uni-leipzig.de/Research/coma.html>.

¹⁶ <http://somer.fc.ul.pt/aml.php>.

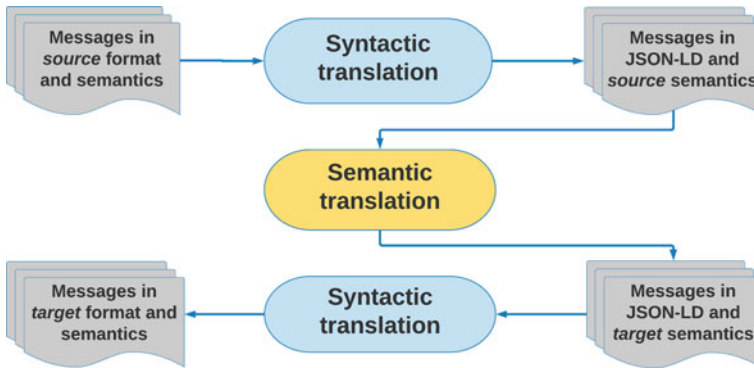


Fig. 4 INTER-IoT translation

5 Alignment Format

As a result of an in-depth analysis, we have decided to use the *Alignment Format* [7] as the basis for alignment representation. We have inspected several other methods of representing alignments, including EDOAL [5], but none of them turned out to be fully satisfactory for our purposes. The Alignment Format is a language independent format dedicated to persisting alignments in a simple and readable way. It was designed with the goal to provide common output format for matching tools. However, we have found that the Alignment Format itself, while working very well in ontology matching competitions, is not going to work well for streaming semantic translations. Therefore, taking the Alignment Format as the base, we have developed a dedicated format, with an RDF [19] representation (in RDF/XML serialization). The *IPSM Alignment Format* (IPSM-AF, in short) is used in the Inter-Platform Semantic Mediator (IPSM) software component (see Sect. 6). The IPSM-AF allows us to express mappings between contents of any valid RDF graphs. At the same time, alignments persisted in the original Alignment API format can be easily translated to the IPSM-AF, because of similarity in their structures.

Listing 1 shows structure of IPSM-AF expressed in RDF/XML. The format allows persisting uni-directional correspondences between ontologies. Mappings are split into separate alignment *cells*. Each cell is a correspondence between two *entities* (or compound entity description). Entity can be a class, instance, object or datatype property. Namespace prefixes used in the examples are expanded in Table 1.

Elements from the *align* namespace are inherited from the Alignment Format. Table 2 lists the elements elements that are used to persist basic metadata about the alignment. The *stripas:cellFormat* property allows to specify data format used in alignment cells. Currently, RDF/XML and Turtle¹⁷ are supported.

¹⁷ <https://www.w3.org/TR/turtle/>.

Table 1 Namespaces in IPSM-AF

Suggested prefix	Namespace
sripas	http://www.inter-iot.eu/sripas#
var	http://www.inter-iot.eu/sripas:node_
pred	http://www.inter-iot.eu/sripas:pred_
align	http://knowledgeweb.semanticweb.org/heterogeneity/alignment#
dcelem	http://purl.org/dc/elements/1.1/
exmo	http://exmo.inrialpes.fr/align/ext/1.0/#

Table 2 Metadata elements in IPSM-AF

Element/Attribute	Meaning
dcelem:title	Name of the alignment
exmo:version	Version of the alignment e.g. 1.0
dcelem:creator	Author of the alignment
dcelem:description	Comment on what is the scope/aim of the alignment
align:xml	Derived from Alignment API. Default value: yes
align:level	Derived from Alignment API. Default value: 2IPSM
align:time	Derived from Alignment API
align:method	Derived from Alignment API. Default value: manual
align:type	Derived from Alignment API. Default value: **

```
<?xml version='1.0' encoding='utf-8' standalone='no'?>
<rdf:RDF xmlns="http://www.inter-iot.eu/sripas#" % other xml
  namespaces% >
<align:Alignment>
  <dcelem:title> % alignment title % </dcelem:title>
  <exmo:version> % alignment version % </exmo:version>
  <dcelem:creator> % alignment creator % </dcelem:creator>
  <dcelem:description> % alignemnt description % </dcelem:
    description>
  <align:xml>yes</align:xml>
  <align:level>2IPSM</align:level>
  <align:type>**</align:type>
  <align:method> % method % </align:method>
  <align:time> % time % </align:time>
  <sripas:cellFormat>
    <iiot>DataFormat rdf:about="http://inter-iot.eu/sripas
      #rdfxml"/>
```

```

</sripas:cellFormat>
<align:onto1>
  <align:Ontology rdf:about="% source ontology URI %">
    <align:location> % source ontology location % </
      align:location>
    <align:formalism>
      <align:Formalism
        align:name="% source ontology formalism
          name %"
        align:uri="% source ontology formalism URI
          %"/>
      </align:formalism>
    </align:Ontology>
  </align:onto1>
<align:onto2> % target ontology information % </align:
  onto2>
<sripas:steps rdf:parseType="Literal">
  <sripas:step sripas:order="% cell order %" sripas:cell
    ="% cell id %"/>
  % more steps %
</sripas:steps>
<align:map> % alignment cell 1 % </align:map>
% ... %
<align:map> % alignment cell N % </align:map>
</align:Alignment> </rdf:RDF>

```

Listing 1 IPSM-AF structure

In principle, an IPSM-AF file (Listing 1) describes a uni-directional alignment comprised of independent mapping cells, each having “input” and “output” entity descriptions. Elements *onto1* and *onto2* describe the “source” and “target” ontologies of the alignment, by giving their URIs and specifying formalism used for their definition (e.g., OWL 2.0). Here, let us note that when bi-directional translations are needed, separate alignments have to be defined (even if the alignment cells all describe equivalences, and are, therefore, trivially reversible). This is because, in the IPSM-AF, source and target ontologies are explicitly specified (information used for communication channel configuration/creation process, within the IPSM).

The *steps* element specifies the (default) *order*, in which *cells* of the alignment will be subsequently *applied* in the message transformation process. Each *step* refers to a cell identifier as given by the *id* attribute of the *Cell* element. Note that a given cell might be referenced here (hence also applied) more than once. The default order may also be overridden during the channel configuration process.

Every *Cell* element represents an “atomic” RDF graph transformation. Here, content of *entity1* describes the *source*, and content of *entity2* establishes the *target* of the transformation. Both should be valid RDF graphs, possibly containing special-purpose nodes, playing the role of “variables”, which are to be bound and referenced within the transformation.

Listing 2 shows an example (unidirectional) alignment between a drone and an ontology used by *dTraffic*. In the sample scenario data coming from drones is integrated and processed by the platform. We also assume that different types of drones use different communication standards. Here, original data is expressed in SAREF ontology. The target ontology is based on SOSA/SSN with extensions. This alignment includes two cells—one defining rules for translation of traffic message from drone to platform, and the other for translation of battery level and location message.

```
<align:Alignment>

% ... alignment metadata ... %

<sripas:steps rdf:parseType="Collection">
  <sripas:step sripas:order="1" sripas:cell="http
    ://www.inter-iot.eu/sripas#1_traffic"/>
  <sripas:step sripas:order="2" sripas:cell="http
    ://www.inter-iot.eu/sripas#2_status"/>
</sripas:steps>

<align:map>
  <align:Cell rdf:about="http://www.inter-iot.eu/
    sripas#1_traffic">
    <align:entity1 rdf:datatype="http://www.w3.
      org/2001/XMLSchema#string">
      var:Device a saref:Device ;
      saref:makesMeasurement var:Meas, var:Pos1
        , var:Pos2 .

      var:Meas a saref:Measurement ;
      saref:hasValue var:Val ;
      saref:hasTimestamp var:Tsp ;
      saref:isMeasuredIn sarefInst:Frequency ;
      saref:relatesToProperty sarefInst:Traffic
        .

      var:Pos1 a saref:Measurement ;
      saref:hasValue var:StartPos ;
      saref:relatesToProperty sarefInst:
        StartPosition .

      var:Pos2 a saref:Measurement ;
      saref:hasValue var:EndPos ;
      saref:relatesToProperty sarefInst:
        EndPosition .
    </align:entity1>
```

```

<align:entity2 rdf:datatype="http://www.w3.
  org/2001/XMLSchema#string">
  var:Device a iiot:IoTDevice, sosa:Sensor
    ;
  sosa:madeObservation var:Meas, var:Pos1,
    var:Pos2 .

  var:Meas a sosa:Observation ;
  sosa:hasResult [ a sosa:Result ;
    iiot:hasResultValue var:Val ] ;
  sosa:phenomenonTime [ a time:Instant ;
    time:inXSDDateTime var:Tsp ] ;
  sosa:observedProperty iiotex:Traffic .

  var:Pos1 a sosa:Observation ;
  sosa:hasResult [ a sosa:Result, geosparql
    :Geometry ;
    iiot:hasResultValue var:GeoStart ] ;
  sosa:observedProperty iiotex:
    StartPosition .

  var:Pos2 a sosa:Observation ;
  sosa:hasResult [ a sosa:Result, geosparql
    :Geometry ;
    iiot:hasResultValue var:GeoEnd ] ;
  sosa:observedProperty iiotex:EndPosition
    .
</align:entity2>
<align:relation>=</align:relation>
<sripas:transformation rdf:parseType="
  Literal">
  ...
</sripas:transformation>
<sripas:filters rdf:parseType="Literal">
  ...
</sripas:filters>
<sripas:typings rdf:parseType="Literal">
  ...
</sripas:typings>
</align:Cell>
</align:map>

<align:map>
  <align:Cell rdf:about="http://www.inter-iot.eu/
    sripas#2_status">

```

```

<align:entity1 rdf:datatype="http://www.w3.
  org/2001/XMLSchema#string">
  var:Device a saref:Device ;
  saref:makesMeasurement var:Meas, var:Pos
  .

  var:Meas a saref:Measurement ;
  saref:hasValue var:Val ;
  saref:hasTimestamp var:Tsp ;
  saref:isMeasuredIn sarefInst:Percentage ;
  saref:relatesToProperty sarefInst:
    BatteryLevel .

  var:Pos a saref:Measurement ;
  saref:hasValue var:Position ;
  saref:relatesToProperty sarefInst:
    Position .
</align:entity1>
<align:entity2 rdf:datatype="http://www.w3.
  org/2001/XMLSchema#string">
  var:Device a iiot:IoTDevice, sosa:Sensor
  ;
  sosa:madeObservation var:Meas, var:Pos .

  var:Meas a sosa:Observation ;
  sosa:hasResult [ a sosa:Result, geosparql
    :Geometry ;
  iiot:hasResultValue var:Geo ] ;
  sosa:phenomenonTime [ a time:Instant ;
    time:inXSDDateTime var:Tsp ] ;
  sosa:observedProperty iiotex:BatteryLevel
  .

  var:Pos a sosa:Observation ;
  sosa:hasResult [ a sosa:Result ;
  iiot:hasResultValue var:Geo ] ;
  sosa:observedProperty iiotex:Position .
</align:entity2>
<align:relation>=</align:relation>
<sripas:transformation rdf:parseType="
  Literal">
  <sripas:function about="str">
    <param order="1" about="&var;Position
      "/>
    <return about="&var;sGeo"/>

```

```

</sripas:function>
<sripas:function about="replace">
  <param order="1" about="&var;sGeo"/>
  <param order="2" val="^(\\d+\\.\\d+)
    \\s+\\d+\\.\\d+$/>
  <param order="3" val="$1"/>
  <param order="4" val="i"/>
  <return about="&var;Lat"/>
</sripas:function>
<sripas:function about="replace">
  <param order="1" about="&var;sGeo"/>
  <param order="2" val="^\\d+\\.\\d+\\s
    +(\\d+\\.\\d+)$"/>
  <param order="3" val="$1"/>
  <return about="&var;Long"/>
</sripas:function>
<sripas:function about="concat">
  <param order="1" val="Point("/>
  <param order="2" about="&var;Lat"/>
  <param order="3" val=" "/>
  <param order="4" about="&var;Long"/>
  <param order="5" val=")"/>
  <return about="&var;Geo"/>
</sripas:function>
</sripas:transformation>
<sripas:filters rdf:parseType="Literal">
  <sripas:filter about="&var;sGeo" datatype=
    "&xsd:string"/>
  <sripas:filter about="&var;Lat" datatype=
    "&xsd:float"/>
  <sripas:filter about="&var;Long" datatype=
    "&xsd:float"/>
</sripas:filters>
<sripas:typings rdf:parseType="Literal">
  <sripas:typing about="&var;Geo"
    datatype="http://www.opengis.net/def/
      sf/wktLiteral"/>
</sripas:typings>
</align:Cell>
</align:map>
</align:Alignment>

```

Listing 2 IPSM-AF alignment between drone and dTraffic

6 IPSM Semantic Translation Tool

IPSM (Inter-Platform Semantic Mediator) is a generic streaming semantic translation software developed by INTER-IoT. It realizes the idea of translation through alignments (see, Sect. 4). IPSM offers highly-scalable, efficient translation architecture applicable in any semantic translation scenario, and configurable with IPSM-AF files, regardless of domain of application.

There are two main interfaces to access IPSM: REST and reactive streaming. The first option offers quick, one-message-at-a-time, service, while the latter is better suited for large asynchronous streams of data. In both cases, translation is done transactionally “per message” by applying rules defined in alignment files. Technically, there is no limit on contents or size of messages, so the software could be used to perform a one-time batch translation of a whole database worth of data. Nevertheless, the design of IPSM makes it better suited for scenarios, where data that needs translation is not known beforehand, and thus cannot be translated in a single operation. This approach lends itself very well to communication between diverse IoT artifacts.

Reactive streaming, in IPSM, rests on semantic translation channels (*channels*, for short), and a *central ontology* (CO). A channel is a one way stream that accepts messages described with a given ontology and translates them to a different ontology. Just like in any reactive streaming implementation, inputs and outputs of channels may be read from, written to, connected to other stream processors, and even to each other. In IPSM, translation is performed “on the fly”, as messages pass the channel, and is configured by two one-way alignments per channel. The 2 alignment requirement is dictated by the central ontology architecture (described further on). Consequently, to set up a two-way communication one needs to configure 2 channels and 4 one-way alignments (2 per channel).

CO is a shared vocabulary, founded on the concept of core ontology (see, Sects. 1.1 and 2) that is going to be used in the process of translation, but does not need to be implemented natively by the communicating artifacts. CO is a special, modular, ontology used as an *intermediary* in translation (see, Fig. 5).

Assuming that IoT artifacts A , B that want to communicate, and with the central ontology Ω in place (see, Fig. 5), an IPSM channel should be configured with one alignment that defines translation from A semantics into Ω (denoted $A \triangleright \Omega$), and another one from Ω into B (alignment $\Omega \triangleright B$). Messages flowing through are first translated using $A \triangleright \Omega$, and then $\Omega \triangleright B$. In order to set up a different channel, say from A to C , the same alignment $A \triangleright \Omega$ may be used as the “input” configuration, with a new alignment $\Omega \triangleright C$ for the “output”. In this way, from the perspective of A , it is enough to learn Ω and provide alignments to and from its own semantics ($A \triangleright \Omega$ and $\Omega \triangleright A$). Once A makes such alignments available publicly by uploading them to an instance of IPSM (i.e. its alignment repository component), anyone may use them to configure communication to and from their own artifact. Technically, IPSM has a built-in “identity” alignment that effectively performs no translation, and is used to relax the two-alignment requirement, in case an artifact natively supports Ω . Using

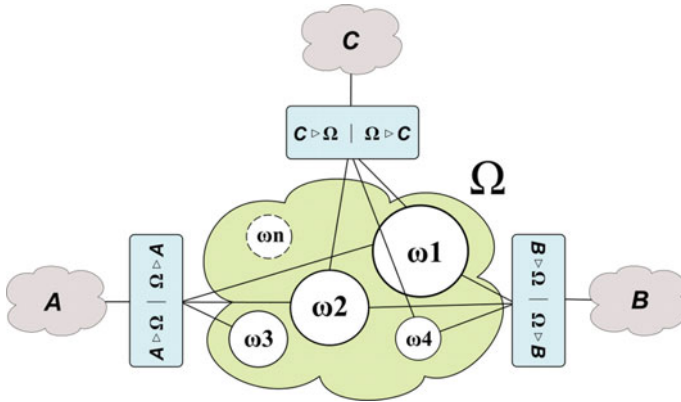


Fig. 5 IPISM configuration with modularized central ontology Ω

CO as an intermediate step, IPISM facilitates translation by mediating it through CO, as opposed to pipelining it directly (one-to-one). Therefore, the cost of joining a deployed IPISM ecosystem, regardless of its size, is always just 2 alignments.

IPISM places no technical requirements on the CO (any ontology can be used), but there are features that make some better suited for a CO than others. First of all, a good CO should have a broad range, and cover, in detail, its domain. Because artifacts translate to and from CO, a broad and detailed (highly granular) semantics will allow for translation of full range of messages from all artifacts without loss of specificity. Second, (well documented) modularity of CO lowers the cost of authoring alignments, because an engineer only needs to focus on the modules that are relevant to their artifact, and can disregard others. For instance, if only two artifacts in an ecosystem use descriptions of sea-faring vehicles, the sea module of a CO will not be of interest to any other participant, because their artifacts simply do not support it, i.e. they do not “talk” about sea vehicles, regardless of translation. Overall, while the CO has to cover topics that the participants “want to communicate about” it does not have to cover the whole range of semantics that participating artifacts do.

By design, all alignments in IPISM are “public” (i.e. the alignment repository in an IPISM instance is available to anyone that has the rights to configure channels, in the same instance), so the division of Ω into modules $\omega_1, \omega_2, \dots$ has bearing on security. It is up to the artifact owner to decide what messages to send (and when), what the scope of its own schema that the artifact will use to communicate with others is, and what CO modules will the alignment apply. Technically, IPISM-AF alignments can be used to redact, obfuscate, and anonymize data, but because IPISM instances are de facto external to the data origin, IPISM should not be used as a data security measure, unless run in a closed, trusted network.

More information about IPSM, its architecture, performance, use-case scenarios, and deployment methods can be found in [32–34], as well as in the INTER-IoT online documentation.^{18, 19}

7 Use Case Processing

Thus far we have considered and discussed different facets of our use-case scenario (see, Sect. 1.2 and Fig. 1). The scenario involves actors with various communication needs that can be realized with deployment of a number of IPSM instances.

Let us recall that the *dTraffic* actor needs interoperability in communication of drones with a central “base of operations”. Here, we assume that *dTraffic* cooperates with several types of drones that use different data models to represent position, battery level, and traffic congestion level. They can be based on one of the core ontologies extended with domain ontologies e.g. drone type 1 uses SOSA/SSN for drone description and W3C Basic Geo Vocabulary for geopositioning, whereas drone type 2 uses SAREF with GeoSPARQL. Additionally, drones may come from producers using their native data models (not based on any standard). In each case, congestion level is usually represented in a drone-specific way, since there are no publicly available standards. Assuming impossibility of implementing the same semantics across artifacts (now and in the future), the two main approaches to establish interoperability with different drones are to implement: (i) a bi-directional (one-to-one) translation mechanism between *dTraffic* and each drone type, (ii) mediator with a CO, to which messages exchanged in both directions are translated (see, Sect. 6). The former approach is considerably more difficult and complex to realize in a dynamic ecosystem with multiple IoT artifacts. Moreover, it implies system modification, every time a new type of drone is added (defeating the 24/7 availability). In the latter approach, preexisting artifacts are not affected by the integration. Addition of new device type(s) requires translation to and/or from the CO. Therefore, it is the mediator-based approach that was selected in INTER-IoT to provide interoperability on data and semantics layer, and is recommended in the drones use case.

Similarly, *iotDelivery* needs to exchange data with trucks. This also requires choice of data models. Since there are several publicly available standards in the transportation and logistics domain, we may assume that e.g. truck company 1 uses LogiCO and LogiServ ontologies, whereas truck company 2 might use OTN.

Let us now consider how semantic translation in our use case can be organized. In the presented scenario, we may consider two IPSM instances that serve as communication hubs: *dTraffic Hub* and *iotDelivery Hub*. The former translates messages exchanged between different drone types and *dTraffic* (which can be assumed to work with the *dTraffic Hub* data model directly). The latter enables exchange of

¹⁸ <https://inter-iot.readthedocs.io/projects/ipsm/en/latest/>.

¹⁹ <https://inter-iot-cookbook.readthedocs.io/en/latest/inter-layer/ds2ds/appendices/products/>.

information between trucks, *iParking* companies, and *dTraffic*, in order to schedule trucks and reserve parking slots for them.

Next, a set of alignments needs to be prepared: (i) bidirectional, between data models of drones of different types and the *dTraffic Hub* central data model, (ii) bidirectional, between data models of *iParking* companies and the *iotDelivery Hub* central data model, (iii) bidirectional, between data models of truck companies and the *iotDelivery Hub*, (iv) unidirectional, between *dTraffic* central data model and the *iotDelivery Hub* central data model, and (v) bidirectional, between data models of *iotDelivery* companies and *iotDelivery Hub* central data model. To illustrate how the proposed approach is going to work, let consider a sample message send by the drone, that should be consumed by *dTraffic*.

```
{
  "@graph" : [ {
    "@graph" : [ {
      "@id" : "InterIoTMsg:meta308c3987-b69e-4672-890b-3
        f3d6229596d",
      "@type" : [ "InterIoTMsg:meta", "InterIoTMsg:
        Thing_Update" ],
      "InterIoTMsg:conversationID" : "conv85e0f5d2-cf65
        -4d23-84b1-ff1381ae01fc",
      "InterIoTMsg:dateTimeStamp" : "2019-02-08T13
        :48:19.428+00:00",
      "InterIoTMsg:messageID" : "msg204d0405-a6da-4054-
        a6db-96d20c413746"
    } ],
    "@id" : "InterIoTMsg:metadata"
  }, {
    "@graph" : [
      {
        "@id": "http://www.drone1.eu/devices/Device_1",
        "@type": "saref:Device",
        "saref:hasState": {
          "@id": "saref:Start"
        },
        "saref:makesMeasurement": [
          { "@id": "_:Pos" },
          { "@id": "_:Meas" }
        ]
      },
      {
        "@id": "_:Pos",
        "@type": "saref:Measurement",
        "saref:hasValue": "45.256 -71.92",
        "saref:relatesToProperty": {
```

```

        "@id": "sarefInst:Position"
    }
},
{
    "@id": " _:Meas",
    "@type": "saref:Measurement",
    "saref:hasTimestamp": {
        "@type" : "http://www.w3.org/2001/XMLSchema#
            dateTime",
        "@value" : "2019-02-08T13:48:18"
    },
    "saref:hasValue": {
        "@type" : "http://www.w3.org/2001/XMLSchema#
            float",
        "@value" : "0.75"
    },
    "saref:isMeasuredIn": {
        "@id": "sarefInst:Percentage"
    },
    "saref:relatesToProperty": {
        "@id": "sarefInst:BatteryLevel"
    }
}
],
"@id" : "InterIoTMsg:payload"
} ],
"@context" : {
    "InterIoTMsg" : "http://inter-iot.eu/message/",
    "InterIoTInst" : "http://inter-iot.eu/instance/",
    "owl" : "http://www.w3.org/2002/07/owl#",
    "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns
        #",
    "xsd" : "http://www.w3.org/2001/XMLSchema#",
    "rdfs" : "http://www.w3.org/2000/01/rdf-schema#",
    "InterIoT" : "http://inter-iot.eu/",
    "sosa" : "http://www.w3.org/ns/sosa/",
    "saref" : "https://w3id.org/saref#",
    "sarefInst" : "https://w3id.org/saref/instances/"
}
}

```

Listing 3 Status message from drone

Listing 3 shows a status message sent periodically by a drone, stating its current position and battery level. Note that each drone type sends this kind of message utilizing its own data model. This message can be translated with alignment given in Listing 2. It matches pattern in *entity1* of cell *2_status* that defines how it should be translated to the data model based on SOSA/SSN and GeoSPARQL (*dTraffic Hub* central data model is assumed to use these ontologies).

Listing 4 shows a sample message that can be sent from one of *iotDelivery* companies. Information is expressed in *iotDelivery1* native semantics and contains Car ID and time period for the reservation. Lets assume that reservation should be done by *iParking* company named *iParking1* that also uses its own native semantics. In such case, we need to use two alignments: one from the *iotDelivery1* to the *iotDelivery Hub* central data model (see, Listing 5), the other from the *iotDelivery Hub* to the *iParking1* data model (see, Listing 6).

```
{
  "@graph" : [ {
    "@graph" : [ {
      "@id" : "InterIoTMsg:meta308c3987-b69e-4672-890b-3
        f3d6229596d",
      "@type" : [ "InterIoTMsg:meta", "InterIoTMsg:
        Thing_Update" ],
      "InterIoTMsg:conversationID" : "conv85e0f5d2-cf65
        -4d23-84b1-ff1381ae01fc",
      "InterIoTMsg:dateTimeStamp" : "2019-02-08T13
        :48:19.428+00:00",
      "InterIoTMsg:messageID" : "msg204d0405-a6da-4054-
        a6db-96d20c413746"
    } ],
    "@id" : "InterIoTMsg:metadata"
  } ], {
    "@graph" : [
      {
        "@id": "iotDeliveryInst:Reservation1",
        "@type": "iotDelivery:Reservation",
        "iotDelivery:hasCarId": "WK 12345",
        "iotDelivery:hasEndTime": {
          "@type" : "http://www.w3.org/2001/XMLSchema#
            dateTime",
          "@value" : "2019-02-08T13:50:00"
        },
      },
      "iotDelivery:hasStartTime": {
        "@type" : "http://www.w3.org/2001/XMLSchema#
          dateTime",
        "@value" : "2019-02-08T14:00:00"
      }
    ]
  }
}
```

```

    }
  ],
  "@id" : "InterIoTMsg:payload"
} ],
"@context" : {
  "InterIoTMsg" : "http://inter-iot.eu/message/",
  "InterIoTInst" : "http://inter-iot.eu/instance/",
  "owl" : "http://www.w3.org/2002/07/owl#",
  "rdf" : "http://www.w3.org/1999/02/22-rdf-syntax-ns#",
  "xsd" : "http://www.w3.org/2001/XMLSchema#",
  "rdfs" : "http://www.w3.org/2000/01/rdf-schema#",
  "InterIoT" : "http://inter-iot.eu/",
  "iotDelivery" : "https://iotDelivery1.org#",
  "iotDeliveryInst" : "https://iotDelivery1.org/instances/"
}
}
}

```

Listing 4 Reservation message from *iotDelivery1*

Listing 5 shows alignment with one cell that matches message from Listing 4.

```

<?xml version="1.0" encoding="utf-8" standalone="no"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
  xmlns:sripas="http://www.inter-iot.eu/sripas#"
  xmlns="http://www.inter-iot.eu/sripas#"
  xmlns:align="http://knowledgeweb.semanticweb.org/heterogeneity/alignment#"
  xmlns:dcelem="http://purl.org/dc/elements/1.1/"
  xmlns:exmo="http://exmo.inrialpes.fr/align/ext/1.0/#"
  xmlns:var="http://www.inter-iot.eu/sripas#node_"
  xmlns:sosa="http://www.w3.org/ns/sosa/"
  xmlns:time="http://www.w3.org/2006/time#"
  xmlns:iiot="http://inter-iot.eu/GOIOTP#"
  xmlns:iiotex="http://inter-iot.eu/GOIOTPex#"
  xmlns:ssn="http://www.w3.org/ns/ssn/"
  xmlns:iotDelivery="https://iotDelivery1.org#"
  xmlns:iotDeliveryInst="https://iotDelivery1.org/instances/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:logico="http://ontology.tno.nl/logico#"

```

>

```

<align:Alignment>

  <dcelem:title>IoTDelivery1_IoTDeliveryCO</dcelem
    :title>
  <exmo:version>1.0</exmo:version>
  <dcelem:creator>SRIPAS</dcelem:creator>
  <dcelem:description>Between IoT-Delivery1 and
    IoT-Delivery Hub.</dcelem:description>

  <align:xml>yes</align:xml>
  <align:level>2IPSM</align:level>
  <align:type>**</align:type>
  <align:method>>manual</align:method>
  <dcelem:date>13-02-2019</dcelem:date>
  <sripas:cellFormat>
    <iiot:DataFormat rdf:about="http://inter-iot
      .eu/sripas#turtle" />
  </sripas:cellFormat>

  <align:ontol1>
    <align:Ontology rdf:about="https://
      iotDelivery1.org#">
      <align:formalism>
        <align:Formalism
          align:name="OWL2.0" align:uri="http
            ://www.w3.org/2002/07/owl#" />
        </align:formalism>
      </align:Ontology>
    </align:ontol1>
  <align:ontol2>
    <align:Ontology rdf:about="http://inter-iot.
      eu/GOIoTPex#">
      <align:formalism>
        <align:Formalism
          align:name="OWL2.0" align:uri="http
            ://www.w3.org/2002/07/owl#" />
        </align:formalism>
      </align:Ontology>
    </align:ontol2>

  <sripas:steps rdf:parseType="Collection">
    <sripas:step sripas:order="1"
      sripas:cell="http://www.inter-iot.eu/sripas
        #1_reservation"/>

```



```

</sripas:steps>

<align:map>
  <align:Cell rdf:about="http://www.inter-iot.
    eu/sripas#1_reservation">
    <align:entity1 rdf:datatype="http://www.
      w3.org/2001/XMLSchema#string">
      var:R a iotDelivery:Reservation ;
      iotDelivery:hasCarId var:CarId ;
      iotDelivery:hasStartTime var:Time1 ;
      iotDelivery:hasEndTime var:Time2 .
    </align:entity1>
    <align:entity2 rdf:datatype="http://www.
      w3.org/2001/XMLSchema#string">
      var:R a iiotex:Reservation ;
      iiotex:hasIssuer [
        a logico:Truck ;
        logico:id [ logico:hasIdValue var
          :CarId ]
      ] ;
      iiotex:forRegion [
        a logico:Region ;
        logico:id [ logico:hasIdValue var
          :ParkingId ;
          logico:hasAgency iiotex:
            ParkingRegistry
        ] ;
        iiotex:hasTimebox [
          a time:Interval ;
          time:hasBeginning [
            a time:Instant; time:
              inXSDDateTimeStamp var:
                Time1
          ] ;
          time:hasEnd [
            a time:Instant; time:
              inXSDDateTimeStamp var:
                Time2
          ]
        ]
      ] .
    </align:entity2>
    <align:relation>=</align:relation>
  </align:Cell>
</align:map>

```

```

    </align:Alignment>
</rdf:RDF>

```

Listing 5 Alignment between *iotDelivery1* and *iotDelivery Hub* central data model

Listing 6 shows alignment with one cell that matches output message, after translation of message from Listing 4 with alignment from Listing 5.

```

<?xml version="1.0" encoding="utf-8" standalone="no"?>
<rdf:RDF xmlns:rdf="http://www.w3.org/1999/02/22-rdf-
  syntax-ns#"
  xmlns:sripas="http://www.inter-iot.eu/sripas#"
  "
  xmlns="http://www.inter-iot.eu/sripas#"
  xmlns:align="http://knowledgeweb.semanticweb.
    org/heterogeneity/alignment#"
  xmlns:dcelem="http://purl.org/dc/elements
    /1.1/"
  xmlns:exmo="http://exmo.inrialpes.fr/align/
    ext/1.0/#"
  xmlns:var="http://www.inter-iot.eu/sripas#
    node_"
  xmlns:sosa="http://www.w3.org/ns/sosa/"
  xmlns:time="http://www.w3.org/2006/time#"
  xmlns:iiot="http://inter-iot.eu/GOIoTP#"
  xmlns:iiotex="http://inter-iot.eu/GOIoTPex#"
  xmlns:ssn="http://www.w3.org/ns/ssn/"
  xmlns:iParking="https://iParking1.org#"
  xmlns:iParkingInst="https://iParking1.org/
    instances/"
  xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
  xmlns:logico="http://ontology.tno.nl/logico#"
>
  <align:Alignment>
    <dcelem:title>IoTDeliveryCO_iParking1</dcelem:
      title>
    <exmo:version>1.0</exmo:version>
    <dcelem:creator>SRIPAS</dcelem:creator>
    <dcelem:description>Between IoT-Delivery Hub and
      iParking1.</dcelem:description>
    <align:xml>yes</align:xml>
    <align:level>2IPSM</align:level>
    <align:type>**</align:type>
    <align:method>manual</align:method>

```

```

<dcelem:date>13-02-2019</dcelem:date>
<sripas:cellFormat>
  <iiot:DataFormat rdf:about="http://inter-iot
    .eu/sripas#turtle" />
</sripas:cellFormat>

<align:onto1>
  <align:Ontology rdf:about="http://inter-iot.
    eu/GOIoTPex#">
    <align:formalism>
      <align:Formalism align:name="OWL2.0"
        align:uri="http://www.w3.org
          /2002/07/owl#" />
    </align:formalism>
  </align:Ontology>
</align:onto1>
<align:onto2>
  <align:Ontology rdf:about="https://iParking1
    .org#">
    <align:formalism>
      <align:Formalism align:name="OWL2.0"
        align:uri="http://www.w3.org
          /2002/07/owl#" />
    </align:formalism>
  </align:Ontology>
</align:onto2>

<sripas:steps rdf:parseType="Collection">
  <sripas:step sripas:order="1"
    sripas:cell="http://www.inter-iot.eu/sripas
      #1_reservation"/>
</sripas:steps>

<align:map>
  <align:Cell rdf:about="http://www.inter-iot.
    eu/sripas#1_reservation">
    <align:entity1 rdf:datatype="http://www.
      w3.org/2001/XMLSchema#string">
      var:R a iiotex:Reservation ;
      iiotex:hasIssuer [
        a logico:Truck ;
        logico:id [ logico:hasIdValue var
          :CarId ]
      ] ;
      iiotex:forRegion [

```

```

    a logico:Region ;
    logico:id [
      logico:hasIdValue var:
        ParkingId ;
      logico:hasAgency iiotex:
        ParkingRegistry
    ] ;
    iiotex:hasTimebox [
      a time:Interval ;
      time:hasBeginning [
        a time:Instant; time:
          inXSDDateTimeStamp var:
            Time1
      ] ;
      time:hasEnd [
        a time:Instant; time:
          inXSDDateTimeStamp var:
            Time2
      ]
    ]
  ] .
</align:entity1>
<align:entity2 rdf:datatype="http://www.
  w3.org/2001/XMLSchema#string">
  var:R a iParking:Reservation ;
  iParking:hasCarId var:CarId ;
  iParking:hasStartTime var:Time1 ;
  iParking:hasEndTime var:Time2 .
</align:entity2>
<align:relation>=</align:relation>
</align:Cell>
</align:map>

</align:Alignment>
</rdf:RDF>

```

Listing 6 Alignment between *iotDelivery Hub* central data model and *iParking*

8 Concluding Remarks

In this chapter we summarized the INTER-IoT approach to semantic interoperability. The simple *dTraffic* mock scenario was deconstructed throughout the text to expose the affluent semantic methods, techniques, and advances weaved into every aspect

of design and implementation of semantically interoperable IoT systems that use INTER-IoT. Arisen from the study of ontology alignments, the INTER-IoT pathway is lined with elucidation of semantics into a very explicit and concrete form. Such concretization empowered the novel IPSM-AF format, and paved way to the practical application of ontology alignments as translation rules for real-time dynamic transformation of data. Driven by the needs and wants commonly emerging in the IoT domain, the design of a flexible stream-driven architecture for efficient, scalable, and reactive semantic translation of messages was crowned with the implementation of IPSM software. Additionally, introduction of the idea of a *central ontology* capacitated multi-deployments of IPSM among considerable amount of communicating artifacts with minimal cost of changes in the ecosystem, even during uninterrupted operation.

The generic nature of proposed solutions makes the INTER-IoT approach to semantics feasible for any domain that requires data semantics interoperability. Additionally we developed two modular and extendable ontologies: *GOIoTP* and *GOIoTPex*, ready to be used as central ontologies for any IoT-related sub-domain. [50].

Fruitful usage of the INTER-IoT approach to semantics, including the theory, designs, and software, in INTER-IoT pilot implementations, as well as other IoT projects (e.g. EU ACTIVAGE, EU Pixel) is a testimony to its potential and practicality.

References

1. Basic Geo (WGS84 lat/long) vocabulary. <https://www.w3.org/2003/01/geo/>
2. Bioportal ontologies. <https://bioportal.bioontology.org/ontologies>
3. Climate and forecast features. <https://www.w3.org/2005/Incubator/ssn/ssnx/cf/cf-feature>
4. DATEX II. <http://www.datex2.eu/>
5. EDOAL: Expressive and declarative ontology alignment language. <http://alignapi.gforge.inria.fr/edoal.html>
6. FHIR OWL Ontology. <https://w3c.github.io/hcls-fhir-rdf/spec/ontology.html>
7. A format for ontology alignment. <http://alignapi.gforge.inria.fr/format.html>
8. Geographic data format. http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=54610
9. geoSPARQL. <https://www.ogc.org/standards/geosparql>
10. The linked earth ontology. <https://linkedeath.github.io/ontology/>
11. Logical—transnational logistics improvement through cloud computing and innovative cooperative business models. <https://trimis.ec.europa.eu/project/transnational-logistics-improvement-through-cloud-computing-and-innovative-cooperative>
12. Logical cloud portal. [http://logical.bayzoltan.org/\[ENG\]/index_eng.html](http://logical.bayzoltan.org/[ENG]/index_eng.html)
13. Logico ontology. <https://ontology.tno.nl/logico/>
14. Logiserv ontology. <https://ontology.tno.nl/logiserv/>
15. Logistic grid ontology
16. The obo foundry. <http://www.obofoundry.org/>
17. Ontology of transportation networks
18. PROV-O: The PROV Ontology. <https://www.w3.org/TR/prov-o/>
19. Resource description framework (RDF). <https://www.w3.org/RDF/>

20. Rewerse: Reasoning on the web. <http://reverse.net/>
21. Seas-weatherontology ontology. <https://ci.mines-stetienne.fr/seas/WeatherOntology>
22. The transport disruption ontology. <https://transportdisruption.github.io/transportdisruption.html>
23. UniversAAL Ontologies. <https://github.com/universAAL/ontology>
24. Achieving technical interoperability—the ETSI approach. ETSI White Paper No. 3, April (2008)
25. Semantic Sensor Network XG final report, 2011
26. Auer, S., Lehmann, J., Hellmann, S.: LinkedGeoData: adding a spatial dimension to the web of data. In: Bernstein, A., Karger, D.R., Heath, T., Feigenbaum, L., Maynard, D., Motta, E., Thirunarayan, K. (eds) *The Semantic Web—ISWC 2009*, pp. 731–746. Springer, Berlin, Heidelberg (2009)
27. Aumüller, D., Do, H.-H., Massmann, S., Rahm, E.: Schema and ontology matching with COMA++. Presented at the (2005)
28. Bermudez-Edo, M., Elsaleh, T., Barnaghi, P., Taylor, K.: IoT-Lite: a lightweight semantic model for the Internet of Things and its use with dynamic semantics. *Pers. Ubiquitous Comput.* **21**(3), 475–487 (2017)
29. Compton, M., Barnaghi, P., Bermudez, L., Garcia-Castro, R., Corcho, O., Cox, S., Graybeal, J., Hauswirth, M., Henson, C., Herzog, A., Huang, V., Janowicz, K., Kelsey, W.D., Le Phuoc, D., Lefort, L., Leggieri, M., Neuhaus, H., Nikolov, A., Page, K., Passant, A., Sheth, A., Taylor, K.: The SSN ontology of the W3C semantic sensor network incubator group. *Web Semant.: Sci., Serv. Agents World Wide Web* **17**, 25–32 (2012)
30. Jérôme Euzenat and Pavel Shvaiko. *Ontology Matching*, 2nd edn. Springer (2013)
31. Faria, D., Pesquita, C., Santos, E., Palmonari, M., Cruz, I.F., Couto, F.M. (eds.): *The AgreementMakerLight Ontology Matching System*, pp. 527–541. Springer (2013)
32. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K.: Streaming Semantic Translations, pp. 1–8. IEEE (2017)
33. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K.: Alignment-based semantic translation of geospatial data. In: *2017 3rd International Conference on Advances in Computing, Communication & Automation (ICACCA) (Fall)*, Dehradun, India, pp. 1–8 (2017)
34. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K., Solarz-Niesłuchowski, B., Suárez de Puga García, J.: Towards high throughput semantic translation. In: Fortino, G., Palau, C.E., Guerrieri, A., Cuppens, N., Cuppens, F., Chaouchi, H., Gabillon, A. (eds.) *Interoperability, Safety and Security in IoT*, pp. 7–74, Cham, 2018. Springer International Publishing
35. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K.: Semantic technologies for the IoT—an Inter-IoT perspective, pp. 271–276. IEEE, Berlin, Germany, April (2016)
36. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K.: Towards common vocabulary for IoT ecosystems—preliminary considerations. In: *Intelligent Information and Database Systems, 9th Asian Conference, ACIIDS 2017, Kanazawa, Japan, April 3–5, 2017, Proceedings, Part I, Volume 10191 of LNCS*, pp. 35–45. Springer (2017)
37. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K.: Towards semantic interoperability between Internet of Things platforms. In: Gravina, R., Palau, C.E., Manso, M., Lotta, A., Fortino, G. (eds.) *Integration, Interconnection, and Interoperability of IoT Systems*, pp. 103–127. Springer (2017)
38. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K., Fortino, G.: Tools for ontology matching—practical considerations from INTER-IoT perspective, pp. 296–307. Springer (2016)
39. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K., Palau, C.E.: From implicit semantics towards ontologies—practical considerations from the INTER-IoT perspective (submitted for publication). Presented at the (2017)
40. Gruber, T.R.: Toward principles for the design of ontologies used for knowledge sharing? *Int. J. Hum.-Comput. Stud.* **43**(5), 907–928 (1995)

41. Jiménez-Ruiz, E., Grau, B.C.: LogMap: Logic-based and scalable ontology matching, pp. 273–288. Springer (2011)
42. Lorenz, B.: Hans Jürgen Ohlbach, and Laibing Yang. Ontology of transportation networks (2005). <http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.133.2953>
43. Mètral, C., Billen, R.: Anne-Francoise Cutting-Decelle, and Muriel Van Ruymbeke. Ontology-based approaches for improving the interoperability between 3d urban models. *J. Inf. Technol. Constr.* **15** (2010)
44. Scheuermann, A., Hoxha, J.: Ontologies for intelligent provision of logistics services. In: 7th International Conference on Internet and Web Applications and Services (ICIW 2012), Germany, May 2012. XPS
45. Staab, S., Studer, R.: Handbook on Ontologies, 2 edn. Springer (2009)
46. Strzelczak, S.: Core ontology for manufacturing and logistics. *Zeszyty Naukowe. Organizacja i Zarzadzanie/Politechnika Slaska* **73**, 603–618 (2014)
47. Stuckenschmidt, H., Parent, C., Spaccapietra, S. (eds.): Modular Ontologies. Concepts, Theories and Techniques for Knowledge Modularization, Volume 5445 of State-of-the-Art Survey, LLNCS (2009)
48. Belsa, A., Sarabia-Jacome, D., Palau, C.E., Esteve, M.: Flow-based programming interoperability solution for IoT platform applications. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 304–309, Orlando (FL) (February 2018)
49. Broring, A., Zappa, A., Vermesan, O., Främling, K., Zaslavsky, A., Gonzalez-Usach, R., Szmeja, P., Palau, C., Jacoby, M., Zarko, I.P., S. Sour-sos, C. Schmitt, M. Plociennik, S. Krco, S. Georgoulas, I. Larizgoitia, N. Gligoric, R. Garcia-Castro, F. Serena, V. Orav. : Advancing IoT Platform Interoperability. River Publishers (2018)
50. Giancarlo, F., Antonio, L., Carlos, P., Raffaele, G., Marco, M. (eds.): Integration, Interconnection, and Interoperability of IoT Systems. Springer (February 2017)
51. Pileggi, S.F., Palau, C.E., Esteve, M.: Building semantic sensor web: knowledge and interoperability. In: Proceedings of the International Workshop on Semantic Sensor Web, Volume 1: SSW, (IC3K 2010), pp. 15.22 (April 2010)
52. Vermesan, O., Friess, P. (eds.): Digitising the Industry Internet of Things Connecting the Physical. River Publishers, Digital and Virtual Worlds (2016)
53. Fortino, G., Garro, A., Russo, W.: Achieving Mobile Agent Systems interoperability through software layering. *Inf. Softw. Technol.* **50**(4), 322–341 (2008)

INTER-Framework: An Interoperability Framework to Support IoT Platform Interoperability



Clara I. Valero, Andreu Belsa, Alejandro Fornes-Leal, Fernando Boronat, Miguel A. Llorente, and Miguel Montesinos

Abstract INTER-Framework solution provides a way to homogenize the use of the interoperability layered infrastructure. By using INTER-FW, any IoT platform can be made interoperable with respect to its device, network, middleware and service layer. This tool offers a complete visual framework to configure and manage in a secure way and to develop new software applications leveraging data from multiple IoT heterogeneous platforms. It facilitates the creation of an ecosystem of interoperable and open IoT platforms. Thus, the development time of novel IoT services and applications can be shortened, and these services can be provided atop interoperable IoT platforms. It is reflected in lower development effort and economical costs for product owners, users, developers and platform integrators. The framework includes components to address several requirements like security, API management, data visualization, scalability and extensibility. INTER-FW also contains the identity and authorization mechanisms of INTER-IoT. In addition, it contains INTER-API, a gateway of the APIs from the different layers. INTER-API is managed and configured within INTER-FW using the API Manager and the Identity Server.

1 Introduction

The overall goal of the INTER-IoT project is to provide an interoperable and open IoT framework, with associated engineering tools and methodology, for seamless integration of heterogeneous IoT platforms [1], regardless of the application domains. As seen in previous chapters, INTER-IoT uses a layer-oriented approach [2]. The interoperability framework addresses interoperability issues between different heterogeneous IoT platforms. By using the INTER-FW, any IoT platform can be made interoperable with respect to its fundamental layers: device, networking, middleware, applications, and semantics. INTER-FW will facilitate creation of an ecosystem of interoperable and open IoT platforms. Thus, development time of novel IoT services and applications can be shortened, and these services can be provided atop interoper-

C. I. Valero (✉) · A. Belsa · A. Fornes-Leal · F. Boronat · M. A. Llorente · M. Montesinos
UPV, Universidad Politécnica de Valencia, Valencia, Spain
e-mail: clavalpe@upv.es

© Springer Nature Switzerland AG 2021

C. E. Palau (eds.), *Interoperability of Heterogeneous IoT Platforms*, Internet of Things,
https://doi.org/10.1007/978-3-030-82446-4_6

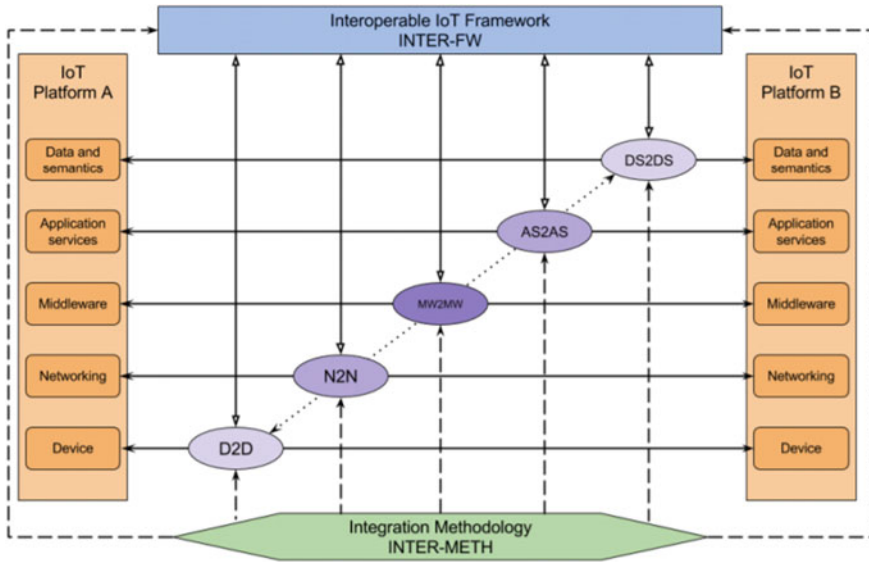


Fig. 1 INTER-IoT multi-layered approach

erable IoT platforms. Furthermore, from a business point of view, interoperability will result in lower costs [3].

The modular layered approach followed in INTER-IoT has multiple advantages such as strong flexibility, adaptability to very different scenarios, separation of concerns, decoupling between functional components, scalability across physical tiers, etc. However, the approach also presents shortcomings in terms of usability or tooling unification, especially in the case of INTER-IoT, where each Interoperability Layer Infrastructure (ILI) is designed to be completely decoupled, being able to work standalone (Fig. 1).

Other important challenge of the INTER-IoT global design, affecting each layer separately, is the future-proof design. One of the fundamental problems solved in INTER-IoT, the isolation of IoT data in information domains and the inability to automatically share these data across these domains [4, 5], is a consequence of the excessive short-term sight in the design or instantiation of IoT systems. INTER-IoT addresses these drawbacks by the introduction of a framework based on the six ILIs (D2D, N2N, MW2MW, AS2AS, DS2DS and CL) exposed APIs. This framework, presented as the INTER-FRAMEWORK or INTER-FW [6], has three main goals identified:

- Enable extension and scalability of the INTER-IoT solution to support present and future applications or more demanding scenarios.
- Configure, monitor and manage the different layers from a unique view.
- Provide and manage a REST-like API to enable the use of the interoperable platforms and INTER-IoT features (Fig. 2).

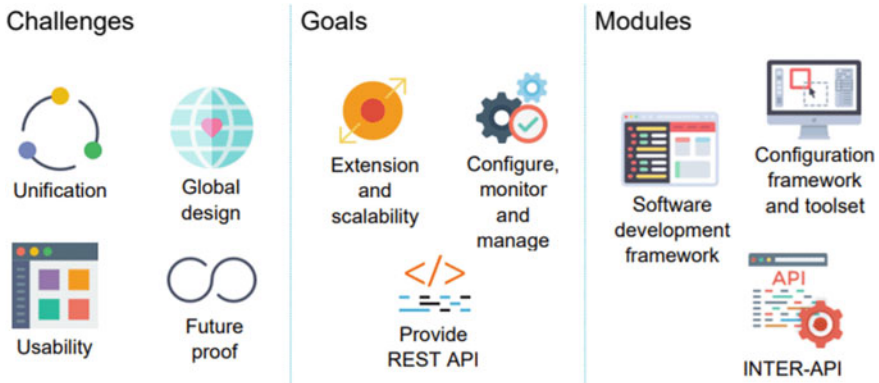


Fig. 2 INTER-FW overview

These goals are addressed in the following functional components developed in INTER-FW:

- Software development framework: a combination of software results, documentation, templates and examples allowing extensibility through the application of the INTER-IoT interoperability patterns in new platforms, devices, services, etc.
- Instantiation, configuration and management framework and toolset: a web based application which unifies in a single environment the monitoring, management configuration of the different IoT components (sensors, network elements, gateways, platforms...) at each interoperability layer. This element also includes key cross-layer elements management, such as authentication or authorization configurations.
- A global, unified API to offer a single-entry point to application and services developers, either platform owners or third parties.

2 Framework Approaches to Interoperable IoT Platforms

In software engineering, the concept of ‘framework’ is wide [7] and can be applied in very different disciplines, depending strongly on the approach and the goal of the software artefact resulting [8]. The meaning of this term and some related concepts are reviewed below. The types of frameworks used in INTER-IoT are also detailed.

2.1 Introduction to Frameworks

There are some related concepts that are frequently related to a software framework. Some of them are quickly reviewed in this section.

Library

A library is a set of functional implementations, coded in a programming language, which provides a well-defined interface for the functionality being invoked. Unlike an executable program, the behavior that implements a library does not expect to be used in an autonomous way, but its purpose is to be used by other programs, independently and simultaneously. Most modern operating systems provide libraries that implement system services. In this way, these services have become a “raw material” that any modern application expects the operating system to provide. As such, most of the code used by modern applications is offered in these libraries.

Toolset

A toolset is a more specific library used to create, debug, maintain, or support other programs and applications. In short, is a software that assists in the creation of new software. It could include advanced interfaces solutions such as graphical widgets or GUI elements either in desktop or web environments. Toolsets usually operate in a higher level of abstraction of a library, being very often library consumers (it is very common to find libraries distributions with an associated toolset e.g. mosquito,¹ jmeter,² docker,³ etc.).

Framework

A framework is a conceptual and technological support structure, usually with specific software artefacts or modules, which can serve as a basis for the organisation and development of software. The definition of a framework is usually linked to one of its key properties, the Inversion of Control (IoC) principle, which means that the program flow is taken over by the framework itself (a pre-existing and external element to the program), instead of the common case in imperative programming, where it is controlled by the developed program. However, frameworks are also targeted either towards a specific output; an application for a specific OS for instance (e.g., MFC⁴ for MS Windows), or for more general-purpose work (e.g., Spring framework⁵).

Software Development Kit (SDK)

An SDK is generally a set of software development tools that allows a software developer to create a computer application for a specific system, for example certain software packages, work environments, hardware platforms, computers, game consoles, operating systems, etc. It can consist of simply an application programming interface (API) created to allow the use of a certain programming language, or it can also include sophisticated hardware to communicate with a certain embedded system. The most common software development tools include support for programming error detection such as an Integrated Development Environment (IDE) and other utilities. SDKs often also include example code and supporting technical

¹ <https://mosquito.org/>.

² <https://jmeter.apache.org/>.

³ <https://www.docker.com/products/docker-desktop>.

⁴ <https://docs.microsoft.com/es-es/cpp/mfc/mfc-desktop-applications?view=msvc-160>.

⁵ <https://spring.io/projects/spring-framework>.

notes or other supporting documentation to help clarify certain points in the primary reference material.

Engine

An engine (in computer science terms) is a software system, or subsystem, that serves as the core foundation for a larger piece of software. An engine can also be defined as a computer program that facilitates automated processes, in which different pieces of software work interactively to minimize human intervention. A common feature of software engines is metadata that provides models of the actual data that the engine processes. The software modules pass data to the engine, and the engine uses its metadata models to transform the data into a different state. Examples of software engines include relational database engines, workflow engines, inference engines and search engines.

Application Programming Interface (API)

An API [9] is a set of subroutines, functions and procedures or methods that provides a certain library to be used by other software as an abstraction layer. An API represents the ability to communicate between software components. It acts as an interface between different software programs facilitating their interaction, such as the way the user interface facilitates interaction between humans and computers. They are used to overtake the interoperability limitations of the continuous creation and expansion of IT services and in the last years have become a trend. Other important driver of the API popularity is the creation of application-centric ecosystems, allowing third parties to create clients, use data and add value to the features of existing systems.

2.2 Types of Frameworks and Uses in INTER-IoT

For the sake of completeness, a short study of different popular types of frameworks has been performed. The results are gathered in this section.

2.2.1 Agent-Based Frameworks

A software agent [10] is a computer program that acts on behalf of a user or other program in a relationship of agency. Such “action on behalf of” implies a number of abilities that the agent need to possess: autonomous behaviour, social ability, responsiveness, proactiveness, and mobility. They represent a very expressive paradigm for modelling dynamic distributed systems, and perfectly fit the generic and specific requirements of IoT systems. Indeed, agent-based modelling could provide even more benefits in the IoT domain than in conventional distributed environments.

Agents may be autonomous or, often, they interact with other agents. In the latter case the environment in which the agents operate is called Multi-Agent Systems (MAS) [11]. A MAS is a middleware that basically provide an API for developing agent-based applications, and an agent server able to execute agents by providing

them with basic services such as agent creation, execution, migration, communication, and access to resources. Three popular agent platforms are briefly introduced in the following list:

- **Java Agent DEvelopment Framework** [12] is an open-source software Framework fully implemented in the Java language. It simplifies the implementation of Multi-Agent Systems through a middleware that complies with the FIPA specifications and through a set of graphical tools that support the debugging and deployment phases.
- **Mobile Agent Platform for SunSPOT** [13] is an innovative Java-based framework specifically developed on SunSPOT technology for enabling agent-oriented programming of WSN applications.
- **Java-based Interoperable Mobile Agent Framework** [14] focus on interoperability as a key issue for a wider adoption of Mobile Agent Systems (MASs) in heterogeneous and open distributed environments where agents, in order to fulfill their tasks, must interact with other non homogeneous agents and traverse different agent platforms to access remote resources.

Use in INTER-IoT

In the context of the INTER-IoT project, there are many advantages enabled (or favoured) using software agents. Agents can allow:

- Protocol encapsulation. In addition, in case of protocol upgrading, a new set of mobile agents can easily replace the old one at run-time.
- Natural orientation to heterogeneity. Mobile agents can act as wrappers among systems based on different hardware and software. This ability can well fit the need for integrating heterogeneous IoT devices based on different platforms. An agent may be able to translate requests coming from a system into specific suitable requests to submit to another different system.
- Robustness and fault-tolerance. The ability of mobile agents to dynamically react to adverse situations and events (e.g. low battery level) can lead to more robust and fault tolerant IoT systems; e.g. the reaction to the low battery level event can trigger the migration of all executing agents to an equivalent device to continue their activity.
- Devices and platforms configuration and management, enabled by agents migrating to the target device/platform or communicating with the local agent.

The use of agents and associated development frameworks can be found in the lower layers of INTER-IoT, such as the D2D level and in some parts of MW2MW. However, given the functional relation between INTER-FW and INTER-Layer [15] components (the former uses and aggregates functions of the latter), an extensive use of agent-based frameworks is not well suited for the INTER-FW, since it performs tasks of orchestration and encapsulation, while IoT interoperability-related business logic is frequently delegated to the lower layers.

2.2.2 Cloud-Based Frameworks

Before describing cloud-based frameworks [16], it is important to explain that cloud computing [17] is a paradigm that allows computing services to be offered over a network, which is usually the Internet. The term “cloud-based framework” is used to describe a wide range of solutions that support the creation, deployment and management of cloud applications. The types of services available in the cloud are described below:

- **Software as a service (SaaS)** [18] is a software distribution model in which a third-party provider hosts software centrally and makes it available to customers over the Internet.
- **Infrastructure-as-a-service (IaaS)** [19] it is an immediate computing infrastructure that is provisioned and managed over the Internet. It avoid the expense and complexity of purchasing and managing your own physical servers and other data center infrastructure, allowing the client to focus solely on the development of software solutions.
- **Platform as a service (PaaS)** [20] is a category between between the two above in which cloud computing services are responsible for the creation and maintenance of a software infrastructure. That way customers can focus on developing and managing applications and leave aside the maintenance of the infrastructure. message-passing.

Some examples of popular cloud based frameworks are briefly summarized below:

- **Amazon Web Services IoT⁶** allows companies to use Amazon Web Services to meet IoT needs by combining device communications, security measures, and a number of versatile application program interfaces (APIs) with Amazon’s robust AWS cloud.
- **Predix⁷** is an open-source platform for both internal and external industrial applications and includes authorization and authentication protocols at both the device and application level for security.
- **Microsoft’s Azure IoT Suite⁸** enables companies to implement a cloud-based IoT solution with minimal development. The suite offers device SDKs to connect devices to the cloud and transmit operational data and statistics.
- **IBM Cloud⁹** is a versatile and modular platform that allows companies interested in IoT deployments to employ a fully cloud-hosted, managed IoT platform. Bluemix can connect to existing devices or gateways through either the MQTT or HTTP protocols and transmit data in real time from a remote site to the cloud.

Use in INTER-IoT

The knowledge and use of cloud-based frameworks in INTER-IoT has a twofold purpose:

⁶ <https://aws.amazon.com/iot>.

⁷ <https://www.ge.com/digital/predix>.

⁸ <https://azure.microsoft.com/suites/iot-suite/>.

⁹ <https://www.ibm.com/cloud>.

- To integrate systems based on these cloud-based frameworks with the INTER-Layer infrastructure.
- To build a cloud-based framework to manage the different layers and the platform nodes connected to them.

2.2.3 REST-Based Frameworks

The term REST [21] is used to describe any interface between systems that directly uses HTTP to obtain data or indicate the execution of operations on the data, in any format (XML, JSON, etc.) without the additional abstractions of pattern-based message exchange protocols. Every REST resource is identified uniquely by an URL, which is operated upon by a subset of HTTP commands: get, post, put and delete. This is a major advantage of REST, as HTTP is the universal protocol of communication over the Internet. Another advantage is its scalability and independence. The protocol used in RESTful API makes the server user interface and data storage work independently. Thanks to this the software can be scaled without much difficulty. It also provides technology independence. REST can be used regardless of the type of language or technology being used in a project. In terms of security, we can protect REST using HTTPS instead of HTTP. Some outstanding examples of REST implementations are:

- **Apache Livy**¹⁰ is a service that enables easy interaction with an Apache Spark cluster over a REST interface. With it one can submit Spark jobs or snippets of Spark code, one can retrieve results synchronously or asynchronously, and also manage SparkContext.
- **Django REST framework**¹¹ is a third representative of REST-based frameworks. It is a toolkit for building web-based APIs, which are self-describing, list API endpoints, describes allowable operations on each and presents them as hypermedia controls that it sends in responses.

Use in INTER-IoT

As described above, this approach is suitable for the INTER-IoT project and is the main mechanism to expose the layers APIs and the global API of the framework. As REST philosophy [22] builds upon representations being completely at the mercy of the client, INTER-FW would not become bloated with addition of new platform types, new devices, new brokers, security features, etc. Existing nodes could be not only easily accessed, but also managed.

¹⁰ <https://livy.incubator.apache.org>.

¹¹ <https://www.django-rest-framework.org/>.

2.2.4 Reactive Streams Based Frameworks

Reactive Streams¹² is an initiative to provide a standard for asynchronous stream processing with non-blocking back pressure. In an asynchronous system, the handling of data streams, especially data whose volume is not predetermined (e.g. “live” data) requires special attention. The main objective is not to overload the destination stream and for this it is necessary to control the source of the data.

The Reactive Streams initiative standard defines a set of interfaces for handling streams in a reactive manner and gives a detailed specification of their intended behaviour. The main idea behind is the “back-pressure” mechanism, based on the Publish-Subscribe model. The Reactive Streams (RS) standard offers the following interfaces:

- **Publisher<T>** is the source of data for Subscriber(s). It offers a possibly unbounded sequence of data elements of type T.
- **Subscriber<T>** is an entity responsible for managing the subscription to a Publisher and handling the incoming data of type T. It also copes with error conditions and subscription cancellation.
- **Subscription** represents the lifecycle of a Subscriber subscribing to a Publisher.
- **Processor<T, R>** represents a reactive stream processing stage which is a Subscriber <T> and, at the same time a Publisher <R>.

The RS standard specification, although presented in the form of just four simple interfaces turns out to be quite complex and demanding at the implementation level. The treatment of asynchronous communication combined with back-pressure and proper error handling mechanisms is not as simple as it may seem at first. Fortunately, the existing libraries and frameworks that support the RS standard offer DSLs which make creating and transforming reactive streams much easier and intuitive.

Some examples of RS frameworks are listed in the following lines:

- **RxJava**¹³ is a lightweight (single-jar) library supporting reactive programming in Java. It offers and advocates functional programming techniques, including in-mutability and statelessness. RxJava is essentially a JVM implementation of the Reactive Extensions (ReactiveX), which builds upon the standard Observer pattern.
- **Reactor Core**¹⁴ similarly to RxJava, is a small (single-jar), highly focused library directly implementing the Reactive Streams standard, adding many useful stream transformation-building mechanisms.
- **Akka**¹⁵ is a toolkit and runtime for building highly concurrent, distributed, and resilient message-driven applications on the JVM. Akka utilizes the Actor Model of concurrency.

¹² <https://www.reactive-streams.org/>.

¹³ <https://github.com/ReactiveX/RxJava>.

¹⁴ <https://github.com/reactor/reactor-core>.

¹⁵ <https://akka.io/>.

Use in INTER-IoT

Concepts of “stream of data” and “stream processing” seem very natural and fundamental for any Internet of Things application. Also, the principles of “reactivity” undeniably apply to the realm of IoT. The popularity/acceptance of the Reactive Streams standard increases, and will probably continue to do so in the future. Solutions for many popular data sources/stores are available at the moment and new ones are being created. Hence, a framework offering Reactive Streams as a tool/concept is well worth considering in the case of INTER-IoT.

For a successful application within the INTER-FW, however, a framework definitely needs to provide more than just plain Reactive Streams support. Any INTER-IoT deployment will surely be distributed and will run on a (cloud) cluster. Therefore, the INTER-FW should also provide some flexible, high-level mechanisms for defining/composing such deployments. A carefully designed DSL would probably be very helpful in this respect. Both Reactor and Akka Streams constitute a part of comprehensive frameworks—Spring Framework and Akka respectively. RxJava, on the other hand, is a targeted library offering “just” an implementation of Reactive Streams standard, together with some tooling.

Taking the above into account, we should probably narrow our choice to Reactor versus Akka Streams, or rather Spring versus Akka. Both frameworks provide comprehensive sets of tools for building distributed applications, and support microservice architecture. Spring has an undeniably longer history, but Akka is definitely a mature project as well. Both frameworks have vibrant user communities, and are actively maintained. Out of the two Akka seems to be more coherent and compact in design. It also offers high-level DSLs which INTER-FW might successfully utilize/extend.

Some of the principles of reactive programming are successfully used in the design and implementation of MW2MW layer (INTER-MW), although without following a specific framework.

2.2.5 SOA-Based Frameworks

The concept of Service-Oriented Architecture [23] is a software design where services can be accessed by other software components using a communication protocol over the network. These services implement protocols that describe how to send and parse message using description metadata. A service can be defined as a software unit piece with a functionality that can be accessed remotely and acted upon and updated independently, these services have four main properties:

1. It logically represents a business activity with a specified outcome.
2. It is self-contained.
3. It is a black box for its consumers.
4. It may consist of other underlying services.

The SO-Architecture is more related on how to compose an application by integration of distributed, separately-maintained and deployed software components.

It is enabled by technologies and standards that make it easier for components to communicate and cooperate over a network, especially an IP network.

Java OSGi¹⁶ technology is the most representative specification to implement software frameworks and application in a service-oriented architecture. Some of the most representative implementation examples are:

- **Equinox**,¹⁷ **Concierge**,¹⁸ **Apache Felix**¹⁹ or **Karaf**²⁰ as runtime framework and service platforms OSGi-based on a module design that allows developers to implement an application in a bundles structure using the common services infrastructure they provide.
- **Swordfish**²¹ is an open source SOA framework intended for applications ranging from enterprise environments to embedded systems. Its features as a SOA runtime platform that leverages three popular projects: Service Component Architecture (SCA) as common programming model and assembly description format, Java Business Integration (JBI) as a common messaging model, and Open Services Gateway initiative (OSGi) as the basis of the runtime platform.
- **WSO2 Carbon**²² the core platform of the middleware, also based in OSGi, that allows components to be dynamically installed, started, stopped, updated and uninstalled, and it eliminates component version conflicts, creating a service oriented structure for an enterprise middleware.

Use in INTER-IoT

INTER-FW and some interoperability layers (gateway, INTER-MW) implementations are partially or completely distributed using services for each functionality of the framework and communicating each other by communication protocols. This is useful in the scalability and extendibility of the framework due to the modularity that this architecture provides. It also presents some caveats, as for instance that the components need to rely on the underlying network, so this has to be strongly configured for resilience.

As INTER-FW has to access all the components of INTER-Layer, it can implement different services to be consumed for each one of the clients on each interoperability solution. hence, each tier of INTER-Layer exposes an API (e.g. REST API) to connect with specific services within the framework where an added value is also implemented. This added value is an environment where future developer can implement a new service included within INTER-FW to be used by one or more interoperability solutions. So that, the modularity that Service Oriented Architecture provides facilitates this extension.

¹⁶ <https://www.osgi.org/>.

¹⁷ <http://www.eclipse.org/equinox/>.

¹⁸ <http://www.eclipse.org/concierge/>.

¹⁹ <https://felix.apache.org/>.

²⁰ <https://karaf.apache.org/>.

²¹ <https://wiki.eclipse.org/Swordfish>.

²² <https://wso2.com/products/carbon/>.

3 Framework Design and Implementation

The solutions provided by INTER-Layer are specific for each layer. INTER-FW is proposed to manage all these solutions from a higher and more abstract global point of view, by providing mechanisms, tools and helper content to properly manage the Layer Interoperability Infrastructure (LIIs) and the Interoperability Layer Interfaces (ILIs). To this end, each INTER-Layer solution will provide an API to be consumed by INTER-FW in order to manage the content and configuration located within the solution. Additionally, the INTER-FW API will expose together with its own extra value API calls, the APIs provided by each layer as a suite for future applications to use not only the API provided by a single solution but several solutions combined [24].

The following diagram depicts the components taking part in the INTER-FW administration and configuration framework. For simplicity, some components that share the same interactions have been represented in one single box. Colours have been applied to identify the main functional groups (Fig. 3).

The functional groups are: (i) web application components, which follows the classical design pattern model-view-controller. It includes all components related to the data storage, organization and visualization for the end-user; (ii) security, whose components intend to avoid the access to anonymous or unregistered users. It also

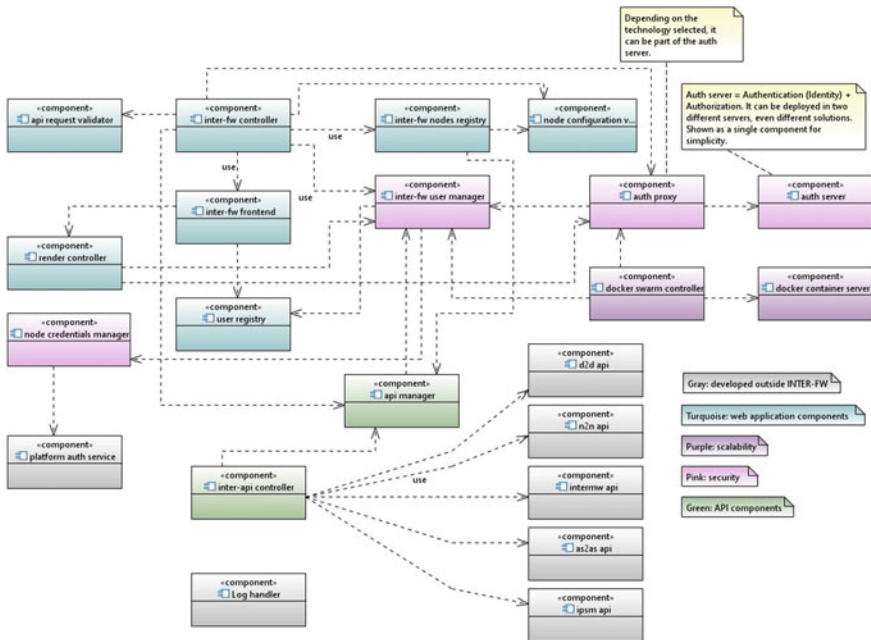


Fig. 3 Diagram of INTER-FW administration and configuration framework

prevents the unauthorised access to data; (iii) API management, which components perform operations related to the global API. In the following subsections a more detailed explanation of the design rules follow to develop the components of these groups as well as the technologies leveraged are presented; (iv) scalability, which based in containerization technologies, contributes to enable the growing of connected devices and users to the INTER-FW; and (v) extensibility, which is based in mechanisms that allow adding new functionalities or improvements of existing services.

3.1 *Web Application*

The target of the web application is to unify the monitoring, configuration and management of many IoT components (sensors, network elements, gateways, platforms ...) of the different interoperability layers in a single environment. It provides, through a Graphic User Interface (GUI), a single management place for all the interoperability layers developed in INTER-IoT, as well as a user management tool for authentication and authorization control. After performing an exhaustive analysis of the requirements and use cases that the application of the GUI had to support, inputs to the designers of the INTER-Layer solutions were provided so INTER-FW could be seamlessly integrated with them. The application consisted of the following components:

- **INTER-FW front-end:** the collection of views that are rendered to the user (main view, platforms, gateways, network, services, semantics, configuration and users).
- **INTER-FW controller:** this component performs the classical operations of a controller: it process and forward data requests to the model, ask for fields, credentials and permission validation, etc.
- **API request validator:** it checks the fields, verbs and paths of the API requests before sending them.
- **User registry:** a store of the INTER-IoT users.
- **Nodes registry:** a storage point for the registered nodes: platforms, gateways, services, etc. Each entry contains the need information to univocally identify the node.
- **Node configuration validator:** it checks that the configuration submitted for each node is correct, before calling the associated API.

Both the front-end and the back-end (which includes some of the aforementioned components) of the application were designed, including the security aspects of the web application and the credentials brokerage needed to manage the security and privacy of the connected platforms. It delegates a great part of its features in the LIIs. However, to bring practical features that may eventually let the INTER-IoT to understand and manage the concepts of multilayer interoperability of IoT Platforms, some back-end operations were needed. In general, these operations include

the features of serving, transforming and persisting the data of the application, so the INTER-Layer components that were affected by these operations were modified accordingly. The back-end was developed with Node.js²³ as web application framework, MongoDB²⁴ to store unstructured data, and Mongoose²⁵ for object modelling. On the other hand, the front-end is the presentation layer where users can see and interact with content in a user-friendly interface. It is usually developed as a mixture of Hypertext Markup Language (HTML), Cascading Style Sheet (CSS), JavaScript (JS) and ancillary libraries. In this case, Vue framework²⁶ was used for developing the interface, which simplifies the process of building user interactive interfaces.

INTER-FW web console structures each one of its modules (see Sect. 5) depending on the architecture and components of each layer solution, showing the structure of each layer and information about the components managed by them (devices, network elements, platforms, applications, etc.). More specifically:

- D2D: the containers running the virtual gateways instances are managed with the web application.
- N2N: the topology of the network is visualized through the application and the configuration of the switches, including QoS, are done within the web application.
- MW2MW: the different platforms connected to INTER-MW are shown. The instance of INTER-MW is unique.
- AS2AS: different containers run one instance of AS2AS for each user.
- DS2DS: the IPSM instance is fully manageable from the web console, including adding/removing channels, alignments and ontologies.
- Cross-Layer: all cross-layer functionalities related with the interaction of different layer solutions will be available through the FW (e.g. utilization of the IPSM by the AS2AS environment).

3.2 Security and Privacy

Security is a key aspect [25] to take in account when more than one solution and more than one client access at the same system. One of the main problems with security is that there are many point were data are gathered (devices, sensor, tags, etc.) and not all of them have the enough capacity or processing power [26] to implement trustable security mechanisms to ensure the privacy and authenticity of the provided data. For this reason, the upper layers have to make up for this flaws and take the leading role related with implementing security [27, 28]. This security can be defined in two key points: trust and privacy. Trust addresses several parts of a system:

²³ <https://nodejs.org/en/>.

²⁴ <https://www.mongodb.com/>.

²⁵ <https://mongoosejs.com/>.

²⁶ <https://vuejs.org/>.

- Device Trust: Need to interact with reliable devices.
- Processing Trust: Need to interact with correct and meaningful data.
- Connection Trust: Requirement to exchange the right data and only with the right service providers.
- System Trust: Desire to leverage a dependable overall system. This can be achieved by providing as much transparency of the system as possible.

On other side, privacy encompasses the sensible information managed by an IoT system that can jeopardize a company, the system itself or the confidentiality of data from individuals [29]. Since INTER-IoT is composed of several layers and modules, the security implementation cannot be monolithic. INTER-FW provides a portal to access several layers of the interoperability architecture, and each one of them has to be isolated and just communicate through the official mechanisms and channels. It aims at providing global and open platform-level interoperability among heterogeneous IoT platforms coupled through specifically developed ILIs. In this sense, the security-awareness and implementation in this component has to take into account those particular aspects and depict an architecture that encloses them and creates a reliable security framework across the different layers of the global INTER-IoT solution [30].

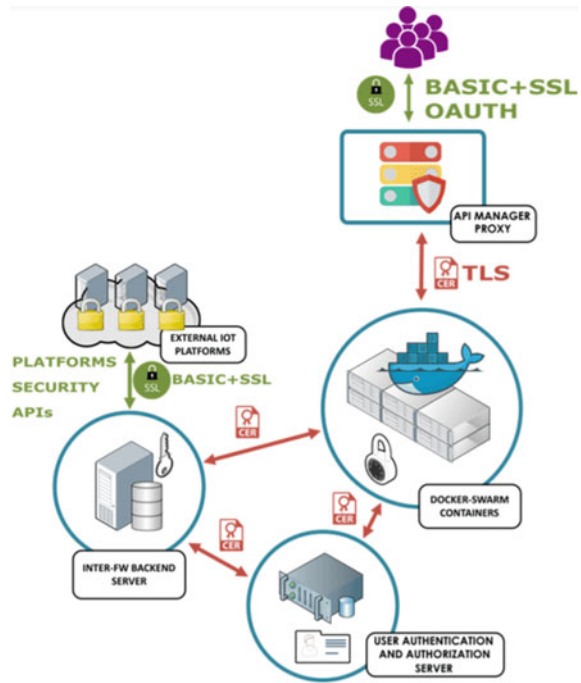
Each layer will be responsible of ensuring the integrity and encryption (if necessary) of the data it manages as well as the data exchanged with other components and with INTER-FW. Nevertheless, the authentication mechanism will be centralized in order to have a coherent access over all components of an INTER-IoT deployment. An INTER-IoT deployment will interact with one or more IoT Platforms and those IoT Platforms could implement their own security mechanisms. For that reason, when a platform is registered in INTER-FW, it will also store (encrypted) every authentication detail needed to interact with that IoT Platform. The following figure explain how this security architecture is implemented.

As it can be observed on the figure, there are security-specific components that, brought together with previously mentioned agents, comprise the whole design of the privacy and security model:

- External IoT Platforms: These are the external IoT platforms that will be configured and available to the rest of the layers
- INTER-FW Backend Server: This is then component where INTER-FW will be installed and deployed. It is also the frontend for the management and configuration of the INTER-IoT deployment. Regarding security aspects, all the authentication credentials and security specific data to interact with the external IoT Platforms will be stored under the Cryptex library.
- User Authentication Server: This server will be responsible to handle all authentication related mechanisms. It will interact with the INTER-FW backend server to provide the configuration and management points of the authentication integrated with the INTER-FW frontend, but its main purpose is to act as a centralised point of authentication for all the other layers deployed in the docker-swarm²⁷

²⁷ <https://docs.docker.com/engine/swarm/>.

Fig. 4 INTER-FW security management



containers. For this reason, the WSO2 Identity Server²⁸ has been chosen as the centralised authentication server. It provides single sign-on and identity federation capabilities, multifactor authentication, management of users, groups and roles, monitoring and auditing and multiple connectors and libraries for easy integration (Fig. 4).

- Docker-Swarm Container: This is the container in which the components of different layers will be deployed and managed. Regarding security and authentication, each layer will be responsible for implementing their security mechanisms in the framework and language chosen for each component. In order to have a coherent and centralised authentication mechanism, each layer will have their connector to the User Authentication Server to take advantage of all the authentication capabilities that provides.

3.3 REST API Management

API management [31] is a process that encompasses publishing, documenting and overseeing application programming interfaces (APIs) in a secure, scalable envi-

²⁸ <https://wso2.com/identity-and-access-management/>.

ronment. It allows organizations that publish their APIs to monitor the interfaces' lifecycles and also make sure that needs of both internal and external developers, as well as applications that are using their APIs, are being met. It thus provides core competencies for ensuring successful API usage in developer engagement, business insights, analytics, security and protection.

3.3.1 API Management Functionalities

Software that implements API management typically provides the following functions:

- Automation and control of connections between the API and applications that use it.
- Ensuring consistency between multiple API implementations and versions.
- Traffic monitoring from individual applications.
- Wrapping of the API into security procedures and policies, thus protecting it from misuse.
- Memory management and caching mechanisms to improve application performance.
- Sandbox environment for developers. Coupled with good API documentation, it enables potential customers to try the API out.

Nowadays, APIs are less dependent upon conventional Service-Oriented Architectures (SOA) and more on lightweight JSON and REST services. It is however possible to convert existing SOAP, JMS or MQ interfaces into RESTful APIs with JSON content.

3.3.2 API Management Components

Typical components of API management systems are:

- Gateway: a server that acts as an API frontend, which receives API requests and passes requests to the backend. It then passes the responses back to the requester. It can modify the requests and responses on the fly, and it can also provide the functionality to support authentication, authorization, security, audit and caching.
- Publishing tools: a collection of tools that API providers use to define APIs, for instance using the OpenAPI or RAML specifications. They also generate API documentation, manage access and usage policies, and can as well be used for testing and debug purposes.
- Developer portal/API store: a community site that enables user's access to documentation, tutorials, sample code, software development kits, and so forth. Here users can also manage subscription keys and obtain support from the API provider and the community.
- Reporting and analytics: a collection of tools that monitor API usage and load.
- Monetization service: a service that monetizes API usage.

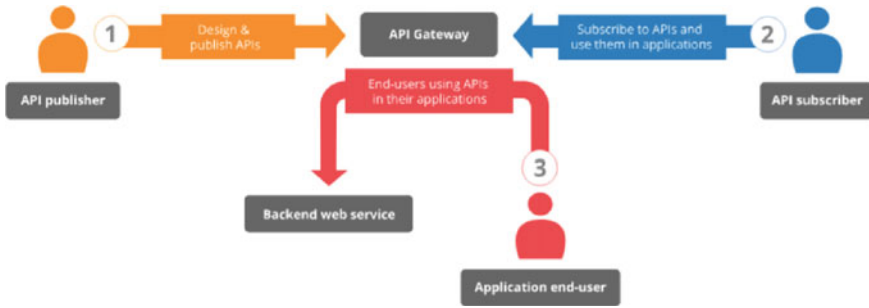


Fig. 5 REST API manager structure

3.3.3 API Management Solutions

An API management solution could be implemented in three different ways. The first one is as a proxy, where the API manager “sits” between the API and API’s user, processing all the traffic between these two. In this way, API manager can implement caching and protection of customer’s API from traffic spikes. However, a proxy also drives the cost up, as well as introduces additional privacy and latency issues. Apigee²⁹ and Mashery³⁰ provide API management solutions that work as a proxy. An API management solution could also work as an agent, that integrates itself with user’s server without acting as a middleman between the API and the user. API managers that work as agents typically do not suffer from latency issues and dependency on third parties, through which all the traffic passes. On the other hand, features like caching are not easy to implement. 3scale³¹ offers a product of this type. Some API managers work as hybrids, incorporating both a proxy and an agent. For example, the proxy could be used for caching, and agent for authentication. Apigee and 3scale are moving towards this solution (Fig. 5).

Many vendors offer their own API management solutions, as for instance Azure API management,³² Oracle,³³ IBM,³⁴ WSO2³⁵ or Red Hat (3scale). Majority of solutions are implemented as proxies, while others are hybrid. Most of them are directed towards SMEs, except for WSO2’s API manager, Microsoft’s Azure API manager and 3scale, which are also suitable for enterprises. CA Layer API manager is unique, as it has advanced support for mobile applications, while WSO2 API

²⁹ <https://cloud.google.com/apigee/>.

³⁰ <https://developer.mashery.com/>.

³¹ <https://www.3scale.net/>.

³² <https://azure.microsoft.com/es-es/services/api-management/>.

³³ <https://www.oracle.com/middleware/technologies/api-manager.html/>.

³⁴ <https://www.ibm.com/es-es/cloud/api-connect>.

³⁵ <https://wso2.com/api-management/>.

manager is open source. Apart from WSO2, among other open source API managers one can find Kong³⁶ and Tyk.³⁷

Different API managers cover different requirements. Some API managers, such as IBM's, Microsoft's and Oracle's, are part of a product or service, while others, such as WSO2's, are free. Some cannot even be deployed on premises, such as Microsoft's Azure. INTER-IoT aims at making use of an open source API manager, highly customizable and feature-full. Out of the handful open source API managers, WSO2 is the one selected for INTER-IoT as fulfilled all these necessities.

3.3.4 API Description

API specification and documentation is a key factor for technology adoption [32, 33], regardless whether the software is being used internally or by third-parties. With the growth of publicly available APIs de facto standards and tools have been emerging. Thus, some top specification formats used are OpenAPI Specification (Swagger³⁸), RAML³⁹ and API Blueprint.⁴⁰ This section introduces briefly each of them showing use examples and possible applications within the domain of INTER-IoT. In their most basic definition, API specification tools generate HTML and CSS code to display API methods, parameters, values, requests, responses, code samples, and more.

The OpenAPI Specification is both a specification format and a framework for generating documentation, server and client API code. Because of its wide adoption, Swagger is becoming a standard for API specification. Swagger-enabled API allows one to get interactive documentation, client SDK generation and discoverability. Swagger is language agnostic and uses JSON notation to describe the API documentation. Swagger is open source and is widely supported by the developers' community. It has a strong ecosystem providing tools, code generators, etc.

3.4 Scalability

The scalability of the components is achieved through the use of Docker containers [34]. Docker is software container platform that encapsulates applications to run and manage them side-by-side in isolated containers to obtain better performance and compute density. These containers can communicate with each other through a Docker network specifying the direction and port, and the Docker tool can handle the lifecycle of the containers in a way that this packages of software run isolated on

³⁶ <https://konghq.com/kong/>.

³⁷ <https://tyk.io/>.

³⁸ <https://swagger.io/specification/>.

³⁹ <https://raml.org/>.

⁴⁰ <https://apiblueprint.org/>.

a shared operating system being started, ran or stopped when needed. Despite other virtualization methods or machines, containers do not build a full operating system, instead only libraries and settings required to make the software work as needed.

3.4.1 Components Containerization and Clusterization

The solutions implemented in INTER-Layer should be wrapped then in Docker containers so that their lifecycles are managed in the framework. Docker-swarm provides a tool to observe and manage a group of Docker nodes as if it was only one cluster, so that one can start several related Docker containers at the same time and treat them as a whole. A swarm is a cluster of Docker nodes where the solutions are deployed, with a CLI and an API to set commands to manage the swarm nodes (initialize them, joining a running container, etc.). This tool is used to clusterize different solutions from the same layer (e.g., several virtual gateways) and manage them as an ensemble. The Dockerization of INTER-FW also comes with some security reinforcement, since only authorized users/roles can access to the different containers or cluster of containers.

3.5 Extensibility

Each INTER-Layer component has several extensibility mechanisms that allow adding new functionalities or improvements to existing services. This will inevitably lead to changes in REST API calls exposed by single layers. INTER-API provides API Lifecycle Management and Versioning, so that every change is typically deployed as a prototype for early promotion. After a period of time during which the new version is used in parallel with the older versions, the prototyped API can be published and its older versions can be deprecated.

With updates to OpenAPI definitions of the Unified INTER-IoT REST API, every change is documented and presented to prospective users in a standard format adopted by a majority of modern software development frameworks. In INTER-IoT pilots and tests, two different deployments have been used: (i) Self-standing API Manager with build-in identity management, and (ii) integration with the WSO2 Identity Manager. In principle, WSO2 API Manager can be integrated with several User Store types, such as LDAP, Active Directory and custom realms.

3.5.1 Identity Server Extensibility

The extensibility of the Identity Server is achieved through several extension points:

- Creation and application of new XACML policies.

- Aggregation of new authenticators and connectors (new identity providers) for identity federation. This could allow users accessing INTER-IoT without having an INTER-IoT account, by leveraging the Identity Server to control and restrict the most sensitive parts of the system. A possible scenario of this extension could be a deployment of INTER-IoT by a public service that allows all users (e.g., the citizens in a Smart City) to access read-only information about different publicly owned IoT platforms operating in an area.
- The INTER-IoT use of the Identity Server has a third expansion point through Entitlement Mediators. An Entitlement Mediator intercepts requests and evaluates actions performed by users against an XACML policy. Identity Server can be used as XACML Policy Decision Point (PDP), where the policy is set. This is also explained in the official documentation, and consists essentially in adding authorization points to the Identity Server for protecting endpoints in the INTER-API and other resources.

4 INTER-API Solution

As explained in Sect. 3.3.2, the solution selected to implement INTER-API [35] is WSO2 API Manager. This API Manager has been deployed as a Docker image in order to facilitate an integrated cloud deployment with the other INTER-FW components. The installation included the compulsory product registration, the Docker image deployment and the authentication with the WSO2 account. In addition, the appropriate URLs and API gateway endpoints have been configured. The creation of a unified API access through the WSO2 API manager consisted in several steps.

- Firstly, to create an API design document for each INTER-Layer component. It is provided through Swagger definitions.
- Afterwards, hose definitions are analyzed through a REST best-practice approach and different naming conventions among INTER-Layers.
- A unified API interface is then defined, with mapping to the corresponding endpoints of the backend systems.
- A unified OpenAPI definition is created, in addition to a “mediator” module that maps the APIs. The user should then subscribe to the APIs through the API subscription web GUI.

The main types of API users with their corresponding set of access and management privileges are (i) INTER-FW core users (users with full access to all INTER-FW features), and (ii) INTER-FW frontend users (users with restricted set of access rights necessary to execute API calls). In addition, a set of access policies, using SAML definitions, are being developed as part of the pilots’ specific requirements.

INTER-API is documented considering a Swagger-JSON format. The INTER-Layer APIs are accessed through an instance of the WSO2 API manager that acts as the main entry point for API management, access and usage. A set of OpenAPI definitions has been delivered for each layer in order to expose their respective

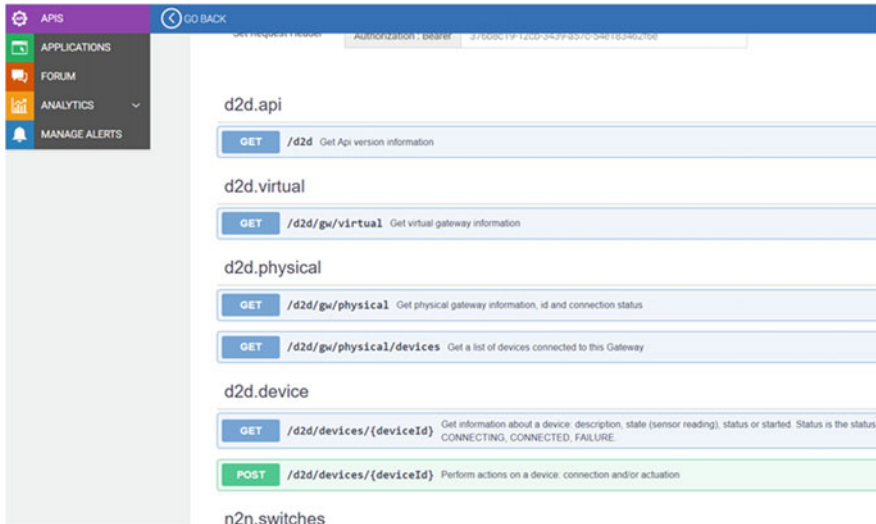


Fig. 6 INTER-API screen shoot

components. These have been integrated in the WSO2 API manager and published to the INTER-IoT API Store. There, users can register and gain access to the INTER-IoT deployment (Fig. 6).

5 INTER-FW Solution

INTER-FW provides a complete visual environment to enable development, configuration and management of interoperable IoT Platforms. The result is presented as a web application capable of controlling multiple aspects of interoperability at different layers. INTER-FW is the world's first visual framework that allows controlling multiple IoT platforms in a single user interface. It provides platform, device, network and service management for scenarios with heterogeneous IoT deployments, while protecting data sovereignty with personal and industrial information in a secure and safe way. The main functionalities covered by the framework are:

- To control all the interoperability layers. In a single screen, all the interoperability means are managed visually with full configurability.
- Comprehensive REST API available for all layers. INTER-API allows controlling all the interoperability aspects making easy building an application over heterogeneous IoT platforms.
- Fine-grained (up to device level) authorization of operations over interoperability layers. Complete control over the API and framework operations by the platform owner to keep sovereignty of the data (Fig. 7).

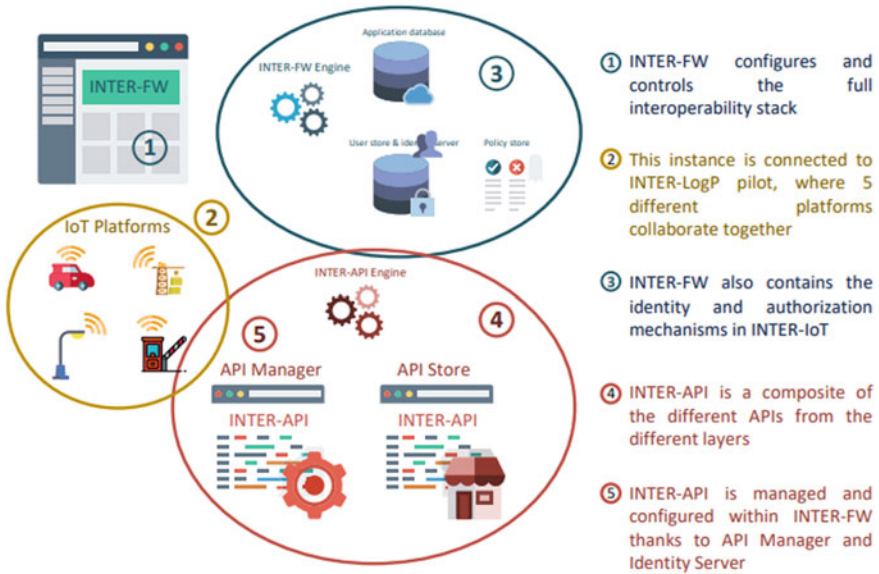


Fig. 7 INTER-FW set-up

An actual instantiation of the INTER-FW solution is presented. As previously explained, apart from unifying the monitoring, configuration and management of different IoT components (sensors, network elements, gateways, platforms, etc.) in a single environment, the web console of INTER-FW also includes key cross-layer management functions, such as configuration of authentication and/or authorization. As it can be seen in Fig. 8, it has eleven different tabs providing different features:

- Devices: all the devices connected to INTER-IoT are represented. Although useful for user experience purposes, tasks related to the registry or elimination of sensors are performed at gateway level.
- Gateways: provision of functionalities at the gateway level. Both physical and virtual gateways can be operated via the application.
- Network: management of virtual networks, QoS and networking rules.
- Platforms: instantiation, configuration and management of IoT platforms.
- Services: management of nodes, flows and instances of the IoT platforms applications and services interoperability solution.
- Semantics: definition of alignments, consultations with the semantic repository and basic operations upon the supported ontologies.
- Policies: utility to define security XML policies for fine-grained authorization.
- API Management: links to the INTER-API API Manager, managing level of access, lifecycle versioning and monitoring, among other tasks.
- Users management: a tool to manage the INTER-IoT users.
- Configuration: configuration of the INTER-FW too.

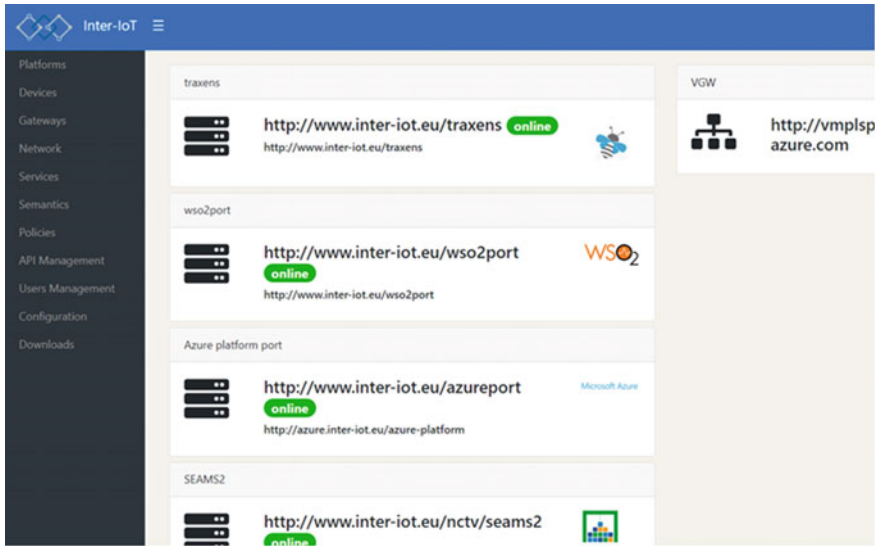


Fig. 8 INTER-FW screen shoot

- Downloads: This tab helps exporting information from INTER-FW to INTER-CASE tool (tool explained in Chap. 7). A web service has also been provided.

The target users of this product are:

- System integrators.
- Multiple IoT platform owners: typically public authorities and big corporations.
- Curious techies.

An overview of the impact that the use of the framework has provided during the development of the project's pilots is shown in the following statistics:

- In a typical deployment of 3 different platforms, an integrator can save up to 24% of configuration time.
- Resources in a single view can reduce 46% errors in configuration and management of devices.
- Cloud deployment increase by 62% the service availability.
- A common API reduces in 48% the boilerplate.
- Less dependency with manufacturers leads to 20% savings in licenses and subscriptions.

Finally, the main benefits obtained using INTER-FW are:

- To avoid vendor-lock: use one tool to control multiple IoT Platforms.
- To provide visual management features for platforms that lack of them (FIWARE, WSO2, OneM2M).

- To manage different interoperability layers in a single stop.
- To develop applications that leverage data from multiple heterogeneous platforms.
- To build applications based on a single API avoiding extra boilerplate.

6 Conclusions

INTER-FW offers the main features to configure, administer and manage heterogeneous IoT platforms in scenarios where interoperability is a key factor. These solutions have been tested and validated in the three INTER-IoT pilots (INTER-LogP, INTER-Health and Cross-domain Pilot) as well as in the Open Calls that joined the project in 2017. A full coverage of the use cases, including mechanisms to expand and adapt to specific interoperability scenarios, has been achieved. Different interoperability layers (D2D, N2N, MW2MW, DS2D2 and AS2AS) have been integrated through a unified management framework and API access.

Taking as starting point the approach of this framework, the goal is to publish some related theoretical works in the future. From the technical point of view, INTER-FW offers a valuable cloud-based integration product. This framework is an interesting complement for all the IoT platforms or projects without GUI and management capabilities. Finally, the API Manager is a key factor to the monetization of cloud-based services.

References

1. Schneider, M., Hippchen, B., Abeck, S., Jacoby, M., Herzog, R.: Enabling IoT platform interoperability using a systematic development approach by example. In: 2018 Global Internet of Things Summit (GIoTS), pp. 1–6 (2018)
2. Fortino, G., Savaglio, C., Palau, C.E., Suarez, J., de Puga, M., Ganzha, M.P., Montesinos, M., Liotta, A., Llop, M. (eds.): Towards multi-layer interoperability of heterogeneous IoT platforms: the INTER-IoT approach. In: Integration, Interconnection, and Interoperability of IoT Systems, pp. 199–232. Springer International Publishing, Cham (2018)
3. Legner, C., Lebreton, B.: Preface to the focus theme section: ‘business interoperability’ business interoperability research: present achievements and upcoming challenges. *Electron. Mark.* **17**(3), 176–186 (2007)
4. Amadeo, M., Campolo, C., Iera, A., Molinaro, A.: Named data networking for IoT: an architectural perspective. In: 2014 European Conference on Networks and Communications (EuCNC), pp. 1–5 (2014)
5. Pileggi, S.F., Palau, C.E., Esteve, M.: Building semantic sensor web: knowledge and interoperability. In: Proceedings of the International Workshop on Semantic Sensor Web—Volume 1: SSW (IC3K 2010), pp. 15–22. INSTICC, SciTePress (2010)
6. INTER-IoT H2020 Project: D4.3. Initial Reference IoT Platform Meta-Architecture and Meta Data Model INTER-Interoperable IoT Framework Model and Engine v1, Oct 2017. <https://files.inter-iot.eu/deliverables/accepted/D4.3%20-%20Interoperable%20IoT%20Framework%20Model%20and%20Engine%20v1.pdf>
7. Pasetti, A.: Software Frameworks and Embedded Control Systems, vol. 2231. Springer (2003)

8. Mnkandla, E.: About software engineering frameworks and methodologies. In: AFRICON 2009, pp. 1–5 (2009)
9. Ofoeda, J., Boateng, R., Effah, J.: Application programming interface (API) research: a review of the past to inform the future. *Int. J. Enterp. Inf. Syst. (IJEIS)* **15**(3), 76–95 (2019)
10. Nwana, H.S.: Software agents: an overview. *Knowl. Eng. Rev.* **11**(3), 205–244 (1996)
11. Dorri, A., Kanhere, S.S., Jurdak, R.: Multi-agent systems: a survey. *IEEE Access* **6**, 28573–28593 (2018)
12. Bellifemine, F., Bergenti, F., Caire, G., Poggi, A.: JADE—a Java agent development framework. In: *Multi-Agent Programming*, pp. 125–147. Springer (2005)
13. Aiello, F., Fortino, G., Guerrieri, A., Gravina, R.: Maps: a mobile agent platform for WSNs based on Java sun spots. In: *Proceedings of the ATSN* (2009)
14. Fortino, G., Garro, A., Russo, W.: Enhancing JADE Interoperability through the Java-based Interoperable Mobile Agent Framework. In: *2007 5th IEEE International Conference on Industrial Informatics*, vol. 2, pp. 1071–1077 (2007)
15. INTER-IoT H2020 Project: D3.3. Methods for Interoperability and Integration Final, June 2018. <https://files.inter-iot.eu/deliverables/accepted/D3.3%20-%20Methods%20for%20Interoperability%20and%20Integration%20Final%20Version.pdf>
16. Mahmood, Z., Saeed, S.: *Software Engineering Frameworks for the Cloud Computing Paradigm*. Springer (2013)
17. Antonopoulos, N., Gillam, L.: *Cloud Computing*. Springer (2010)
18. Buxmann, P., Hess, T., Lehmann, S.: Software as a service. *Wirtschaftsinformatik* **50**(6), 500–503 (2008)
19. Bhardwaj, S., Jain, L., Jain, S.: Cloud computing: a study of infrastructure as a service (IAAS). *Int. J. Eng. Inf. Technol.* **2**(1), 60–63 (2010)
20. Keller, E., Rexford, J.: The “platform as a service” model for networking. *INM/WREN* **10**, 95–108 (2010)
21. Fielding, R.T.: Fielding dissertation: Chapter 5: Representational state transfer (rest). http://www.ics.uci.edu/~fielding/pubs/dissertation/rest_arch_style.htm (2000)
22. Masse, M.: *REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces*. O’Reilly Media, Inc. (2011)
23. Jerstad, I., Dustdar, S., Thanh, D.V.: A service oriented architecture framework for collaborative services. In: *14th IEEE International Workshops on Enabling Technologies: Infrastructure for Collaborative Enterprise (WETICE’05)*, pp. 121–125 (2005)
24. Belsa, A., Sarabia-Jacome, D., Palau, C.E., Esteve, M.: Flow-based programming interoperability solution for IoT platform applications. In: *2018 IEEE International Conference on Cloud Engineering (IC2E)*, pp. 304–309, Orlando, FL, Apr 2018. IEEE (2018)
25. Frustaci, M., Pace, P., Aloï, G., Fortino, G.: Evaluating critical security issues of the IoT world: present and future challenges. *IEEE Internet Things J.* **5**(4), 2483–2495 (2018)
26. Altolini, D., Lakkundi, V., Bui, N., Tapparelo, C., Rossi, M.: Low power link layer security for IoT: implementation and performance analysis. In: *2013 9th International Wireless Communications and Mobile Computing Conference (IWCMC)*, pp. 919–925 (2013)
27. McGraw, G.: Software security. *IEEE Secur. Priv.* **2**(2), 80–83 (2004)
28. Broring, A., Zappa, A., Vermesan, O., Främling, K., Zaslavsky, A., Gonzalez-Usach, R., Szmeja, P., Palau, C., Jacoby, M., Zarko, I.P., Sour-sos, S., Schmitt, C., Plociennik, M., Krco, S., Georgoulas, S., Larizgoitia, I., Gligoric, N., Garcia-Castro, R., Serena, F., Orav, V.: *Advancing IoT Platform Interoperability*. River Publishers, The Netherlands (2018)
29. Acquisti, A., Brandimarte, L., Loewenstein, G.: Privacy and human behavior in the age of information. *Science* **347**(6221), 509–514 (2015)
30. Fortino, G., Liotta, A., Palau, C., Gravina, R., Manso, M. (eds.): *Towards Multi-layer interoperability of heterogeneous IoT platforms: the inter-IoT approach* (2017)
31. Fremantle, P., Kopecký, J., Aziz, B.: Web API management meets the Internet of Things. In: *The Semantic Web: ESWC 2015 Satellite Events*, pp. 367–375. Springer International Publishing, Cham (2015)

32. Shi, L., Zhong, H., Xie, T., Li, M.: An empirical study on evolution of API documentation. In: *Fundamental Approaches to Software Engineering*, pp. 416–431. Springer, Berlin, Heidelberg (2011)
33. Giménez, P., Molina, B., Palau, C.E., Esteve, M.: SWE simulation and testing for the IoT. In: *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 356–361 (2013)
34. Hasselbring, W., Steinacker, G.: Microservice architectures for scalability, agility and reliability in E-Commerce. In: *2017 IEEE International Conference on Software Architecture Workshops (ICSAW)* (2017)
35. INTER-IoT H2020 Project: D4.6. Interoperable IoT Framework API and Tools, Model and Engine v2, June 2018. <https://files.inter-iot.eu/deliverables/accepted/D4.6%20-%20Interoperable%20IoT%20Framework%20API%20and%20Tools%20v2.pdf>

INTER-Meth: A Methodological Approach for the Integration of Heterogeneous IoT Systems



Giancarlo Fortino, Raffaele Gravina, Wilma Russo, Claudio Savaglio, Katarzyna Wasielewska, Maria Ganzha, Marcin Paprzycki, Wiesław Pawłowski, Paweł Szmeja, and Rafał Tkaczyk

Abstract The Internet of Things (IoT) is a jeopardized ecosystem in which heterogeneity is intrinsic at all levels, from physical devices to communication protocols till high-level application semantics. The absence of IoT standards increases the complexity of integration and interoperability among heterogeneous platforms. This generates a strong demand for proper methodologies in order to fully support the development of heterogeneous, yet interoperable, IoT systems. To fill this gap, in this chapter the INTER-METH engineering methodology is presented. Developed in the context of the European H2020 INTER-IoT project, INTER-METH supports the integration of heterogeneous IoT platforms from the analysis to the maintenance phase. Its abstract and instantiated process schema are described, with particular focus on the analysis and design phases that are fundamental drivers of the whole integration process. Relevant interoperability design patterns, the building blocks of the design phase, will be discussed. The chapter also presents the INTER-CASE tool associated to the methodology which is useful to guide integrator designers in properly following the INTER-METH workflow. Finally, the chapter shows the pro-

G. Fortino (✉) · R. Gravina · W. Russo · C. Savaglio
University of Calabria, Rende, Italy
e-mail: g.fortino@unical.it

K. Wasielewska · M. Ganzha · M. Paprzycki · P. Szmeja · R. Tkaczyk
Systems Research Institute, Polish Academy of Sciences, Warsaw, Poland
e-mail: katarzyna.wasielewska@ibspan.waw.pl

M. Ganzha
e-mail: maria.ganzha@ibspan.waw.pl

M. Paprzycki
e-mail: marcin.paprzycki@ibspan.waw.pl

P. Szmeja
e-mail: pawel.szmeja@ibspan.waw.pl

R. Tkaczyk
e-mail: rafal.tkaczyk@ibspan.waw.pl

W. Pawłowski
Faculty of Mathematics, Physics and Informatics, University of Gdańsk, Gdańsk, Poland
e-mail: wieslaw.pawlowski@ug.edu.pl

© Springer Nature Switzerland AG 2021

C. E. Palau (eds.), *Interoperability of Heterogeneous IoT Platforms*, Internet of Things,
https://doi.org/10.1007/978-3-030-82446-4_7

posed methodology and its tool in action, with the practical integration of BodyCloud and UniversAAL platforms adopted in the INTER-Health pilot of the INTER-IoT project.

1 Introduction

The IoT domain is stimulating the interest of academia and industry, thus generating considerable yet uncoordinated research efforts. As a result, high degree of heterogeneity is characterizing IoT scenarios at all levels, obstructing interoperability of IoT devices, systems, and applications [1] and generating major technological and business development issues. For instance, lack of interoperability causes impossibility to plug third-party IoT devices into existing IoT platforms, makes very hard the development of IoT applications exploiting multiple platforms, discourages the adoption of IoT technology, increases maintenance costs, reduces reusability of existing technical solutions, and as a natural consequence generates user dissatisfaction. To tackle the rapid proliferation of poorly interoperable IoT systems, the H2020 EU-funded project INTER-IoT aimed to design, implement and evaluate methods and tools to enable voluntary interoperability among different IoT platforms by using a bottom-up approach [2]. Indeed, in the absence of global IoT standards, the INTER-IoT results allows IoT stakeholders to design open IoT devices, smart objects, services, and platforms leveraging on the existing ecosystem, and bring them to market quickly. In particular, INTER-IoT is based on hardware/software tools (INTER-LAYER) granting multi-layer interoperability among IoT system layers (i.e., device, networking, middleware, application service, data and semantics), on frameworks for open IoT application and system programming and deployment (INTER-FW), and on a full-fledged engineering methodology for IoT platforms integration (INTER-METH) complemented by its Computer Aided Software Engineering (INTER-CASE) tool.

INTER-METH, that is probably the most peculiar feature of the INTER-IoT project, is the subject of this chapter. INTER-METH 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. INTER-METH, whose relevance is emphasized by the absence in the literature of any other full-fledged methodology for the integration of IoT platforms, is based on a workflow composed by six phases: Analysis, Design, Implementation, Deployment, Testing and Maintenance that are in turn divided into sub-tasks. Each phase generates work-products that are inputs for the successive phase(s). It is worth noting that INTER-METH is domain agnostic by definition, intended to be extensible and customizable, and it can be associated to any specific IoT systems integration approach.

The remainder of the chapter is organized as follows. Sect. 2 provides an overview of available methodologies for integrating IoT systems and making them interoperable. Section 3 presents several interoperability patterns that represent the building blocks of the integration design phase. Section 4 introduces, phase by phase, the

INTER-METH methodology, with particular emphasis on its INTER-IOT instantiation, with the goal of showing how the integration process between two heterogeneous IoT platforms/systems can be concretely carried out by exploiting the INTER-METH guidelines and INTER-IoT products. Section 5 describes INTER-CASE, the tool associated to the methodology which is useful to guide integrator designers in properly following the INTER-METH workflow. Ultimately, in Sect. 6, a practical use of INTER-METH and its INTER-CASE tool is presented by showing the analysis and design workflows applied for the integration of the two platforms adopted in the INTER-Health pilot. Final remarks conclude the chapter.

2 Background

The aim of this section is to provide background to the research and design of methodologies for interoperable IoT systems and their integration. State-of-the-art (SotA) analysis includes discussion of: (i) the definition of methodologies, (ii) relevance of the reviewed methodologies in the IoT domain, and (iii) characteristics of such methodologies useful for defining INTER-METH. Specifically, they are organized in two main categories: (a) General-Purpose Software Engineering Methodologies (see Sect. 2.1) and (b) More Specific Methodologies for System Integration (see Sect. 2.2). For each category, we provide an overview and finally an overall analysis towards INTER-METH definition, i.e. the characteristics of surveyed methodologies useful to support the definition of INTER-METH.

2.1 *Software Engineering Methodologies*

In software engineering, a software development methodology (also known as a system development methodology, software development life cycle, software development process, software process) is a splitting of software development work into distinct phases (or stages) containing activities with the intent of better planning and management. It is often considered a subset of the systems development life cycle. The methodology may include the pre-definition of specific deliverables and artefacts that are created and completed by a project team to develop or maintain an application. Common methodologies include waterfall, prototyping, iterative and incremental development, spiral development, rapid application development, extreme programming and various types of agile methodologies.

The Waterfall development methodology [11] is a step-by-step guide to the development of software systems. Briefly, it focuses on gathering the features of the final product, designing it, and then implementing it following that design. The term depicts the idea that only once a higher step in the process is complete (full of water), it will spill its results in the following step below itself. The classic Waterfall methodology is composed of five main steps, in the following sequence: (1)

Requirements gathering, (2) System design, (3) Implementation. (4) Verification, and (5) Maintenance. This engineering method is one of the first such methods used for the structured production of software solutions, and is still often used in several industries, though the so called ‘agile’ methods have displaced the Waterfall method in several areas, in particular in rapidly evolving systems.

The VOLERE methodology [12] helps to describe, formalize and track the project market analysis, requirements, use cases and scenarios in an explicit and unambiguous manner. VOLERE has been used by thousands of organizations around the world in order to define, discover, communicate and manage all the necessary requirements for any type of system development (e.g. software, hardware, commodities, services, organizational, etc.). The VOLERE methodology provides several templates to deal with the different techniques and activities that it includes.

Interestingly, several development methodologies are built around the concept of Agent and are known as Agent-Oriented Software Engineering (AOSE) methodologies.

Gaia is an AOSE Methodology [13] specifically tailored to the analysis and design of agent-based systems. It is intended to allow an analyst to go systematically from a statement of requirements to a design that is sufficiently detailed that it can be implemented directly. It is worth pointing out that Gaia authors view the requirements capture phase as being independent of the paradigm used for analysis and design. For this reason Gaia does not deal with the requirements capture phase but it considers the requirements statement as an input for the methodology. Analysis and design can be thought of as a process of developing increasingly detailed models of the system to be constructed moving from abstract to increasingly concrete concepts.

Tropos [14] is an AOSE methodology strongly focuses on early requirements analysis where the domain stake-holders and their intentions are identified and analysed. This analysis process allows the reason for developing the software to be captured. The software development process of TROPOS consists of five phases: Early Requirements, Late Requirements, Architectural Design, Detailed Design and Implementation.

ELDAMeth [15] is a methodology specifically designed for the simulation-based prototyping of distributed agent systems (DAS). It is based on an iterative development process covering the modeling, coding and simulation phases of DAS. ELDAMeth can be used both stand-alone and in conjunction/integration with other agent-oriented methodologies which fully support the analysis and (high-level) design phases. The Modeling phase produces an ELDA-based MAS design object that is a specification of a MAS fully compliant with the ELDA MAS meta-model (MMM). This design object can be produced either by (i) the ELDA-based modeling which uses the ELDA MMM and the ELDATool [16], a CASE tool supporting visual modelling and coding of ELDA-based MAS [33, 34], or by (ii) translation and refinement of design objects produced by other agent-oriented methodologies. The Coding phase produces an ELDA-based MAS code object which is a translation of the ELDA-based MAS design object carried out manually or automatically (by means of the ELDATool). The developed code could be also mapped onto heterogeneous MAS platforms [38]. The Simulation phase produces the Simulation Results

in terms of MAS execution traces and calculation of the defined performance indices that must be carefully evaluated with respect to the functional and non-functional requirements [39]. Such evaluation can lead to a further iteration step which starts from a new (re)modelling activity.

Other interesting AOSE methodologies include MaSE [17], Prometheus [18], MESSAGE [19].

General AOSE methodologies deal with providing engineering support to systems modeled as multi-agent systems. Thus, they could be used for general software development support (from analysis to implementation) for implementing or re-engineering software systems. Nevertheless, some of their methods could be generalized and reused to support integration of systems. For instance, TROPOS proposes a goal-oriented analysis that could be reused to elicit integration requirements among different components/part of systems/systems. In particular, we reused TROPOS to identify/refine integration goals among IoT platforms.

Agent-oriented methodologies are suitable for the development of distributed applications and systems in terms of multi-agent systems. They can be categorized in basic (e.g. Gaia, Message, TROPOS, MaSe, Prometheus) and simulation-based (e.g. ELDAMeth). However, they do not aim at supporting (hw/sw) distributed systems integration, thus they cannot directly support the definition of INTER-METH. Nevertheless, some of the techniques proposed by such methodologies could be reused as basic techniques for defining INTER-METH:

- Goal-oriented analysis (from TROPOS) to analyse integration goals;
- Agent-oriented domain conceptualization (from Gaia and ELDAMeth), to formalize integration requirements in the form of a high-level agent system design;
- Simulation-based validation (from simulation-based methodologies) to validate integration (i.e. the high-level agent system design) before its implementation.

2.2 *IoT Methodologies*

In the following, we analyse currently available IoT methodologies. It is worth noting, however, that they are still at an earlier stage of development with respect to methodologies presented in the previous section.

Despite a variety of research efforts that tackle different specific issues within an IoT systems development process, a full-fledged IoT engineering methodology is still missing. Several studies proposed domain specific best practices [5], guidelines, checklists, and templates. For instance, Slama et al. [20] and Collins [15] created a repository of technology-dependent solutions coming from the experience in the industrial/business world and specifically directed to the IoT makers and enterprises. In fact, they proposed reference architectures and guidelines to make specific purpose devices interoperable through abstraction data models and high-level software interfaces.

By means of different views, perspectives and metamodels, IoT-A aims to offer a unified approach to the development of IoT systems, in order to promote cross-domain interaction, to support interoperability and to reduce fragmentation within an IoT context. Notably, IoT-A introduced an Architecture Reference Model (ARM) [21] with the capability of generating architectures for specific systems. A reference model is, according to the OASIS [22] is “an abstract framework for understanding significant relationships among the entities of some 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”. Most of the indications provided by IoT-A have inspired AIOTI (Alliance for the Internet of Things) [28] particularly for the domain model. From IOT-A we re-used its functional architecture in INTER-METH Analysis Phase.

Zambonelli [23] proposed a software engineering methodology centered on the main general-purpose concepts related to the analysis, design and implementation phases of IoT systems and applications. Such concepts are used to identify the key software engineering abstractions as well as a set of guidelines and activities that may drive the IoT systems development. The envisioned methodology, however, lacks the definition of models and tools to represent different conceptual and software artifacts.

Fortino et al. [36] proposed the ACOSOMeth approach for the agent-oriented development of IoT systems [3, 37]. ACOSOMeth uses a model-driven development approach seamlessly covering the analysis, design and implementation phases. ACOSOMeth also enables the development of even complex IoT systems of systems [6, 7, 35].

Although the overviewed methodologies have been specifically defined for developing IoT systems totally or partially fulfilling the reference requirements for IoT systems development, they have not been devised for IoT systems integration and interconnection. Even though such methodologies have another scope, INTER-Meth took inspiration by borrowing the ideas of:

- meta-modeling approach that is typical of the model-drive development (MDD) approach;
- agent-oriented-like approach (see [23]) allowing to simplify the definition of integration requirements analysis;
- AIOTI [28] and IoT-A [24] meta-models to have reference IoT architectures during the integration process, in order to align the meta-models of the IoT systems/platforms to be integrated/interconnected or made interoperable to the reference meta-model.

In [25, 26], the addressed IoT interoperability using “model-driven development” tools and techniques. In particular, there are three key contributions:

1. Interoperability models are reusable, visual software artifacts that model the behavior of services in a lightweight and technology independent manner; these models are a combination of architecture specification and behavior specification and are based upon Finite State Machines (FSM);

2. A graphical development tool to allow the developer to create and edit interoperability models and to also execute tests to report interoperability issues;
3. The Interoperability monitoring and testing framework captures systems events (REST operations, middleware messages, data transfers) and transforms them into a model specific format that can be used to evaluate and reason against required interoperability behavior.

Hence, such model-driven development approach allows the developer to create, use and re-use “models of interoperability” to reduce development complexity in line with the following requirements to ensure interoperability is correctly achieved. Different stakeholders are defined in the engineering methodology:

- Interoperability testers create new IoT applications and services to be composed with one another;
- Application developers model the interoperability requirements of service compositions. They create interoperability models to specify how IoT applications should behave when composed.

This research discusses about interoperability of IoT services, systems, and (virtualized) devices and was taken in consideration in our CASE-driven integration methodology (see INTER-CASE) supporting the integration process of heterogeneous IoT platforms. In [27] System of Systems (SoS) integration is considered. SoS is defined as a “set of systems that are cooperating and interoperating while the different systems are simultaneously working as independent entities”. Authors proposed methods to improve the way in which an IT architect addresses the integration problem, focusing on how to select the best integration approach in SoS context depending on the features of the environment and systems to be integrated. Some design patterns from popular patterns catalogs are analyzed by the authors who proposed a process for creating SoS based on patterns as a central architectural concept.

2.3 An Analysis Toward INTER-METH

The analyzed methodologies and techniques directly address the issue of systems integration by adopting different approaches. Some of them are purposely related to IoT systems integration but they do not provide any systematic methodology that, starting from two or more systems to integrate, provides a clean process (along which tools for each phase) to support the integration.

Nevertheless, the Waterfall model can be used, after enhancement, to support INTER-Meth. In fact, the proposed INTER-Meth process is based on an iterative version of the waterfall model, as the basic waterfall is too static. Agent-oriented methodologies are suitable for the development of distributed applications and systems in terms of multi-agent systems. However, they do not aim at supporting (hw/sw) distributed systems integration, thus they cannot directly support the definition of

INTER-METH. Nevertheless, some of the techniques proposed by such methodologies could be reused as basic techniques for defining INTER-METH:

- Goal-oriented analysis (from Tropos) to analyse integration goals;
- Agent-oriented domain conceptualization (from Gaia and ELDAMeth), to formalize integration requirements in the form of a high-level agent system design;
- Simulation-based validation (from simulation-based methodologies) to validate integration (i.e. the high-level agent system design) before its implementation.

Regarding the overviewed IoT methodologies, they have been specifically defined for developing IoT systems totally or partially fulfilling the reference requirements for IoT systems development, but they are not devised for IoT systems integration and interconnection. Thus, even though some of the ideas on which they are founded could be reused, such methodologies (as the reviewed agent-oriented methodologies) have another scope. INTER-Meth could borrow mainly:

- the meta-modeling approach that is typical of the model-drive development (MDD) approach;
- The agent-oriented-like approach (see [23]) allowing to simplify the definition of integration requirements analysis;
- AIOTI [28] and IoT-A [24] meta-models to have reference IoT architectures during the integration process, in order to align the meta-models of the IoT systems/platforms to be integrated/interconnected or made interoperable to the reference meta-model.

3 Design Patterns for IoT Systems

With the proliferation of IoT artifacts (devices/platforms/systems/services) the need arises to analyze existing solutions from the software engineering perspective. Specifically, in the context of integration and interoperability new design patterns materialized and required to be analyzed and catalogued. Note that design pattern is understood as a general reusable solution to a problem that recurs repeatedly within a specific context in software design, whereas a pattern catalog is a collection of related patterns, subdivided into a (small) number of categories. Here, we outline the design patterns identified in INTER-IoT project either by defining them from scratch or extending some existing pattern. They address issues that can be extended beyond INTER-IoT project. For more detailed information INTER-IoT deliverable D5.1.

Since so far there have been no formal guidelines to IoT integration, in INTER-IoT we have decided to decompose the the problem into layers: D2D (Device-to-Device), MW2MW (Middleware-to-Middleware), AS2AS (Applications and Services-to-Applications and Services, DS2DS (Data and Semantics-to-Data and Semantics) and CROSS layer relating them. For each of these layers design patterns have been proposed and described using the following template:

- *Pattern name*—unique name of the pattern.

- *Inspired by*—name(s) of pattern(s) that a given one is based on/extends. In most cases, when pre-existing patterns did not fully solve specific problems, new patterns were created, extending existing ones.
- *Related patterns*—other patterns, related to the given one.
- *Intent (summary)*—short description of the goal behind the pattern and the reason for using it (an extension of the “Pattern name”, explaining its action/purpose).
- *Problem & Solution*—scenario that illustrates a problem and how the pattern solves it.
- *Applicability*—situations, in which the pattern is usable; context for the pattern.
- *UML representation*—structure of the pattern modeled with a UML diagram (mostly deployment and component diagrams).
- *Implementation*—extension of the “UML representation” property, i.e. description of realization and architecture.
- *Known uses*—an example usage of the pattern within the INTER-IoT pilot installation.

Fields: *UML representation*, *Implementation*, *Related patterns* and *Known uses* have not been included in what follows (find them in INTER-IoT deliverable D5.1).

3.1 D2D Patterns

The two following patterns are related to design on device-to-device layer. IoT Gateway Event Subscription describes an approach to data forwarding, whereas D2D REST Request/Response describes approach to communication on D2D layer.

IoT Gateway Event Subscription

Inspired by: (1) “Publish/Subscribe” (IoT Patterns: Design Patterns for Interaction), (2) “Publish-Subscribe Channel” (EIP: Messaging Channels), (3) “Facilitator” , and (4) “Proxy” (Agent Design Patterns: by Kendall).

Intent: D2D gateway allows data forwarding (any type). Flexibility in the D2D layer is achieved by decoupling a gateway into: a physical part that handles network access and communication protocols, and a virtual part dealing with remaining gateway operations and services. Note that, in this way, data providers (communicating within the network) and their identities (known to the virtual layer) can also be decoupled.

Problem and Solution: To provide interoperability between two heterogeneous IoT devices, the solution should establish bidirectional, asynchronous communication with the ability to publish, filter and consume data. Here, the IoT gateway is used as a subscription mechanism. It is an intermediary between IoT artifacts (in D2D communication, between two devices). It allows transmission of data generated by “sensors” to the destination, and asynchronous messaging between artifacts that interact with it. If required, the gateway should perform protocol conversion to enable communication. Senders of messages (publishers) do not program messages sent directly to specific receivers (subscribers). Instead, they publish them, using defined

classes, without knowledge of subscribers. Similarly, subscribers express interest in one or more classes and receive only messages of interest (without knowing publishers). Significant is the structure of the message, which should contain subscription information (e.g. message endpoint; see: “D2D REST Request/Response” pattern).

Applicability: Used within event-based communication, when asynchronous data is to be pushed/pulled to/from the gateway.

D2D REST Request/Response

Inspired by: (1) “Request-Response” (Reactive Patterns: Message Flow), (2) “Request-Reply” (EIP: Messaging Patterns), (3) “Request/Response” (IoT Patterns: Design Patterns for Interaction).

Intent: A request/response solution for gateway communication within the D2D layer.

Problem and Solution: IoT Gateway needs to communicate with IoT artifacts [9]. It should be accessible to authorized external elements to enable reception of information and execution of control and configuration orders. For example, the main goal of IoT ecosystems is to allow heterogeneous IoT artifacts to retrieve information. Thus, the artifacts’s middleware should be able to communicate with the IoT gateway to enable needed information flows. Thus, it is desirable to connect IoT artifacts (if possible) through a HTTP/REST API using the Request/Response pattern. This communication pattern allows message exchange, in which a requester (e.g. middleware or gateway) sends a request message to a replier system, which receives and processes the request (e.g. middleware or gateway), ultimately returning a message, in response.

Applicability: Used when communication between the middleware of an IoT artifact and the IoT gateway is performed through a REST API (middleware → gateway is typically based on Publish/Subscribe). Also, for management purposes, the gateway will expose a REST endpoint where configuration and management actions can be performed using the Request/Response patterns.

3.2 N2N Patterns

On network-to-network layer we have identified one pattern that addresses the problem of orchestration of different SDN network elements.

IoT Pattern for Orchestration of SDN Network Elements

Inspired by: (1) “Software-defined networking (SDN) orchestration”, (2) “Network virtualization (NV)”.

Intent: Monitoring and configuration of SDN elements (virtual-switches) with an orchestrator component (Controller) exchanging flow and control messages. To provide interoperability between different domains connected to a network or between different network topologies and/or configurations.

Problem and Solution: Domain-focus of IoT deployments isolates them from each other. One of approaches to interconnection is, instead of realizing it at the device/gateway level, to move it to upper layers. In particular, at the network layer, interoperability and exchange of information can be achieved by applying pattern that manages elements of the network that provide connection from different domains to the network itself. The pattern is used in development of virtual SDNs, where all elements are virtual resources, or instances, controlled within a central point, or orchestrator. N2N interconnection can then be performed through the SDN. Different networks (in different locations), can be virtually interconnected and belong to a single Virtual LAN. Thus, physical separation of networks becomes “invisible”, thus facilitating elastic definition of needed connectivity.

Applicability: Used when an IoT SDN is deployed, to enable its functionality. Allows total software control over network functions, and transparent N2N interoperability.

3.3 MW2MW Patterns

Patterns on middleware-to-middleware layer address the issues of components decomposition, communication infrastructure and messaging between IoT artifacts.

IoT Artifact’s Middleware Simple Component

Inspired by: (1) “Simple component” (Reactive Patterns: Fault Tolerance and Recovery).

Intent: Every IoT artifact is designed to perform (in full) a single task (single responsibility principle, where each class should have only one reason to change).

Problem and Solution: In complex systems with multiple functions, it may be necessary to have these functions performed by different components. Their responsibilities are to be divided recursively, until desired component granularity is reached. This enables testing, debugging and extending complex system more efficiently, simplifying all operations.

Applicability: Basic pattern that can be universally applied. Does not impose level of granularity to be achieved, but indicates that analysis should be performed in order to end up with the best component decomposition. Should be applied recursively, remembering to not to divide components too far, to avoid trivial ones.

IoT artifact’s Middleware Message Broker

Inspired by: (1) “Message Broker” (EIP: Message Routing), (2) “Broker” (IoT Patterns: Design Patterns for Interaction).

Intent: Facilitates passing messages between IoT artifacts.

Problem and Solution: In middleware, composed of several independent components, point-to-point connections should be avoided, as they result in multiple interfaces that expose operations of each component. Furthermore, having point-to-point interfaces complicates dynamic reconfiguration, matching of security constraints, and quality of service (QoS) requirements management. Message broker helps to

overcome those limitations by enforcing common messaging interface upon different components. This allows components to initiate interactions with other components, no matter their architecture and purposes. Each component communicates directly with the broker only, while within the broker, each component is represented with a logical name, making its internal operation hidden from other components. Crucial is the proper format of message, which consists of the payload and the label, storing information needed by the broker.

Applicability: Central Message Broker receives messages from multiple message producers, determines their destinations (message consumers), and routes them to channels specific for their destinations. Allows decoupling the sender from the destination and maintains central control over the flow of messages. This can be achieved through usage of topics, to which consuming components can: subscribe, and proceed to consume awaiting messages.

IoT Artifact's Middleware Self-contained Message

Inspired by: (1) “Self-contained message” (Reactive Patterns: Message flow), (2) “Messaging Metadata” (SOA Patterns).

Intent: Messages contains complete information needed for execution of a specific action.

Problem and Solution: Within middleware, messages should be “pure and complete” representations of events (or commands), regardless when they are to be interpreted. Each middleware component must always be able to extract from the message, stored there, complete information needed for its routing and interpretation, with only minimal data stored within the middleware components. Each message has distinct set of types associated with it. Each middleware component processes and routes messages based solely on these types. For each message that travels “downstream”, there can be a response that travels “upstream”. Such messages might, for example, include additional response message type. Matching messages that go downstream with responses that go upstream can be done through remembering and distinguishing different chains of messages (conversations).

Applicability: Allows middleware components to be “contextfree, storing only a minimal information needed for message processing and routing. Can be also employed when there is no need to reference past messages, except for responses, and even then, these are only semantically linked to original messages (could exist without original messages).

IoT Artifact's Middleware Message Translator

Inspired by: (1) “Message Translator” (EIP: Message Transformation), (2) “Data Format Transformation” (SOA Patterns).

Intent: Translation of messages to/from IoT artifact's middleware internal message format and platform's proprietary data models and data formats.

Problem and Solution: The purpose of the middleware is to pass information between different IoT artifacts. However, artifacts produce/consume messages in “their” formats and data models. Hence, to make sense of exchanged messages, they

have to be syntactically and semantically translated. A message translator enables conversion between proprietary data models and data formats, used by artifacts, and the internal data model and data format, used by IoT middleware components.

Applicability: Enables interoperability between different platforms without the need to introduce translations between every possible pair of platforms, i.e. translation into and out of the common INTER-IoT data model. Semantic translation from and into the internal message format is done by a dedicated IoT semantic translation component, while syntactic translation is completed in bridges to/from artifacts, as only they know the internal data syntax.

3.4 AS2AS Patterns

On applications and services-to applications and services layer design patterns address the issues of service composition, orchestration and discovery.

AS2AS Flow-based Service Composition

Inspired by: (1) “Flow-based programming”.

Intent: Generate execution flow that allows interoperation and composition of services from different IoT artifacts.

Problem and Solution: Pattern necessary to define execution flow that allows specific sequence of execution of multiple IoT services. Flow-based programming (FBP) defines applications and services as networks of “black box” processes, which exchange data across predefined connections by message delivery, where connections are specified externally to processes. Considered pattern allows creation of sequential execution flows using those services, thus allowing composition among different IoT services. Black boxes that represent IoT services can be linked by wiring the output of a service with the input of a different one (output messages from a service are routed to another service input). Thus, by wiring IoT services execution flow can be instantiated.

Applicability: Used in black box representation of IoT services to be interconnected through an FBP link, generating a flow.

AS2AS Service Orchestration

Inspired by: “Service Orchestration” (SOA Patterns).

Intent: To specify orchestration of services to facilitate interactions among different IoT services.

Problem and Solution: Cooperating, diverse, heterogeneous IoT artifacts involve huge number of different services that have to work together. Important is not only the message flow from point(s) to point(s) but also triggering necessary actions (during the flow). The common problem is that existing processes/actions are duplicated (not reused). This pattern allows union and orchestration of heterogeneous IoT services, creating a specific process. The main idea is to define a set of IoT nodes, i.e. services

and interfaces that run within the integrated platforms. Internal, central, element wires nodes necessary to handle the specific task and controls processes.

Applicability: Reuse of process fragments. Orchestration enables composition of IoT service workflows, based on services from IoT artifacts.

AS2AS Discovery of IoT Services

Inspired by: (1) “Discovery” (IoT Patterns: Design Patterns for Interaction), (2) “Enterprise inventory” (SOA Patterns).

Intent: Registering and claiming specific services, used by the artifacts (within the IoT ecosystem).

Problem and Solution: Multiple IoT services, from different IoT platforms, provide a wide range of functionalities that have to be discoverable, to be aware of them and to use them. This pattern enables registration of services (in a service catalog), in order to find them and (potentially) use through an AS2AS layer solution. Here, only registered services, indicating their associated features, can be discovered.

Applicability: Pattern for providing service interoperability via registration and service retrieval. Applied to services that run within the IoT ecosystem, and used by other IoT artifacts.

3.5 DS2DS Patterns

On data and semantics-to-data and semantics layer pattern address the problem of semantic translation, specifically how to persist translation rules and how to organize the translation process.

Alignment-based Translation Pattern

Inspired by: (1) “Message Translator” (EIP: Message Transformation), (2) “Data Model Transformation” and (3) “Metadata centralization” (SOA Patterns), (4) “Market Maker” (Agent Design Patterns).

Intent: Semantic translation of RDF messages exchanged between IoT artifacts, based on alignments (sets of correspondences) defined between artifacts’ ontologies.

Problem and Solution: Building the IoT ecosystem involves combining existing solutions, which (likely) belong to different owners and have been developed using different technologies (e.g. Web Services “combined with” a graph database, communicating using JSON messages, to exchange information with application that uses XML messages). Consequently, they differ both on syntactic and semantic level. Interoperation between artifacts should be achieved regardless of the underlying technology. Without loss of generality, we assume RDF message format, since other formats can be transformed to RDF. Semantics of messages is artifact specific (ontology can be natively supported, or semantics, e.g. expressed in XSD, can be lifted to an OWL ontology). Hence, for semantic interoperability, a method for defining correspondences should support mapping between specific URIs, parts of the RDF structure, transformations etc. The component implementing the translation

should provide interfaces to submit messages to be translated and publish translated messages.

Applicability: Providing semantic translation between RDF messages exchanged between heterogeneous IoT artifacts. Translation, based on one-to-one translation (alignment), should be possible to define for any two artifacts.

Translation with Central Ontology

Inspired by: (1) “Message Translator” (EIP: Message Transformation), (2) “Data Model Transformation” and (3) “Metadata centralization” (SOA Patterns), (4) “Market Maker” (Agent Design Patterns).

Intent: Semantic translation of RDF messages exchanged between IoT artifacts, where one involves central/common data model.

Problem and Solution: To provide common understanding in the semantic translation process a modularized central ontology can be created on the basis of IoT and domain ontologies. Here, a domain ontology is a conceptual model for a specific domain, e.g. transportation, health, etc. IoT ontology describes different IoT aspects, e.g. platforms, devices, sensors, services, etc. Central ontology should reuse/be based on existing standards (e.g. SSN, SOSA, SAREF, etc.). This approach is highly scalable: it is possible to add artifacts to the existing IoT ecosystem by instantiating translations with the central ontology (i.e. creation of alignments; see above). This approach requires less preparation/work, as only a single “point of joining” has to be instantiated. Furthermore, the long-term maintenance is simplified, as changes in a single platform require localized adjustments only. The component implementing the translation should provide interfaces to submit messages to be translated and publish messages after translation (realizable via an appropriate pattern, above).

Applicability: Providing semantic translation between multiple heterogeneous IoT artifacts that are to exchange RDF messages.

3.6 *CROSS-Layer Patterns*

IoT SSL CROSS-Layer Secure Access

The CROSS-Layer pattern addresses issue that is common to the IoT-based systems regardless of the layers, i.e. security.

Inspired by: (1) Security Patterns, (2) IoT Patterns: Design Patterns for IoT Security.

Intent: Ensuring security of interactions with external interfaces (i.e. APIs) of every layer of the IoT ecosystem.

Problem and Solution: As IoT architecture is composed by diverse layers, access to each of them, as well as interactions between them, must be secure. To ensure a sufficient level of security on each of the IoT layers, different security mechanisms can be implemented: authentication of credentials, use of authentication tokens, or Secure Sockets Layer (SSL). In an IoT ecosystem, layer access will be secured with the SSL that employs the IoT SSL pattern. Every IoT layer exposes a REST API that represents an external interface accessible to the outside actors, such as other IoT layers, users, or IoT artifacts. To enable use of such APIs to only allowed actors the

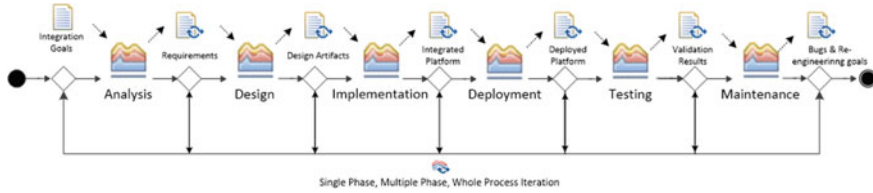


Fig. 1 INTER-METH abstract process schema

access is secured through SSL. REST APIs are accessible through a browser, which should provide a trusted certificate. Only after the establishment of a secure connection authentication through login will be allowed, to open the access to the layer API. Further operations on the layer API will be done using this secure connection. **Applicability:** Pattern applied in interactions of any actor with the IoT environment layer’s APIs. Access can also be done internally between pairs of different layers.

4 INTER-METH

In this section, we will briefly introduce the abstract process of our INTER-METH methodology. Then, we will describe in more details its INTER-IoT instantiation, with particular attention to the Analysis and Design phases.

4.1 INTER-METH Abstract Process

The engineering methodology INTER-METH aims at supporting the integration process of heterogeneous IoT platforms to obtain interoperability and support implementation and deployment of IoT applications on top. In this section, we introduce the abstract process of INTER-METH whose SPEM¹-based schema is shown in Fig. 1. The process is envisioned as iterative waterfall and is composed of six main phases, each of which is divided into tasks and produces workproducts that are inputs for the successive phase. In particular:

- *Analysis Phase* supports the definition the IoT platform (non-)functional integration requirements.
- *Design Phase* produces both artifacts (e.g., diagrams) and programming/management patterns to fulfill the elicited requirements and define the integration design.

¹ OMG, SPEM, and O. M. G. Notation. “Software and systems process engineering meta-model specification.” OMG Std., Rev 2 (2008).

- *Implementation Phase* drives the implementation of the design workproduct to obtain the full-working (hardware/software implemented) integrated IoT platform.
- *Deployment Phase* involves the support to the operating set-up and configuration of the integrated IoT platform.
- *Testing Phase* defines the performance evaluation tests to validate the integrated platform according to the functional and non-functional requirements.
- *Maintenance Phase* manages the upgrade and evolution of the integrated system.

In detail, at the Analysis Phase, on the basis of the Integration Goals (representing high-level integration requirements), each platform is analysed according to a shared reference architecture model. Functional and non-functional integration requirements of the platforms are hence formalized in a document whose format will be specified according to the specific instantiation of the abstract INTER-METH methodology (see Sect. 4). On the basis of the elicited requirements at the Analysis Phase, an initial design specification is produced for each layer and iteratively refined at the Design Phase. The final workproduct is a formalized specification containing the design of the integration of the IoT platforms/systems to be interconnected/integrated. This full-fledged design specification is actually implemented through multiple refinement steps at the Implementation Phase according to the integration specifications, aiming at obtaining the final integrated platform. The final workproduct is the integrated platform, that will be based on the specific IoT platforms/systems to be integrated/interconnected and that will be deployed according to deployment goals and requirements at the Deployment Phase. Hence, the deployed platform is set-up according to the defined configuration and run specifications. The integrated and deployed platform is then executed and validated through testing according to well-defined test cases at the Testing Phase: specifically, functional and non-functional test cases (previously defined to respectively validate functional and non-functional requirements) are executed by the platform and results collected according to formalized analysis documents. Finally, to maintain and allow the evolution of the integrated IoT platform, the Maintenance Phase aims at identifying both bugs and evolution points, activities which imply to totally/partially re-execute the integration process.

4.2 INTER-METH Instantiated on INTER-IoT

The INTER-METH abstract methodology has been customized for the INTER-IoT approach [2] with the aim of showing how the integration process between two or more heterogeneous IoT platforms/systems can be concretely carried out by exploiting the INTER-METH guidelines and the two INTER-IoT products INTER-LAYER and INTER-FW. INTER-LAYER is a set of interoperability solutions dedicated to each specific INTER-IoT architectural layer (Device-to-Device D2D, Networking-to-Networking N2N, Middleware-to-Middleware MW2MW, Application&Services-to-Application&Services AS2AS, Data&Semantics-to-Data&Semantics DS2DS).

Thanks to its layered approach, INTER-LAYER makes the interoperability flexible and allows it to reflect the interests/needs of the stakeholders, integrators or application developers. INTER-FW, instead, allows the development of new applications and services by customizing INTER-METH and exploiting INTER-LAYER. Indeed, INTER-FW is a global framework for programming and managing interoperable IoT platforms by means of specific programming interfaces and interoperability tools for every architectural layer. As in Sect. 4.1, for each phase of the instantiated process, an overall description is reported along with a brief description of the performed tasks and obtained workproducts.

- *INTER-IoT-based Analysis Phase*: given two or more heterogeneous IoT platforms to be integrated according to certain Integration Goals, these are analyzed according to the INTER-IoT RA that is, as previously reported in Sect. 2, based on IoT-A [4]. Integration requirements are then defined for each architectural layer (according to some iterative tasks that will be elicited in Sect. 4.2.1) and formalized in the INTER-Goal Oriented Model (INTER-GOM) as final workproduct.
- *INTER-IoT-based Design Phase*: for each layer, on the basis of the elicited requirements reported in the INTER-GOM, an initial INTER-IoT Design Pattern is produced and iteratively defined by exploiting the INTER-LAYER product (D2D, N2N, MW2MW, AS2AS, DS2DS), thus producing five instantiated INTER-IoT Design Patterns. On the basis of these Patterns, a full-fledged specification is iteratively produced as workproduct by incorporating also the CROSS-LAYER and INTER-FW INTER-IoT Design Patterns. These patterns are integration solutions structured according to certain templates (describing pattern's main properties, e.g., pattern name, its intent, its known uses) and formalized through XML files providing domain-specific guidelines to be exploited for driving the INTER-IoT-based Implementation Phase.
- *INTER-IoT-based Implementation Phase*: it consists in the (i) configuration of the components of the Integrated Platform by means of the INTER-FW; (ii) potential extension of the components of the Integrated Platform (e.g., a new functionality enabled by the integration needs to be implemented); and (iii) implementation, in terms of software bridges connected to INTER-LAYER, of the INTER-IoT based design patterns. The final output workproduct is the INTER-IoT-based Integrated Platform (if needed, an ontology alignment for finding correspondences among entities and sub-structures from different ontologies can be performed). Indeed, at this point, the heterogeneous IoT platforms are integrated according to the INTER-IoT approach, thus obtaining interoperability among them and allowing implementation and deployment of IoT applications on top of them.
- *INTER-IoT-based Deployment Phase*: the following six tasks have to be performed for the deployment of the Integrated Platform according to deployment goals and requirements: (i) Platform Configuration, which aims to instantiate and deploy an IoT Platform Middleware in the INTER-FW; (ii) Gateway Configuration, focused on the device to device interoperability, addressed in the scope of INTER-IoT in the D2D Layer; (iii) Networking Configuration, which achieves network interoperability via network virtualization to support the needs of the N2N Layer [10];

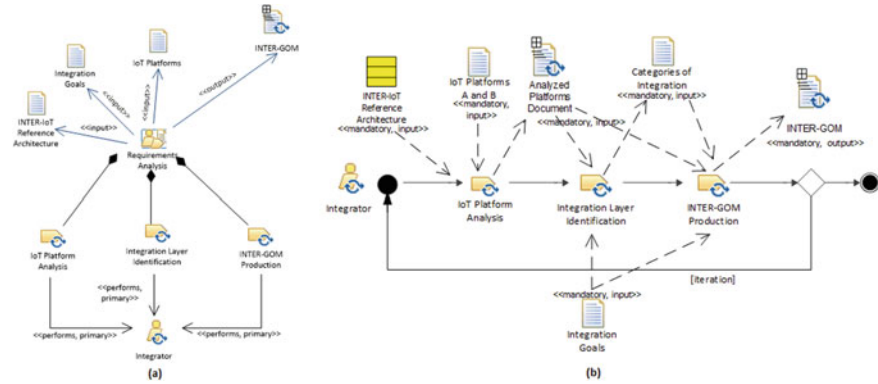


Fig. 2 The INTER-IoT-based analysis phase: **a** requirement analysis activity overall description and **b** its workflow

- (iv) Application Services Configuration, which includes a graphic tool for service orchestration, namely the underlying interoperability mechanism for AS2AS Layer;
- (v) Semantics Configuration, which manages all the processes and mechanism of DS2DS Layer enabling ontologies interoperability;
- (vi) User Management Configuration, to configure and manage the users of the INTER-FW and their authorized access to the IoT resources connected in INTER-IoT.
- *INTER-IoT-based Testing Phase*: the integrated, configured and deployed platform is executed and validated through well-defined test cases. In particular, to determine if the requirements of a specification are met, Factory Acceptance Testing (FAT) task is performed by simulation whereas Site Acceptance Testing (SAT) task takes place after integration at the customer site and tests if the solution has been correctly integrated into the customer’s environment. The Defect Reporting task, instead, is in charge of identifying and reporting issues emerged in FAT and SAT tasks as well as implementing, integrating and testing the related solutions. FAT and SAT documents are the output workproducts of such phase.
- *INTER-IoT-based Maintenance Phase*: it allows the maintenance and evolution of an integrated and already deployed IoT platform. It first identifies bugs and/or evolution points at each layer as well as at cross-layer of the integrated platform (i.e., at INTER-LAYER) and framework level (i.e., at INTER-FW), and then at actually develops the required changes.

4.2.1 INTER-IoT-Based Analysis Phase

The INTER-IoT-based Analysis Phase involves the Requirement Analysis activity, which is described in Fig. 2a in terms of tasks, roles, and workproducts as well as in terms of workflow in Fig. 2b. The Requirement Analysis comprising the following

three main tasks that are performed by the integrator: the IoT Platforms Analysis, the Integration Layer Identification, and the INTER-GOM Production.

The *IoT Platforms Analysis Task* receives two or more heterogeneous IoT systems as inputs and, according to the INTER-IoT RA, produces the Analyzed Platforms Document. This step allows heterogeneous IoT platforms with even notably different architectures to be compared by means of a common set of architectural solutions and building blocks. In particular, INTER-IoT RA is depicted in Fig. 3 and consists in the following Functional Groups (FG):

- Service Interoperability FG, supporting the AS2AS interoperability through the definition and execution of new compound services that make use of already existing services in the underlying IoT Platforms; it uses services from different IoT Platforms and create new services based on them.
- Semantics FG, addressing the challenges related to semantic interoperability of IoT Platforms; it provides support for the Service Interoperability FG, the Platform Interoperability FG and the Device Interoperability FG.
- Platform Interoperability FG, interacting with the different IoT Platforms to be interconnected; it is the responsible for accessing the IoT Platforms.
- Device Interoperability FG, addressing the challenges of making legacy devices and non-real IoT Platform interoperable with other IoT Platforms and systems.
- Device Access FG, that is responsible for offering a common interface to services and virtual entities that represent and expose functionality of physical devices; it abstracts all the necessary functions for managing the devices and interacting with them.
- Management FG, considering all the functionalities to rule the interoperability among different IoT Platforms; it is responsible for initializing, monitoring and modifying the operation of the interoperability among IoT Platforms.
- Security FG is responsible for ensuring all the security aspects involved in the interoperability of IoT Platforms.

The *Integration Layer Identification Task* receives the Analyzed Platform Document and a set of Integration Goals as input. As extensively reported, INTER-IoT presents a layer-oriented solution for interoperability, to provide interoperability at any layer and across layers among different IoT systems and platforms. Although a layer-oriented approach is more challenging than an application-level approach, it has a higher potential to deliver tight bidirectional integration among heterogeneous IoT platforms, thus providing flexibility, modularity, higher performance, reliability, and security. This layer-oriented solution is achieved through INTER-LAYER and includes several interoperability solutions dedicated to specific layers. The INTER-LAYER architecture is reported in Fig. 4 and comprises the following six layers: (i) Device allows the seamless inclusion of novel IoT devices and their interoperation with already existing ones; (ii) Network(ing) aims to provide seamless support for smart objects mobility and information routing; (iii) Middleware enables seamless resource discovery and management system for the IoT devices in heterogeneous IoT platforms [8]; (iv) Application&Services enables the use of heterogeneous services among different IoT platforms; (v) Semantics&Data allows a shared interpretation of data and information among heterogeneous IoT systems and data sources,

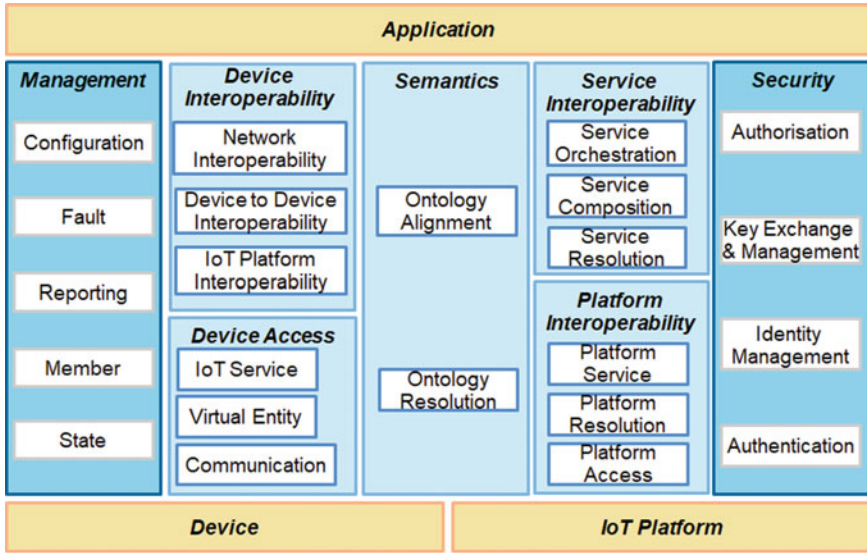


Fig. 3 INTER-IoT reference architecture schema

achieving semantic interoperability; and (vi) CROSS-LAYER covers and guarantees non-functional aspects that must be present across all layers: trust, security, privacy, and quality of service (QoS).

Finally, the *INTER-GOM Production Task* receives the Analyzed Platform Document, the Integration Goals and the Categories of Integration, and produces the INTER-GOM. This task can be iterated one or more time, thus obtaining the final INTER-GOM that will represent the formal requirements model and drive the INTER-IoT-based Design Phase. In particular, the INTER-GOM comprises the following components, according to its Metamodel depicted in Fig. 5(a): (i) the Analyzed Platform Document, which compares the platforms using the shared the INTER-IoT RA to identify their integration points; and (ii) one or more Integration Points IP [14], which put together parts of the platforms according to the CoI and one or more Requirements. Requirements represent the states to be achieved by the IP; they are progressively refined into intermediate goals, until the process produces actionable goals/tasks that need no further decomposition and can be executed. According to the Analyzed Platform Documents and the IP, the GOM is defined following an iterative procedure as shown in the activity diagram of Fig. 5(b).

After having analyzed IoT platforms/systems whose formal integration requirements are reported in INTER-GOM model, process is ready to move toward the INTER-IoT-based Design phase.

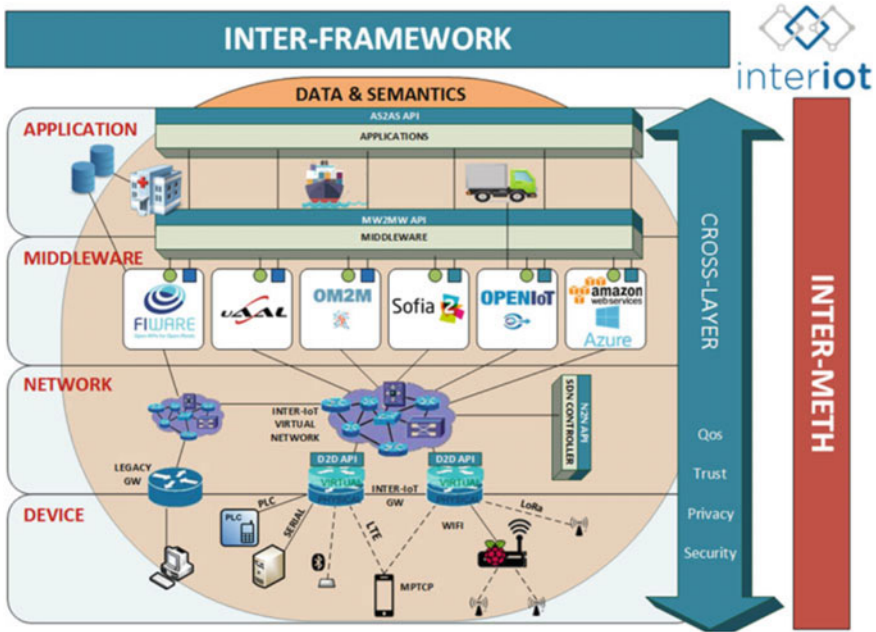


Fig. 4 The INTER-LAYER approach schema

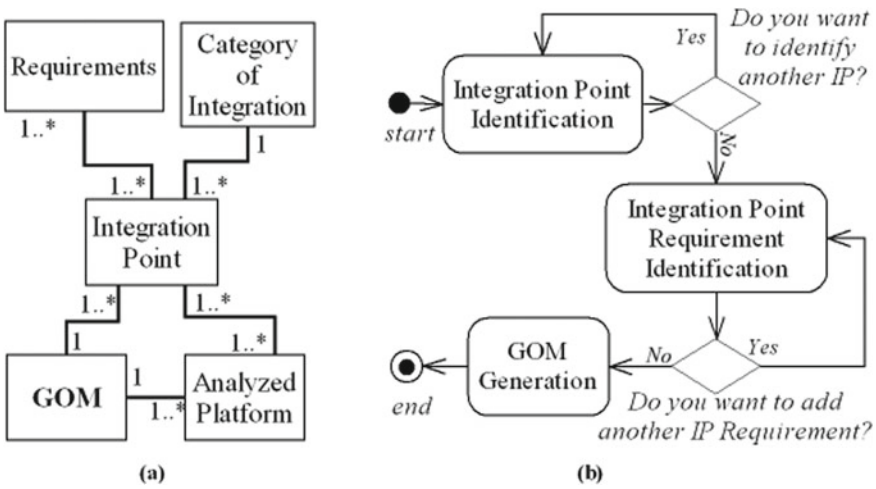


Fig. 5 a INTER-GOM metamodel and b UML activity diagram of INTER-GOM production

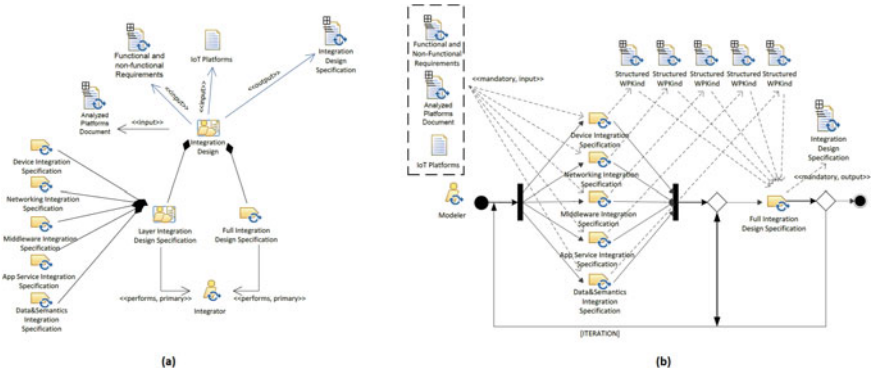


Fig. 6 The INTER-IoT-based design phase: **a** design activity overall description and **b** its workflow

4.2.2 INTER-IoT-Based Design Phase

Given two or more IoT platforms/systems whose formal integration requirements are reported in INTER-GOM model, a set of INTER-IoT Design Patterns have to be produced (see Sect. 3), by instantiation, on the basis of the available pattern templates (semi-instantiated patterns). For each layer, on the basis of the elicited requirements reported in the INTER-GOM, an initial INTER-IoT Design Pattern is produced and then usually iteratively refined. The Integration Design activity, which is the only main activity of the INTER-IoT-based Design phase, is subdivided into two sub-tasks that are performed by the Integrator according to the workflow depicted in Fig. 6: (i) INTER-IoT Layer Integration Design Specification and (ii) INTER-IoT Full Integration Design Specification. On the basis of the INTER-GOM and Analyzed Platforms Document, for each INTER-IoT Layer a layer integration specification is iteratively defined by exploiting the INTER-LAYER product. Such task will produce instantiated INTER-IoT Design Patterns (see Sect. 3), one for each INTER-IoT layer. Then, a full-fledged specification is iteratively produced by incorporating the CROSS-LAYER and INTER-FW INTER-IoT Design Patterns. In particular, INTER-IoT PATTERNS (or INTER-PATTERNS) are design patterns directly corresponding to the integration solutions already achieved in the WP3 (particularly, according to INTER-LAYER) and WP4 (particularly, according to the INTER-FW for contextualize solutions in different application domains, e.g. m-Health, Transportation and Logistics) and they aim at furnishing well-formalized domain-specific guidelines. Note that WP5 depends on WP3 and WP4 outcomes, while WP3 and WP4 are independent from WP5.

The work product of this phase is a set of instantiated INTER-IoT Design Patterns to be exploited for driving the implementation phase.

4.2.3 INTER-IoT-based Implementation to Maintenance Phases

The Implementation phase is the third step of our methodology; it takes in input two or more IoT platforms whose integration has been designed according to the instantiated INTER-IoT Design Patterns. The objective of this phase is to concretely integrate/interconnect the considered IoT platforms by physically implement the instantiated patterns according to successive refinement steps that involve to (i) configure the components of the Integrated Platform by means of the INTER-FW, (ii) extend the components of the Integrated Platform (e.g., a new functionality enabled by the integration needs to be implemented), and (iii) implement, in terms of software bridges connected to INTER-LAYER, the INTER-IoT based INTER-LAYER design patterns. The final output work-product is the INTER-IoT-based Integrated Platform. At this point, the heterogeneous IoT platforms/systems are integrated according to the INTER-IoT approach.

The Deployment phase is the fourth step of the methodology and follows the Implementation phase. The objective is the deployment of the integrated and implemented platform. The Integrated Platform is deployed according to deployment goals and requirements. In particular, the INTER-FW web framework is the entry point to the INTER-IoT Configuration and Management Framework (CMF). The INTER-IoT based Deployment activity, which is the only main activity of the Deployment phase, comprises six main tasks, described in the following. The Platform Configuration task deals with the deployment of complete IoT Platforms for interoperating them towards rich applications. Although the technical focus of this module is the deployment of interoperable middlewares of platforms, the whole content (i.e., the platform) has been assumed to the concept of ‘middleware’ since the IoT platforms are univocally bound to the concept of platform (there are few or none platforms without a middleware, while there are middlewares not linked with specific platforms). To instantiate and deploy an IoT Platform Middleware in the INTER-FW to enable middleware interoperability, the following steps are followed: A bridge of the platform to interoperate must be available. INTER-IoT provides a series of reference bridges. If the platform is not in the list of reference implementations, this must be done following the “developing new bridges” instructions that will be publicly available by the end of the project in the project site in GitHub. These instructions will extensively use the Software Development Framework of the project. The Gateway Configuration task focuses on the device to device interoperability, addressed in the scope of INTER-IoT in the D2D Layer through the “Gateway Event Subscription” and “REST Request/Response” patterns (as described in the Deliverable 5.1). To add a new gateway, the following steps are followed: A gateway with the gateway software of INTER-IoT must be available. The hardware must be compatible with the INTER-IoT Gateway software. The compatibility list will be published in the INTER-IoT Gateway development site in GitHub. The Networking Configuration task focuses on network interoperability, achieved via network virtualization and the “Virtual Network Orchestration” pattern is configured and managed. The Application Services Configuration task includes a graphic tool for service orchestration. This is one of the less intrusive views in the INTER-FW, since the implementation

tool, the open source project “node-red” has a powerful user interface which allows this service orchestration from a visual perspective. In the Semantics Configuration task, configurable parameters and processes related to the semantics interoperability are configured for the deployment of interoperable IoT platforms. The last task of the Deployment phase is called User management Configuration and contains configurations valid for all the previous modules; in particular, it configures and manages the users of the INTER-FW and the authorized access of them to the IoT resources connected in INTER-IoT. The final work-product of the deployment phase is, therefore, a configured and deployment integrated IoT platform.

The fifth step of our methodology is the Testing phase. The integrated and deployed platform is executed and validated through testing according to well-defined test cases. Acceptance testing is a test conducted to determine if the requirements of a specification are met [29]. In systems engineering it may involve black-box testing performed on a system, such as for example for software modules. International Software testing Qualifications Board (ISTQB), which is a software testing qualification certification organisation, defines acceptance as formal testing with respect to user needs, requirements, and business processes conducted to determine whether a system satisfies the acceptance criteria [30]. There can be many types of acceptance testing for a system, service or product. The acceptance test can be performed multiple times in the case of defect resolving or when test cases are not executed within a single test iteration. In INTER-IoT acceptance testing is performed in two tasks: Factory Acceptance Testing (FAT) and Site Acceptance testing (SAT). Factory Acceptance Test (FAT) and Site Acceptance Test (SAT) are performed to test and evaluate the INTER-IoT-based integrated system implemented in the Implementation Phase and deployed in the Deployment Phase. The FAT task is performed to test and prove the system in a lab environment and tests if solution meets the specifications and if it is functional before it is deployed in the field. FAT tests can be performed by simulation or a functional test. The SAT task takes place after integration at the customer site and tests if the solution has been correctly integrated into the customer’s environment and meets all the requirements. During the SAT testing process the actual deployed system is tested and proven.

Maintenance is the final phase of our methodology with the objectives to maintain and track the evolution of the integrated IoT platform during time. Maintenance is referred to the identification of a list of bugs and/or a list of evolution points at specific INTER-IoT layers and/or products and to the consequent correction of bugs and/or implementation of new functionalities. The Maintenance activity is subdivided into two main tasks. Change Identification task aims at identifying bugs and/or evolution points of the integrated platform. Change Implementation task is the actual development of the changes, i.e. bug fixing or analysis, design, implementation, deployment, and validation of new functionalities; the latter could imply to re-execute, totally or partially, the integration process.

5 INTER-CASE

INTER-METH is supported by a CASE (Computer Aided Software Engineering) tool called INTER-CASE that helps supporting each aforementioned phase of the integration process and specifically provides the following functionalities:

- Support for workflow execution in each phase;
- Web-based Graphical facilities;
- XML-based project data repositories.

INTER-CASE is specifically intended to guide the IoT integrator in properly following and applying the integration workflow of the INTER-IOT instantiated INTER-METH. It is particularly effective to keep trace of integration choices at each phase of the methodology, so to favor and simplifying documentability of the IoT platforms integration project. In addition, it supports the reduction of inconsistencies during specification refinements from one step to the following, with warning and error messages provided to the integrator when inconsistent conditions are detected.

A simple, intuitive graphic user interface (GUI) characterizes the INTER-CASE Tool. It is composed of a:

- Navigation bar, with a menu that allows the integrator user to interact with the application
- Dashboard message bar, that displays the opened project
- Container, in which all the documents are presented to the user
- Footer, that contains application and copyright information.

The use of the INTER-CASE Tool is allowed after authentication, by entering a username and password in a traditional-looking form. Each authenticated user can choose to open a previously saved integration project or create a new one from scratch. Obviously, the user can modify or delete an existing project.

Each project has a status page showing the integration project summary at a glance. For each phase, a card contains a list of documents produced. In every card, the Integrator user can edit or export a document by using the specific buttons placed to right of the name of every document. In case a produced document contains inconsistencies (e.g. platforms name mismatch across documents), an alert icon will be shown so to allow the Integrator with quick visualization of the issue. Figure 7 depicts an example of Project Status in which only the Analysis and Design phases are displayed.

A menu to define the activities of a given phase becomes accessible to the integrator user once all tasks related to the previous phase are complete. In the following, we describe the various INTER-CASE functionalities for each phase of the INTER-METH methodology.

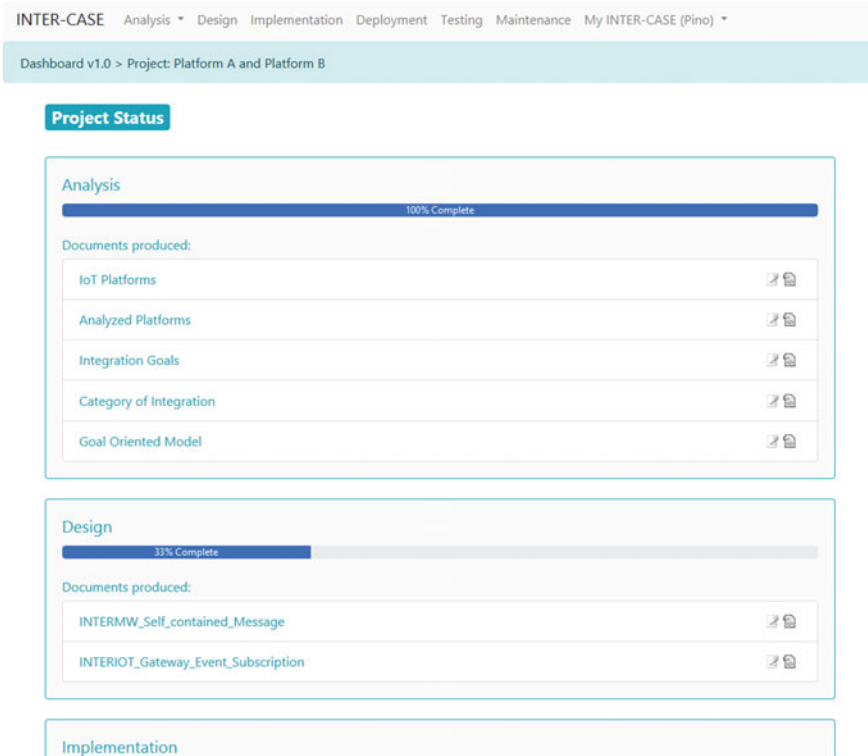


Fig. 7 INTER-CASE project status page

5.1 Analysis Phase

Through the Analysis menu, the analysis phase can be carried out. It is composed by five tasks. The completion of a task enables the access to the following one.

The first task is the IoT Platforms definition, where the integrator user defines generic information about the platforms to integrate, such as the platform type, the platform owner, the ontology type used.

The second task is the Analyzed Platforms Document (APD) definition. In this task the Integrator user has to analyze the platforms to integrate in terms of the INTER-IOT platform reference model described in the Deliverable D4.1. The APD, exemplified in Fig. 8, is a fundamental document in the integration project because represents the basis to analyze the platforms according to a homogeneous representation, which is necessary to identify the integration points among the platforms. The platforms in this document must be obviously the one identified in the IoT Platform document, so INTER-CASE automatically fills in the Platform Name field, although the Integrator could still edit the pre-compiled information; in case of information mismatch with

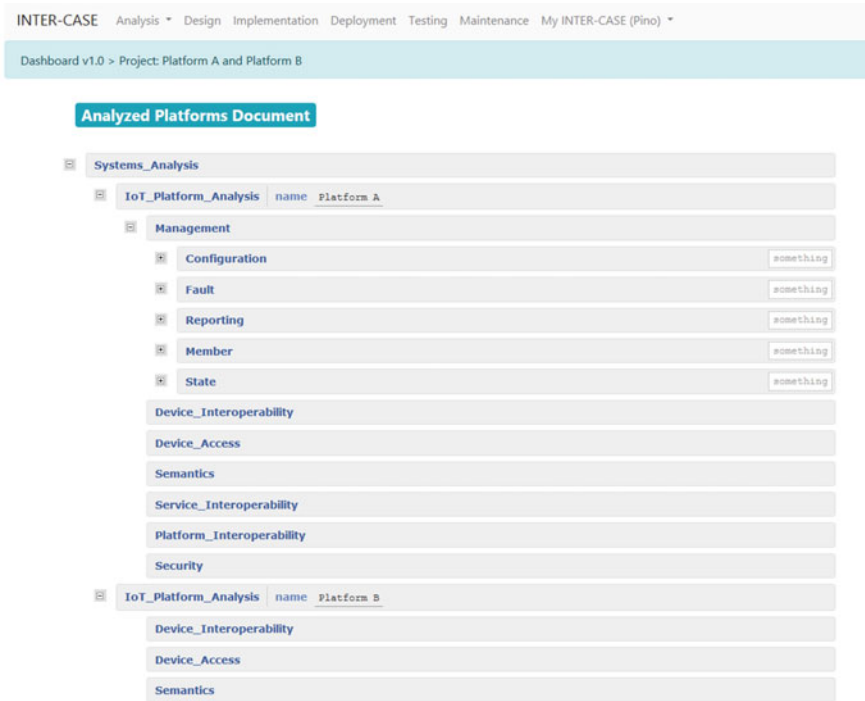


Fig. 8 Example of analyzed platforms document

the IoT Platform document, however, the Tool will report a warning to the Integrator user.

The third task is the Integration Goals definition where the integrator user identifies high-level integration requirements and objectives.

The fourth task is the Category of Integration definition. The integrator user identifies the integration layers, selecting them among those defined in INTER-Layer.

The last task generated the final output of the Analysis phase: the Goal Oriented Model (GOM) document. In this task the integrator user refines, in terms of functional (FR) and non-functional requirements (NFR) and according to the APD, the identified integration goals. The Goal Oriented Model document also includes the list of Category of Integration document produced in the previous step. In Fig. 9 an example of the GOM document is depicted.

5.2 Design Phase

INTER-CASE enables the Design phase when the integrator completes the Analysis phase. The page showed to the user is generated by the application according to

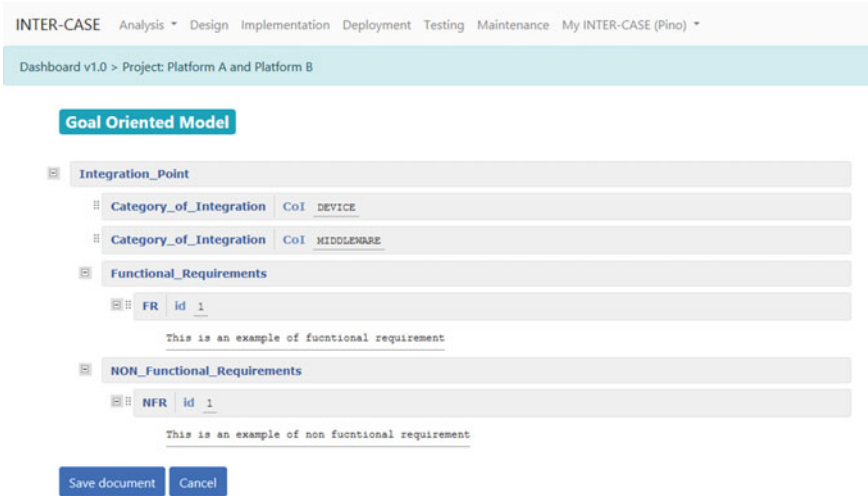


Fig. 9 INTER-CASE goal oriented model

the GOMI document defined in the previous Analysis phase. Based on the identified layers of integration, the integrator user must instantiate the design patterns proposed in this phase. The instantiation of a pattern occurs by selecting the specific pattern template, and involves opening the page of the corresponding pre-instantiated pattern (see Sect. 3). For a complete description of each identified and pre-instantiated pattern, the interested reader can refer to the Deliverable D5.2 of the INTER-IOT project. Figure 10 depicts an example of the Design document in which both middleware layer and device layer pattern are proposed.

5.3 Implementation to Maintenance

INTER-CASE also supports the following integration phases, from Implementation to Maintenance.

In the Implementation phase the integrator user creates a document to specify the information related to the public or private repository (or repositories) of the software source code under development.

In Deployment phase, INTER-CASE requests the integrator to specify information related to integrated platform deployment in terms of configuration parameters of the INTER-FW product (see Deliverables D4.x).

The Testing phase the integrator user creates a systematic document with the relevant tests to be carried out on the integrated platform. For each defined test, the integrator has to specify the objectives, the requirement to validate (preferably also

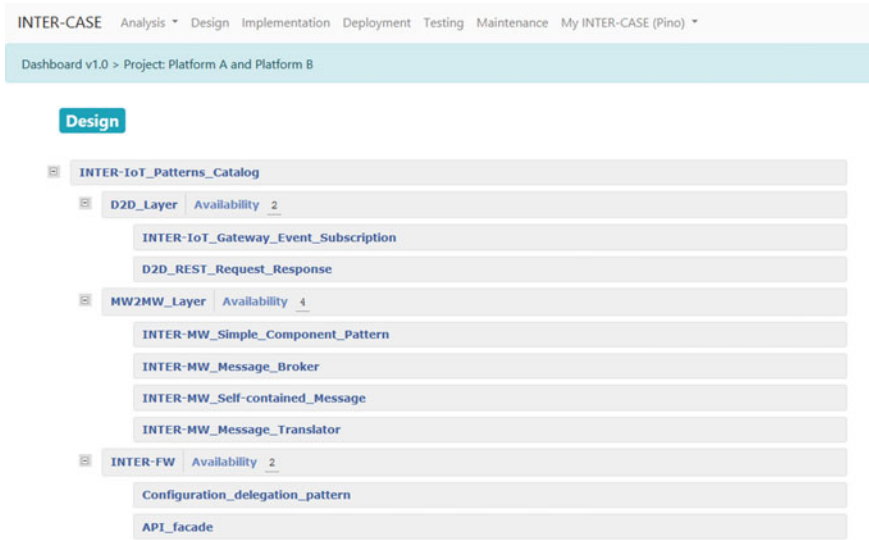


Fig. 10 INTER-CASE design phase: list of identified patterns

including NFRs identified in the Analysis phase), the tools necessary to execute the test, the test strategy, and of course the expected results of the test execution.

Finally, the Maintenance phase allows to create a “live” document in which the integrator inserts. From time to time, notes related to discovered bugs (including possible resolution actions) and evolution points which essentially represent future developments and functionalities of the integrated platform.

6 The INTER-Health Use Case: From Analysis to Design

To exemplify the use of INTER-METH for the integration of two heterogeneous IoT platforms, we will summarize the work done in the context of INTER-Health (see Deliverable D6.3), one of the two pilots developed in INTER-IOT, with particular focus on the Analysis and Design phases that are carried out and documented through our INTER-CASE tool.

The integration scenario, shown in Fig. 11 involves two different IoT platforms, BodyCloud [31] and UniversAAL [32], that need to be integrated to develop the INTER-Health Pilot.

To execute the INTER-METH workflow with INTER-CASE, the integrator logs in the tool and creates a new Integration Project first. The Analysis phase starts with the definition of the two platforms: each one is characterized by name, type, ontology, and owner.

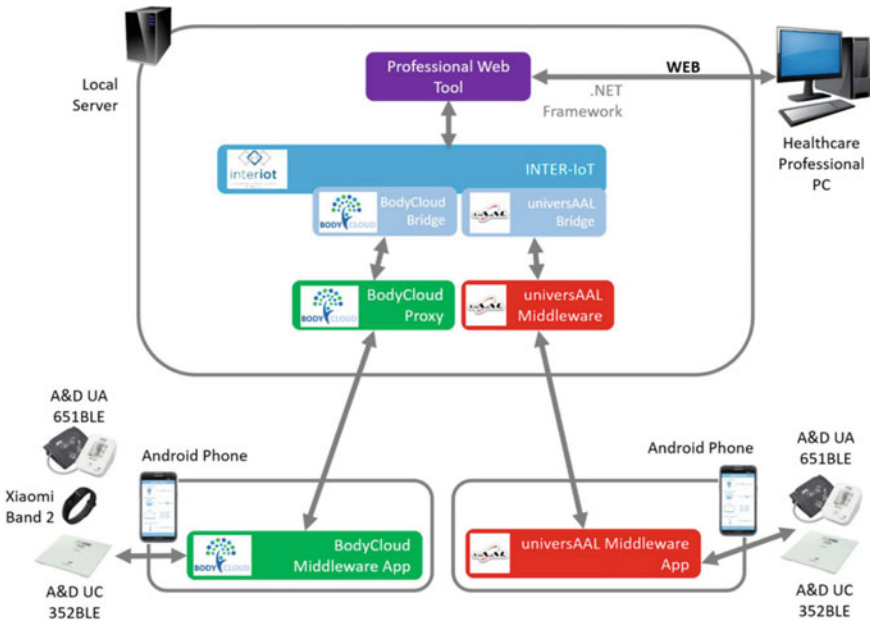


Fig. 11 INTER-Health integration scenario

Then, IoT Platforms Analysis is carried out by representing BodyCloud and UniversAAL using the common, homogeneous model: the INTER-IoT Reference Architecture (depicted in Fig. 3 and described in Deliverable D4.1). The representation of this model (called Analyzed Platform Document) is reported in Fig. 12.

The next task requires to define the high-level integration goals. In particular, for INTER-Health the following goals were identified by the integrator:

- IG1: Data generated from the two platforms have to be shared and transparently accessed from both platforms;
- IG2: Common notion of the users registered to each platform;
- IG3: Subscription support from one platform to the other to be notified upon the availability of new data of a given user.

After the elicitation of the integration goals and by studying the possible integration points with the support of the Analyzed Platform Document, the integrator user made the choice to integrate BodyCloud and UniversAAL at the “MIDDLEWARE” layer among the INTER-LAYER stack.

The final task of the Analysis phase is the INTER-GOM Document production. The high-level integration goals are refined in functional (FR) and non-functional (NFR) requirements; for INTER-Health the result of such work led to the definition of the following:

- FR1: Common knowledge and sharing of users IDs

INTER-CASE Analysis ▾ Design Implementation Deployment Testing Maintenance My INTER-CASE (Pilot) ▾

Dashboard v1.3 > Project: BodyCloud and UniversAAL

Analyzed Platforms Document

Systems_Analysis

- IoT_Platform_Analysis** name: ecoscut
 - Management**
 - Configuration (Device, Processing)
 - Reporting (Data)
 - Member (User Profile)
 - Device_Interoperability**
 - Network_Interoperability (Multi-technology, smartphone, gateway)
 - Device_Access**
 - IoT_Service (Data acquisition services, Data visu...)
 - Communication (Standard-based (IEEE 802.15.4, 802.11...)
 - Virtual_Entity (Not supported)
 - Service_Interoperability**
 - Service_Composition (REST-based)
 - Service_Resolution (REST-based)
 - Platform_Interoperability**
 - Platform_Access (REST-based API, Java-based API)
 - Security**
 - Authentication (Single authentication (username + p...))
- IoT_Platform_Analysis** name: universa
 - Management**
 - Configuration (Middleware, profiling tool, UI)
 - Fault (Context history, log monitor tool)
 - State (Only in supported technologies (jav...))
 - Member (User Profile Tool (It does not link...))
 - Device_Interoperability**
 - Device_To_Device_Interoperability (Adaptor-based for physical device a...)
 - Device_Access**
 - IoT_Service (API, Web IoT Service SDK)
 - Communication (HTTP)
 - Virtual_Entity (API Sensors)
 - Semantics**
 - Ontology_Alignment (Semantic reasoner)
 - Ontology_Resolution (Tools to support the basic opera...)
 - Service_Interoperability**
 - Service_Composition (Via an Interoperable Service Bus (ISB...))
 - Platform_Interoperability**
 - Platform_Access (Single middleware API)
 - Platform_Service (Mechanisms to allow the coexistence...)
 - Security**
 - Authentication (Multi-factor)

Save document Cancel Previous Step Next Step

Fig. 12 INTER-health integration: IoT platforms analysis

- FR2: Syntactical and semantic translation of messages (BodyCloud uses JSON while UniversAAL is based on RDF)
- FR3: Subscription management to topics (e.g. messages generated from a given user)
- FR4: User Access Gateway for Patients. The main functionalities for patients are: Access to services (providing username and password); Setting Profile communication and devices pairing; Managing measures on the device and releasing them to the gateway which stores them on a local database; Possibilities of inserting measures manually; Sending measures to the platform.
- FR5: Definition of reference meaning for health information. Health information can be detected using different devices according to different way of measurement (unit of measure that could differ from country to country and also depending on devices manufacturers). To use same information coming from different systems and going to others, it is mandatory to establish specific criteria to: (i) define a common meaning if it is possible, (ii) determine a correspondence between different data that have the same meaning and different values, (iii) set transcoding tables between different values of the same data.
- NFR1: Application response time. The “navigation” functionalities on different contents by using both Smartphone or Personal Computer to access to the platform, have to guarantee a response time of a few seconds.
- NFR2: Availability of sensor data. Health monitoring data must be accessible from a remote location to facilitate patient triage and inform decision making.
- NFR3: User Authentication to access INTER-Health services. Users shall authenticate to the services using their username and password.

The design phase of the INTER-Health integration process involves the requirements elicited in the INTER-GOM model. The integrator instantiates a set of INTER-IoT Design Patterns (see Sect. 3 and Deliverable D5.2) related to the integration of BodyCloud and UniversAAL platforms. The first step is the choice of the necessary pre-instantiated design patterns, among the available ones that are automatically filtered out by the INTER-CASE tool according to the input from the Analysis phase.

Given the choice of make the two platforms interoperable at Middleware layer, the integration design patterns that are selected belongs to the INTER-Middleware category. Specifically, three patterns will drive the implementation phase. *INTER-MW Message Broker* template is instantiated as follows:

- *Intent*: A component that facilitates passing of messages between decoupled INTER-Health components.
- *Problem and Solution*: Having point-to-point communication interfaces among several interacting components, makes dynamic reconfiguration, matching of different security constraints and QoS, requirements between components difficult. The employment of a message broker, can help to overcome limitations of point-to-point connection and enforce a common messaging interface upon different middleware components.

- *Applicability*: Each Inter-Health component communicates directly only with the message server broker which in turn handles the communications and dispatches messages to the respective destination component.
- *Implementation*: There is a message broker service that analyzes each received messages to check if there exists a subscription for the user that sent the message. If a subscription exists, the broker obtains (from the subscription message initially received) the necessary information to send the current message. If the subscription does not exist, the message is not forwarded to the external middleware components.

INTER-MW Message Translator template is instantiated as follows:

- *Intent*: Syntactical and semantic translation of messages between BodyCloud and UniversAAL.
- *Problem and Solution*: BodyCloud and UniversAAL use different message formats and models, respectively based on JSON and RDF, so direct communication between the two platforms is not possible. An effective solution is to introduce a translator component in the middle that is able to understand both message structures so to translate from the source platform format/model to the destination one.
- *Implementation*: Message translation is actually composed of two steps. First, there is need for syntactic translation to/from proprietary message format which is done in the bridge; the second step is semantic translation of the message and it is done in the Inter Platform Semantic Mediator (IPSM) component of Inter-IoT.

INTER-MW Self-contained Message template is instantiated as follows:

- *Intent*: Each message contains all the information needed to subscribe or unsubscribe to a given topic (or conversation).
- *Problem and Solution*: BodyCloud and UniversAAL need to communicate. To do so, it is necessary to know where one platform has to send the messages to the other platform. Hence, the identified solution is the creation of a subscription mechanism to conversations.
- *Implementation*: A bridge component of INTER-MW will process the messages to perform the proper action (i.e. subscription or unsubscription). The bridges then registers a corresponding callback to the action, in case of subscription message.

7 Conclusions

Interoperability among heterogeneous IoT platforms is a complex multi-faced issue which requires effective approaches that are difficult to implement and test. Indeed, it is not straightforward defining boundaries and requirements of the IoT platforms to be integrated as well as controlling the development environment. Traditional software/systems engineering approaches showed poor applicability in the IoT domain and, therefore, ad hoc and closed integration/interoperability solutions are often

adopted. INTER-IOT aimed at addressing the lack of systematic, generalizable methods and tools for IoT platforms interoperability. This chapter, in particular, described INTER-METH, developed in the project, that is first full-fledged, general-purpose engineering methodology completely supporting the integration process among IoT platforms. Particular focus has been given to the INTER-IoT-based Analysis and Design phases, that are fundamental drivers for the integration process. Relevant interoperability design patterns, the building blocks of the design phase, have been discussed. The chapter also presented INTER-CASE, a web-based tool that guides integrator designers to follow the methodology; INTER-CASE supports semi-automatic integration refinements and particularly useful to easily document the choices taken during the integration workflow. The practical use case related to the integration of the two IoT platforms (BodyCloud and UniversAAL) adopted in the context of the INTER-Health Pilot, is finally shown.

References

1. Savaglio, C., Fortino, G., Zhou, M.: Towards interoperable, cognitive and autonomic IoT systems: an agent-based approach. In: *Internet of Things, 2016 IEEE 3rd World Forum on*. IEEE, pp. 58–63 (2016)
2. Fortino, G., et al.: Towards multi-layer interoperability of heterogeneous IoT platforms: the INTER-IoT approach. In: *Integration, Interconnection, and Interoperability of IoT Systems*, vol. 199–232. Springer (2018)
3. Fortino, G., Russo, W., Savaglio, C., Shen, W., Zhou, M.: Agent-oriented cooperative smart objects: from IoT system design to implementation. *IEEE Trans. Syst., Man, Cybern.: Syst.* **48**(11), 1939–1956 (2018)
4. Bassi, A., et al.: *Enabling Things to Talk: Designing IoT Solutions with the IoT. Architectural Reference Model* (2013)
5. Houser, P.: *Best Practices for Systems Integration*. Northrop Grumman Corporation, Engineering & Product Excellence (November 2011)
6. OUSD AT&L.: *Systems Engineering Guide for Systems of Systems*. Pentagon, Washington, DC (August 2008)
7. Vaneman, W.: The system of systems engineering and integration “Vee” model. In: *Systems Conference (SysCon), 2016 Annual IEEE*. IEEE (2016)
8. Gama, K., Touseau, L., Donsez, D.: Combining heterogeneous service technologies for building an Internet of Things middleware. *Comput. Commun.* **35**(4), 405–417 (2012)
9. Aloï, G., et al.: Enabling IoT interoperability through opportunistic smartphone-based mobile gateways. *J. Netw. Comput. Appl.* **81**, 74–84 (2017)
10. Ishaq, I., et al.: Internet of things virtual networks: Bringing network virtualization to resource-constrained devices. In: *2012 IEEE Intl. Conf. on Green Computing and Communications (GreenCom)*. IEEE (2012)
11. Waterfall methodology: https://en.wikipedia.org/wiki/Waterfall_model, last accessed May 2019
12. Robertson, S., Robertson, J.: *Mastering the requirements process: getting requirements right*. Addison-Wesley (2012)
13. Zambonelli, F., Jennings, N.R., Wooldridge, M.: Developing multiagent systems: The Gaia methodology. *ACM Trans. Softw. Eng. Methodol. (TOSEM)* **12**(3), 317–370 (2003)
14. Giunchiglia, F., Mylopoulos, J., Perini, A.: The TROPOS software development methodology: processes, models and diagrams. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part, p. 1*. (2002)

15. Fortino, G., Russo, W.: ELDA^{Meth}: an agent-oriented methodology for simulation-based prototyping of distributed agent systems. *Inf. Softw. Technol.* **54**(6), 608–624 (2012)
16. ELDA^{Tool}, documentation and software, <http://lisdip.deis.unical.it/software/eldatool>, last accessed May 2019
17. DeLoach, S.A., Wood, M.F., Sparkman, C.H.: Multiagent systems engineering. *Int. J. Softw. Eng. Knowl. Eng.* **11**(3), 231–258 (2001)
18. Padgham, L., Winikoff, M.: Prometheus: a methodology for developing intelligent agents. In: *Proceedings of the First International Joint Conference on Autonomous Agents and Multiagent Systems: Part*, p. 1. (2002)
19. Garijo, F.J., Gomez-Sanz, J.J., Massonet, P.: The MESSAGE methodology for agent-oriented analysis and design. *Agent-Oriented Methodol.* **8**, 203–235 (2005)
20. Slama, D., Puhlmann, F., Morrish, J., Bhatnagar, R.M.: *Enterprise IoT: Strategies and best practices for connected products and services*, O'Reilly media (2015)
21. Bassi, A., et al.: Enabling things to talk. Springer (2013). <https://www.oasis-open.org/committees/soa-rm/faq.php>
22. <https://www.oasis-open.org/committees/soa-rm/faq.php>, last accessed May 2019
23. Zambonelli, F.: Towards a General Software Engineering Methodology for the Internet of Things. arXiv preprint [arXiv:1601.05569](https://arxiv.org/abs/1601.05569) (2016)
24. The IoT-A Unified Requirements list, http://www.iot-a.eu/public/requirements/copy_of_requirements, last accessed May 2019
25. Grace, P., et al.: Taming the interoperability challenges of complex iot systems. Presented at the (2014)
26. Grace, P., Pickering, B., Surridge, M.: Model-driven interoperability: engineering heterogeneous IoT systems. *Ann. Telecommun.* **71**(3–4), 141–150 (2016)
27. Kazman, R., et al.: Understanding patterns for system of systems integration. Presented at the (2013)
28. Report on High-Level Architecture (HLA) http://ec.europa.eu/newsroom/dae/document.cfm?action=display&doc_id=11812, last accessed May 2019
29. Acceptance Testing. [Online]. https://en.wikipedia.org/wiki/Acceptance_testing, last accessed 9 Aug 2021
30. Standard glossary of terms used in Software Testing, Version 2.1. ISTQB. 2010., [Online]. https://en.wikipedia.org/wiki/Acceptance_testing#cite_ref-ISTQB_Glossary_2-0, last accessed May 2019
31. Fortino, G., Parisi, D., Pirrone, V., Di Fatta, G.: BodyCloud: a SaaS approach for community body sensor networks. *Futur. Gener. Comput. Syst.* **35**(6), 62–79 (2014)
32. UniversAAL website, <https://www.universaal.info>, last accessed 9 Aug 2021
33. Fortino, G., Russo, W., Zimeo, E.: A statecharts-based software development process for mobile agents. *Inf. Softw. Technol.* **46**(13), 907–921 (2004)
34. Fortino, G., Garro, A., Russo, W.: An integrated approach for the development and validation of multi-agent systems. *Comput. Syst. Sci. Eng.* **20**(4) (2005)
35. Fortino, G., Savaglio, C., Spezzano, G., Zhou, M.C.: Internet of Things as system of systems: a review of methodologies, frameworks, platforms, and tools. *IEEE Trans. Syst. Man Cybern. Syst.* **51**(1), 223–236 (2021)
36. Fortino, G., Russo, W., Savaglio, C., Shen, W., Zhou, M.: Agent-Oriented Cooperative Smart Objects: From IoT System Design to Implementation. *IEEE Trans. Syst. Man Cybern. Syst.* **48**(11), 1939–1956 (2018)
37. Savaglio, C., Ganzha, M., Paprzycki, M., Badica, C., Ivanovic, M., Fortino, G.: Agent-based Internet of Things: state-of-the-art and research challenges. *Future Gener. Comput. Syst.* **102**, 1038–1053 (2020)
38. Fortino, G., Garro, A., Russo, W.: Achieving Mobile Agent Systems interoperability through software layering. *Inf. Softw. Technol.* **50**(4), 322–341 (2008)
39. Cossentino, M., Fortino, G., Garro, A., Mascillaro, S., Russo, W.: PASSIM: a simulation-based process for the development of multi-agent systems. *Int. J. Agent Oriented Softw. Eng.* **2**(2), 132–170 (2008)

Interoperability Application in e-Health



Gema Ibáñez-Sánchez, Alvaro Fides-Valero, Jose-Luis Bayo-Monton, Margherita Gulino, and Pasquale Pace

Abstract This chapter describes INTER-HEALTH use case, a real application of the INTER-IoT framework in a healthcare environment. INTER-HEALTH provides a solution that allows health experts to prevent and reduce obesity, which is one of the main causes of chronic diseases. Through INTER-IoT, two different platforms are able to interoperate to exchange information and so, to provide aggregated information to health experts. The results show a clear improvement in the health of the participants compared with those that did not use INTER-HEALTH solution.

1 Introduction

It is a well-known fact that the world is aging rapidly. Living longer does not mean living better, which can signify a reduction in Quality of Life (QoL). It leads to a population with greater dependence that make health systems be overwhelmed. It generates a necessity of solutions to alleviate this burden. Health systems need better data to understand the health risks faced by population to target appropriate prevention and intervention services. It is not about providing medicines to mitigate a health problem; is to provide an optimal health service to contribute to having a **good quality of life throughout their lives** [1].

G. Ibáñez-Sánchez (✉) · A. Fides-Valero · J.-L. Bayo-Monton
UPV, Universitat Politècnica de València, Valencia, Spain
e-mail: geibsan@itaca.upv.es

A. Fides-Valero
e-mail: alfiva@upv.es

J.-L. Bayo-Monton
e-mail: jobamon@upv.es

M. Gulino
Dipartimento di Prevenzione, ASL TO5, Nichelino, Torino, Italy

P. Pace
University of Calabria, Rende, Italy

© Springer Nature Switzerland AG 2021

C. E. Palau (eds.), *Interoperability of Heterogeneous IoT Platforms*, Internet of Things, https://doi.org/10.1007/978-3-030-82446-4_8

Nowadays, Personalised Medicine is a new field where one of its objectives is to achieve the best outcomes for preventing or managing a patient's disease. Health is determined by intrinsic characteristic of humans combined with their lifestyle and environment. The combination of this socio-medical information from humans' life can help to determine individual risk of developing a disease, early detection of illnesses, or even more effective interventions to improve health.

In that line, the Internet of Things (IoT) brings to the market a portfolio plenty of opportunities, enabling pervasive communication between the physical and the virtual world [2]. The IoT has the potential to contribute to many e-Health applications such as remote health monitoring, chronic diseases, or elderly care among others [3, 4]. The number of e-Health applications in the market is increasing rapidly, which generate a huge amount of data, being medical data the backbone of the healthcare systems [5–7].

At the same time, the increasing shift to patient-centric generates the need for an integrated healthcare platform to achieve scalability and engagement. It is not common for healthcare organizations to have a unique Electronic Health Records (EHR). Usually, they may have several data sources for different purposes. This fragmentation leads to put at risk the integrity of the data, to define time-consuming processes and to create challenges in coordinating care for the patient, rising maintenance costs [8].

The aim of connected healthcare begins with an integrated system to a data-oriented healthcare. Nonetheless, it is not always possible to integrate everything in the same platform.

Due to medical data importance, IT challenges for handling medical data, becomes more serious. Interoperability is a critical issue for communicating and handling different formats of medical data. At the same time, various legal and technical standards are involved in handling, processing and communicating the medical data. Interoperability and integration in healthcare system depends on various connectivity levels. It covers a very vast technical spectrum and poses huge challenges to connect different pieces (health platforms, m-Health solutions, IoT devices or any other resource that contribute to a better patients' life) in a regulated and standardized way for the healthcare system workflows [9–11].

INTER-IoT framework applied in the health sector, INTER-Health, is designed and built to specifically accommodate the communication and processing needs of patients and Health Professionals (HP). INTER-Health integrates different IoT architectures, e-Health services and health sensors. It is a proof of concept of the potential of INTER-IoT in the provision of interoperability within a clinical environment. INTER-Health improves healthcare service offered by the hospital by providing continuum of care.

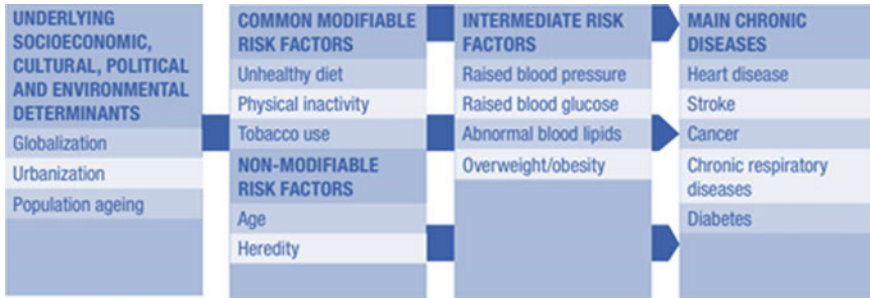


Fig. 1 Causes of chronic diseases

2 INTER-Health Motivation

Obesity is one well-known risk factors faced by doctors today, with the corresponding allocation of medical care resources for the disease and related comorbidities. Obesity is associated with a high incidence of a number of diseases, including diabetes, cardiovascular disease, strokes, chronic respiratory diseases and cancer (Fig. 1) [12]. It happens in almost all parts of the world, regardless of age, gender or geographical origin. It is because of following an unhealthy lifestyle, in terms of a high calories intake diet, and lack of physical activity. Every year in the world [13]:

- **2.8 millions of people die** from obesity and overweight;
- **2.6 millions of people die** from high cholesterol levels;
- **7.5 millions of people die** from hypertension;
- **3.2 millions of people die** from absence of physical activity.

2.1 INTER-Health Scenario

John is 40 years old, works in the office, he is slightly overweight. For some time he feels a little tired and therefore went to the family doctor for a check-up. The doctor visits him and prescribes some control tests. The test results are normal but the doctor suggests that he contacts the service of Hygiene Nutrition Unit (ASLTO5) [14] in order to change his lifestyle and to prevent diseases such as hypertension, diabetes and so on.

The doctor of the ASLTO5 in addition to provide a set of behavioural and dietary recommendations proposes to John to undergo a period of monitoring to help changing lifestyle and prevent diseases caused by obesity. Using INTER-Health the ASLTO5 operator

- registers the John personal data

- provides devices (weight scale, blood pressure monitor, activity monitoring sensor) for use at home to make periodic measurements and tells it how to use them
- collects the John measures
- sets the frequency of measurements
- sets the schedule for the completion of the questionnaire.

John starts the program compiling the online questionnaire on eating habits and physical activity using INTER-Health.

Each day John wears the monitoring activity sensor and aims to reach the number of steps suggested checking on INTER-Health.

Periodically (following the frequency of measurements suggested by ASLTO5) he measures the pressure and weight and sends the data using INTER-Health.

The ASLTO5 operator periodically checks if John is following the protocol and checks the measures and the activities on INTER-Health. In case of any detected problem (e.g. absence of measures for an interval of time, outside threshold values) he can contact John.

2.2 INTER-Health Interoperability Components

Integrating data from different types of diverse sources and clinical systems is a fundamental challenge for any healthcare entity in order to enhance patient care and performance indicators. This section describes how INTER-Health deals with interoperability and the different components to interoperate [11, 15, 16].

2.2.1 universAAL Platform

universAAL [17] is a semantic and distributed software platform. It was designed to ease development of integrated Ambient Assisted Living applications. Its main advantage resides on its suitability for IoT, wearables, Big Data and so on [18, 19].

The semantic nature of universAAL makes it ideal for highly heterogeneous environments. It allows representing the world through ontologies, i.e. different devices, from different vendors, can be modelled using the same ontological concepts, thus applications will not see the difference between them. Different health and social services have been validated at large scale pilots in the make IT ReAAL [20] project with more than 5500 users.

In INTER-Health, the service provided by universAAL allows the collection of patient's data through quantitative measurement of different physiological values, weight scale and sphygmomanometer, at ASLTO5 facilities.

2.2.2 BodyCloud Platform

BodyCloud [21] is a SaaS architecture that supports the storage and management of body sensor data streams and the processing (online and offline analysis) of the stored data using software services hosted in the Cloud. In particular, BodyCloud endeavours to support several cross-disciplinary applications and specialized processing tasks. It enables large-scale data sharing and collaborations among users and applications, and delivers services via sensor-rich mobile devices.

In INTER-Health, Body-side component, currently based on Android, monitors an assisted living through wearable sensors and provides the collected data through INTER-IoT.

2.2.3 Professional Web Tool (PWT)

This website is an ad hoc solution that recollects data from universAAL and Body-Cloud platforms through INTER-IoT. PWT is a dedicated website for monitoring the progress of patients, helping health professionals in their daily activity [22].

The two platforms, universAAL and BodyCloud, share some high-level characteristics while differ in objectives and technology. Specifically, they are both e-Health platforms, based on Bluetooth technology to interact with measurement devices. However, they have different specific objectives and are not interoperable from a technological point of view [23]. Their specific objectives are complementary: universAAL is focused on non-wearable measurement devices at ASLTO5 premises, whereas BodyCloud provides remote monitoring through online eating and physical activity habits questionnaires and wearable devices organized as body sensor networks [24–26]. Their integration delivers a mHealth integrated platform on top of which multitudes of mHealth services could be developed, such as the PWT, **contributing to the continuum of care.**

3 INTER-Health Solution

The goal of the INTER-Health [27] pilot is to foster healthy lifestyle and prevent chronic diseases by monitoring subjects' physical characteristics, nutritional behaviour and activity.

From the health professionals' perspective, to work on preventing chronic diseases following a healthy lifestyle would be translated into savings time and more efficient health treatments in ASLTO5, increasing efficiency with the same resources used.

From patients' point of view, an e-Health solution would help to improve their adherence to the prevention program, i.e. patients do not have to go to the hospital premises so often, motivational messages...

The integrated open platform supports health monitoring at health-care centre through the centre facilities, at home through a set of medical consumer devices,

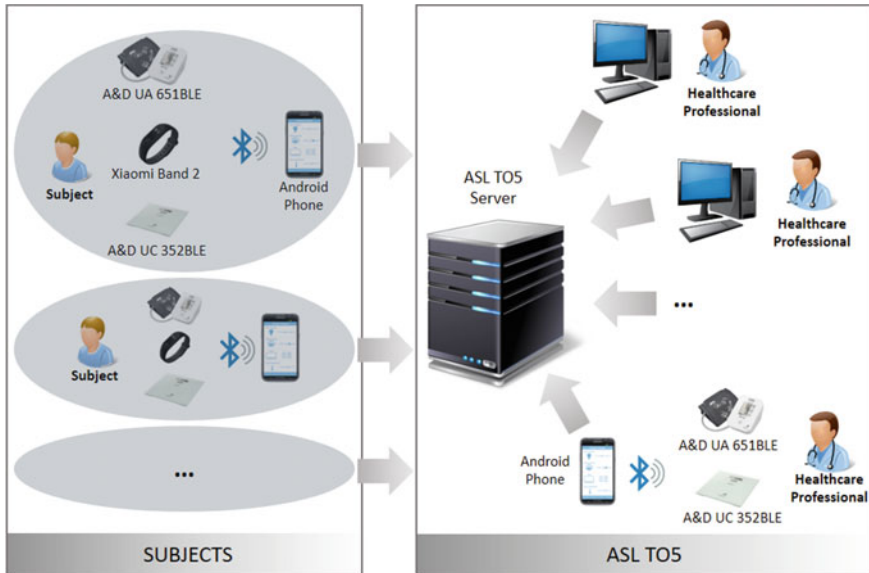


Fig. 2 INTER-Health conceptual design

and in mobility based on body sensor networks (Fig. 2). In order to evaluate the integration from functional and non-functional perspectives, atop the interoperable platform, we develop and deploy in a controlled medical testbed, a fully-working application, related to the lifestyle monitoring. The application has as specific objectives to improve and overcome the currently available methods, instruments and protocols.

In Fig. 3 can be found Local Server at the premises of the clinical centre. It runs the following components: The INTER-IoT Framework, the instance of universAAL, the PWT in .NET and its Database in a SQL Server.

The INTER-IoT Framework runs in its Virtual Machine and contains all INTER-IoT modules needed for the pilot, of which the ones of interest in INTER-Health are the INTER-IoT Middleware and INTER-IoT API.

The universAAL instance is an OSGi container with all the universAAL modules needed for the pilot, of which the ones of interest in INTER-Health are the REST API, and all the modules that compose the basic universAAL Middleware.

The .NET Framework hosts the PWT web application, which finally allows HPs to manage all the data within the pilot.

The SQL Server hosts the Database used by the PWT to store its data.

The setup of the mobile phone used at the clinical centre differs from those used at each patient's home. The mobile phone used at the clinical centre is an Android Phone running the universAAL Android App and a dedicated app for getting measurements from Bluetooth devices. The mobile phone used by the subjects is an Android Phone running the BodyCloud Android App.

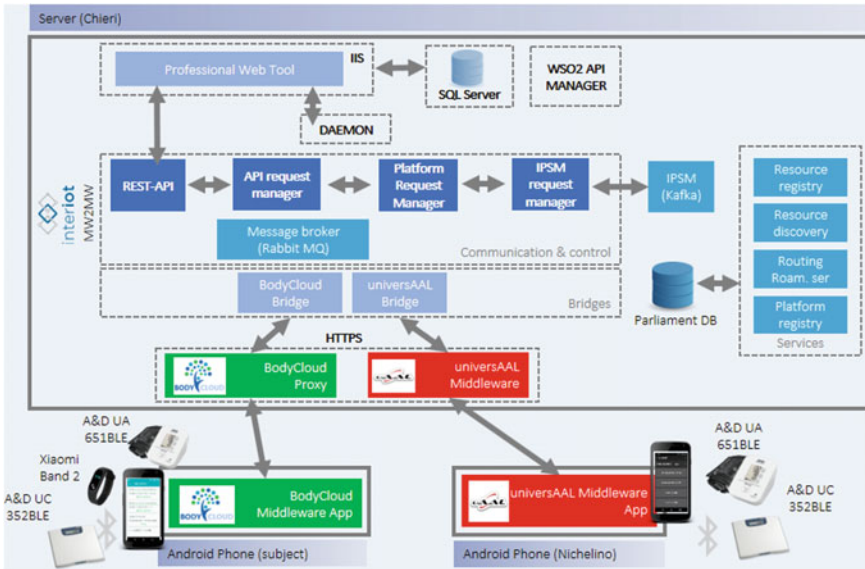


Fig. 3 INTER-Health high-level pilot design

The Bluetooth devices used at the clinical centre differ from those used at each subject’s home. The models used at the clinical centre and by each subject are A&D Medical UA 651BLE (Blood Pressure) and A&D Medical UC 352BLE scales which are Bluetooth Low Energy devices, in addition to the Xiaomi Band 2 for Physical Activity.

The PCs used by HPs at the clinical centre to access the PWT are their own regular PCs.

3.1 INTER-Health Features

INTER-Health is focused on providing prevention services, for healthy people with different level of risk of developing chronic diseases, based on a monitoring system helping to follow the prevention program to change their lifestyle.

The monitoring services are obtained by the interoperability between different platforms offering the overall needed features, medical and wearable devices and use of mobile phone with applications [19].

This scenario requires similar services based on the integration of two e-Health existing platforms and used to handle different health problems. The chronic disease prevention is based on four main use cases (Fig. 4):

- Creates and operates patients and associated services
- Sets patients protocol parameters (kind of measures, thresholds, periodicity)

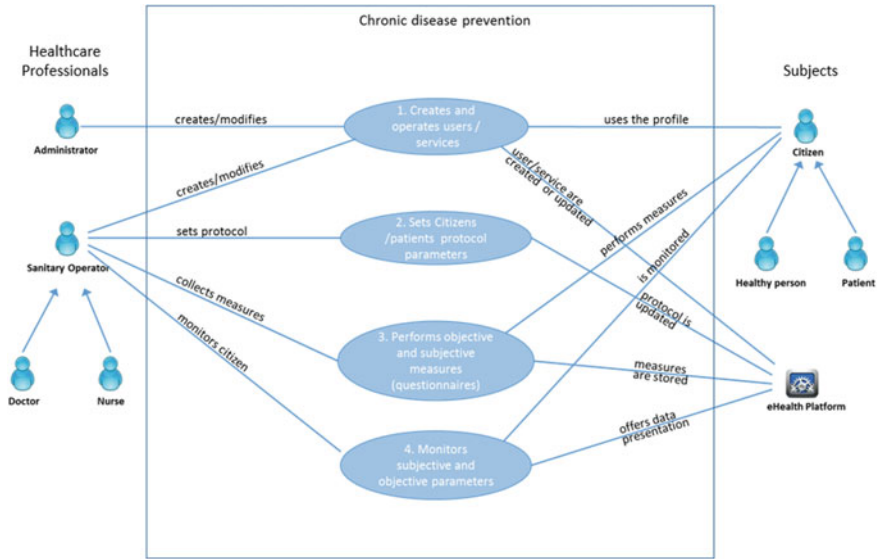


Fig. 4 INTER-Health use cases

- Performs objective measures (e.g. weight, activity) and subjective measures (lifestyle questionnaires)
- Monitoring measures and trends.

The PWT is divided into two main sections: (1) Patient’s Profile to manage subjects’ data and (2) Patient’s Progress to see their evolution.

3.1.1 Creates and Operates Users/Services (PWT)

HP enters his/her user and password in the welcome page and press LOGIN button (Fig. 5). If it is correct Patient’s List screen is open (1). Patient’s list screen provides to HP a quick overview of the general status of the patients, including if there is any alert. List can be ordered by column (ascending/descending order).

To add a new patient, the HP clicks on the ADD button. Fill the data and click on SAVE button (2). Warnings appear when mandatory fields are not completed properly. When it is created correctly, go back to the Patient’s List and the new patient appears (3).

Alerts

Alerts can be triggered based on the values sent by the patients through INTER-IoT. After the reception of an alert, a direct call is done to the patient in order to check his/her status. Below is shown the criteria to launch alerts:

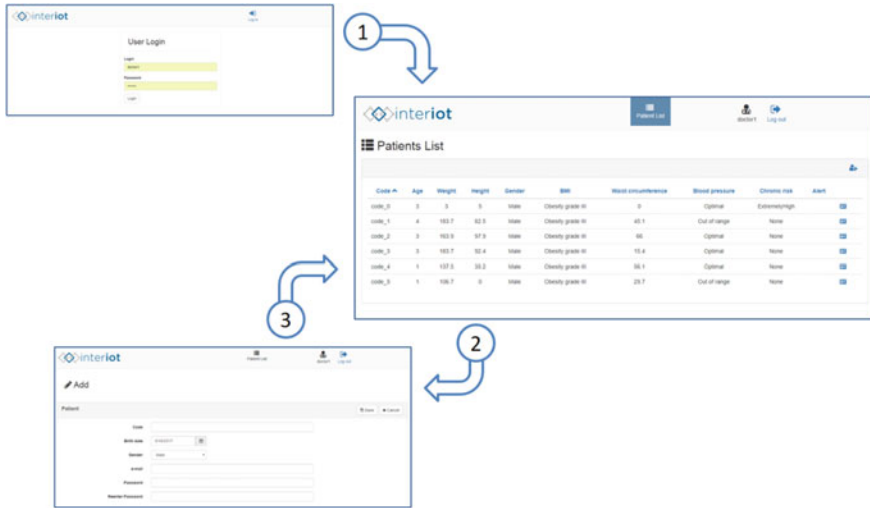


Fig. 5 Creates and operates users/services use case

- **Weight** is evaluated every 4 weeks: if the weight increases, it is notified to the user by means of an informative message (“Beware, your weight has increased! Improve your lifestyle”) and then the health professional through the PWT (Patient code—Alert weight); In the case that the weight decreases, it is the patient who receives a motivational message (“Congratulations, you are losing weight! Continue like this”).
- **Blood pressure** is evaluated every 7 days: if the patient’s blood pressure increases, the HP is alerted by indicating an alert in the PWT (“Patient Code—Pressure Alert”). HP contacts the patient to guide him with healthy lifestyle advices.
- **Physical activity (steps)** value is evaluated daily. Depending on the following ranges, it has been shown a different message to the patient:
 - 0–4999: “Beware, you’re sedentary, move more!”. In this case, also the HP has been alerted (“Patient Code—Alert steps”), who contacts by phone with the patient to advice him/her.
 - 5000–7499: “Your level of physical activity is low and take advantage of every opportunity to move!”
 - 7500–9999: “Congratulations! You just have a little effort to reach the recommended physical activity level!”
 - 10000–12499: “Great! Continue to keep you alive”
 - ≥12500: “Great! Continue to keep you alive”
- **Physical activity (duration)** is evaluated weekly
 - <150 min/week. The following message is shown to the patient: “The physical activity you are doing is not enough yet”. The HP also is alerted: “Patient Code—

Duration Physical Activity Alert”, who contacts by phone with the patient to advice him/her.

- ≥ 150 min/week. A motivational message is shown only to the patient (“Congratulations, keep moving so!”)

- **Questionnaire of eating and physical activity habits.** It is evaluated per question, where each one has identified the right and the wrong answers. When a question is answered wrongly, it is shown a warning message. Otherwise, it is shown a motivational message. If the patient has answered all questions incorrectly, then it triggers an alert that is shown in the PWT for the HP. The HP calls the patient by phone to advice him/her.

3.1.2 Sets Citizens/patients Protocol Parameters (PWT)

Before going to the Patient’s List, it is worth to mention the Main Menu Bar, as central navigation component. From this bar, HPs can go to the Patient’s Profile and Patient’s Progress screens or go back to the Patient’s List to choose another subject for counselling (Fig. 6).

Patient’s folder is managed in this first tab. It is composed by two sub-sections: (1) Patient’s folder and (2) check-ups. Patient’s profile contains personal information relevant to the counselling. Check-ups stores information derived from the face-to-face interviews carried out in ASL TO5 facilities. Click on EDIT button to modify patient’s profile data, change values and then click on SAVE button to keep the modifications or CANCEL to discard everything (4). To modify principal data, click on EDIT IMPORTANT DATA and change it, then click on SAVE button to keep the modifications or CANCEL button to discard everything. In the same screen, a new counselling can be added by clicking on the ADD button. Then a new screen with nutritional status, and lifestyle data. Notice that the patient’s profile data appears but cannot be modified. Click on SAVE button to keep the entire information or CANCEL button to discard all. HP can print all check-ups registered in the PWT by clicking on the PRINT button.

When a new check-up is added, it appears in the list of check-ups. To see the details of a check-up, click on VIEW button. To edit an existing check-up, click on EDIT button, then a new screen will appear. Collected data is the following.

- Objective data:
 - Personal data (name, surname, gender, age, address);
 - Personal data (civil status, educational level, social and economic status);
 - Anthropometric data (weight, height, Body Mass Index—BMI, blood pressure, waist circumference)
 - Hematochemical data (blood glucose, blood insulin, hemoglobin, glycated hemoglobin, cholesterol level, HDL, LDL, blood triglyceride, blood nitrogen, AST, ALT, GGT, blood creatine, urea, blood urea, blood albumin, prealbuminemia, TSH, T3, T4, erythrocyte sedimentation rate).

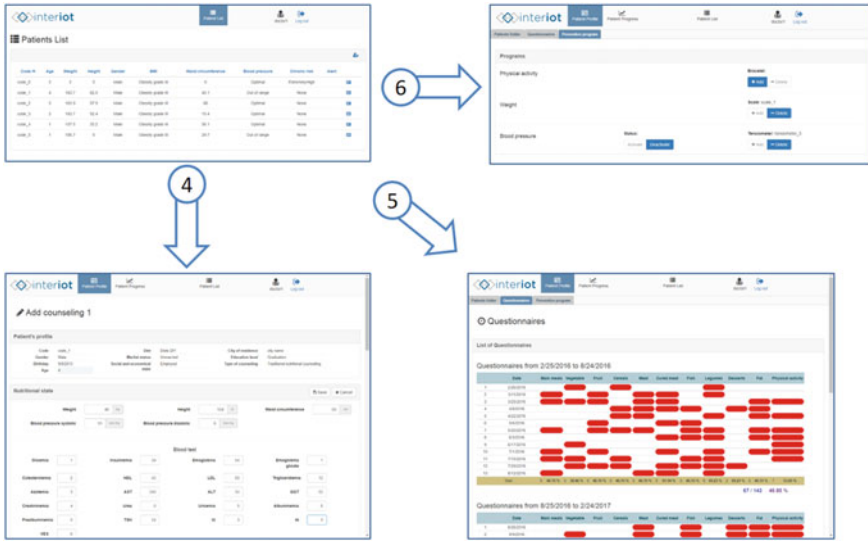


Fig. 6 Set citizens/patients protocol parameters use case

• Subjective data:

- Food anamnesis (breakfast, main meals, vegetables consumption, fruit consumption frequency, red and/or white meat consumption frequency, processed meat consumption frequency, egg consumption frequency, cheese consumption frequency, fish Consumption frequency, legumes consumption frequency, bread and pasta and substitutes consumption frequency, dry fruit consumption frequency, oil consumption frequency, animal fats consumption frequency, salt consumption frequency, herbs and spices consumption frequency, sugar and/or honey consumption frequency, sweetening consumption frequency, sweet consumption frequency, water consumption frequency, alcohol consumption frequency, sugary drink consumption frequency)
- Physical activity practice (daily physical activity, organized physical activity).

Tab Questionnaires shows data received of the online questionnaire from the mobile app (5). Each row represents an online questionnaire. Prevention programme is intended to assign devices to subjects belonging to EG (6). By default all the subjects of this group has a bracelet and a scale. In the case of having a high-pressure diagnosis, (s)he may need a blood pressure device, in which case, that option should be activated and then assign a device.

universAAL app

Measurements of the anthropometric data is done through universAAL platform. The HP is authenticated in the mobile application, then selects the device to use and the measurement is done (Fig. 7).

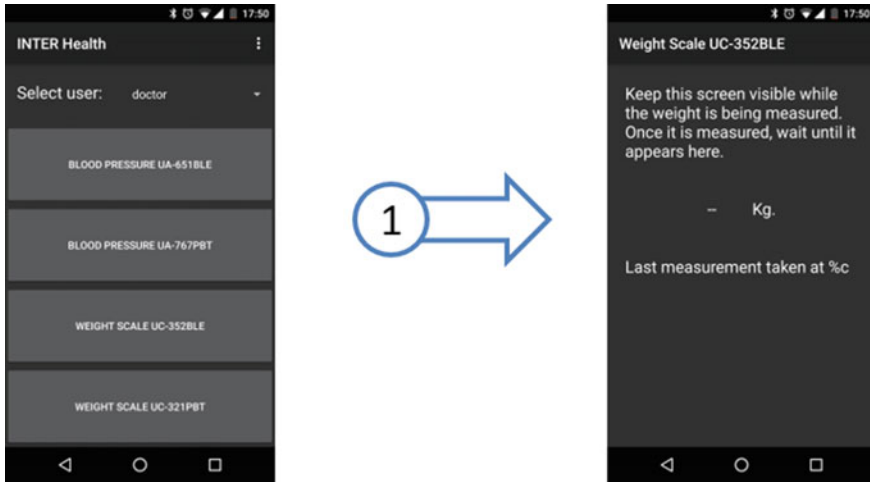


Fig. 7 Measurements of the anthropometric data

Prevention program

The prevention program assigned to patient was as follow:

- **Weight** reported weekly;
- **Blood pressure**, for those patients that at the first nutritional counseling were identified as Normal-High pressure values (systolic pressure ≥ 130 and/or diastolic pressure ≥ 85) with a daily frequency (morning and evening);
- **Physical activity** (number of steps and duration of physical activity practiced) reported daily. In particular, to accomplish with the physical activity objective, EG patients had to achieve at least 10.000 steps daily and 150min of activity per week. Additionally, the questionnaire available in the mobile app gathered information on eating and the physical activity habits, and provided feedback to the patients to motivate them. The eating habits and the physical activity practice were taken biweekly.

3.1.3 Performs Objective and Subjective Measures Use Case (BC Platform)

Patient login and accesses to the Summary screen (1) (Fig. 8). This screen shows activities pending. If the patient click on any of them is redirected to the corresponding screen. From main menu, the patient can go to the measurements screen (2). The patient can measure anthropometric values by using the medical devices. Eating and physical activity habits questionnaire is available in the next screen (3).

It is shown a question per screen, with two options to answer, and the possibility to save and resume the questionnaire. When the patient answers, the app shows a



Fig. 8 Performs objective and subjective measures use case

motivational or warning message. Patient can see his/her weight, blood pressure and activity progress in the available graphs (4).

3.1.4 Monitors Subjective and Objective Parameters Use Case (PWT)

Patient Progress shows the progress of the patient in terms of weight (Fig. 9) (7). The graph allows seeing different zooms: 1, 3, 6 months, Year-to-date (YTD), 1 year and all the available data. It is possible to zoom in and navigate between dates by using the date range. To zoom in click and drag on an area of the graph. To reset zoom, click on RESET ZOOM button that will appear after doing zoom in at the right top of the graph. And navigation bar lets navigate and do zoom in/out.

Next tab is about physical activity and here can be checked steps and minutes of activity per day (8). As in the previous graph, you can do zoom in by clicking and dragging, using the pre-defined zooms (1m, 3m, 6m, YTD, 1y or all), the date range or the navigation bar.

The last tab is aimed at blood pressure information (9). The first graph is about the measurement taken every day in the morning and at evening. By clicking on the



Fig. 9 Monitors subjective and objective parameters use case

legend, text you can enabled/disabled the lines of the graph. As in the previous graph, you can do zoom in by clicking and dragging, using the pre-defined zooms (1m, 3m, 6m, YTD, 1y or all), the date range or the navigation bar. The second graph (Blood pressure average) shows the average per day of systolic and diastolic measurements.

3.1.5 Device Management (PWT)

Administrator user has access in PWT to the screens (Fig. 10) to manage the medical devices: bracelets (7), weight scales (8), and sphygmomanometers (9). (s)he is able to add, edit and unabled devices.

3.1.6 Diagnostic Tool (PWT)

Administrator user has access to the diagnostic tool, which allows monitoring data flow of the system (Fig. 11). By selecting one of the platforms available, the diagnostic tool shows the list of “things” registered (4). Selecting one “thing” the list of messages associated to that one is shown (5).

3.2 Privacy and Security

In order to guarantee the data protection and personal data treatment the following security mechanisms have been applied:

1. Data transfer procedure to the server involves deleting the local data copy stored in the Smartphone Memory Buffer Received on the ASL TO5 Server;
2. Bi-directional communication between the application resides on the subject’s Smartphone and the server is done by using login mechanisms, combined with the HTTPS secure communication protocol, to assure server authentication, privacy protection, encryption, and integrity of the server Data exchanged between the communicating parties; Also the biomedical data travels in anonymous form as they is associated with an alphanumeric code generated by the server and therefore not associated with the subject’s identification data;

More concretely, the mobile app guarantees the data protection and personal data treatment throughout the following techniques:

1. The data communication protection between biomedical wireless devices and the developed mobile gateway application installed on the patient’s Smartphone, is guaranteed by proprietary mechanisms supported by device manufacturers; it is worth noting that such mechanisms are not altered in any way by the partner.
2. Data collected by the mobile gateway application is temporarily stored in the memory buffer of the patient smartphone waiting to be transmitted to the ASL TO5 server; it is worth noting that the procedure for transmitting data to the server

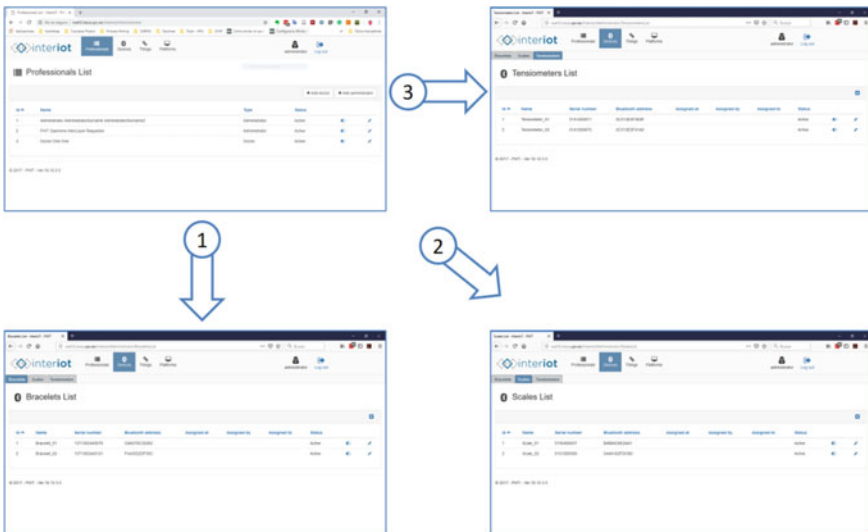


Fig. 10 Device management—PWT

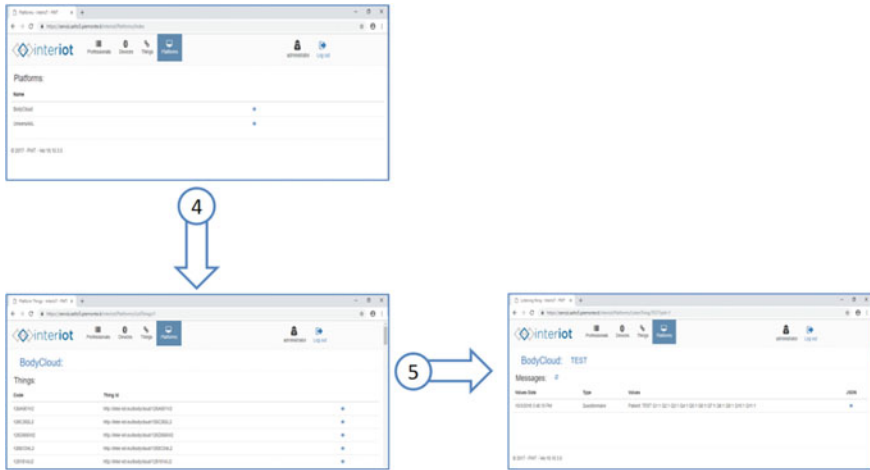


Fig. 11 Diagnostic tool—PWT

implies the automatic delete of the local data copy stored, once the reception and the correct feedback have been made.

3. The bi-directional communication protection between the mobile gateway application installed on the patient’s Smartphone and the remote server located at the ASL TO5 structure is guaranteed by using the HTTPS secure communication protocol that secures server authentication, privacy protection, the encryption and the integrity of the data exchanged between the communicating parties.
4. Within the communication mentioned in the previous point, it should be noted that biomedical data travels anonymously as they will be associated with an alphanumeric code generated by the server and therefore they are not directly associated with any patient personal identification data.
5. Update and maintenance service of the mobile gateway application is provided in case of errors and/or malfunctions of operation.

Regarding the INTER-Health pilot, to avoid the destruction and loss, even accidental, of data, the following steps are done:

1. Data exchanged between biomedical wireless devices and the mobile gateway application that resides on the patient’s Smartphone is protected from destruction and/or loss through proprietary mechanisms supported by device manufacturers; It is specified that such mechanisms is not, in any way, altered and/or modified by the UNICAL partner.
2. Data collected by the application is protected from destruction and/or loss through a temporary storage mechanism within the patient’s smartphone until the transmission to the ASLTO5 server is completed and until the receipt of a specific acknowledgment message activates a deletion procedure of the data copy stored in the local memory buffer.

3.3 General Data Protection Regulation

Likewise, the INTER-Health pilot has been subjected to an ethical board examination, according to the Article 6 of Legislative Decree 24 June 2003, no. 211 [28] to carry out clinical trials is mandatory to request authorization to the Ethics Committee. ASLTO5 got the approval to perform the study by the Ethical Committee of the A. O. U. San Luigi Gonzaga of Orbassano (Torino). Documentation needed is the following: privacy and security management, sensitive data, data controller, data processor, eternal data processor, information sheet, informed consent, data collection folder, research protocol, CE devices and datasheets, European project financial documents plus:

- Information sheet of the experimental study
- Declaration of consent for the participation in the experimental study
- Information on processing personal data
- Declaration of consent for processing personal data.

4 INTER-Health Execution

The piloting has been executed in Turin (Italy) at the premises of ASLTO5 hospitals during one year. It was divided into two groups: (1) 100 patients, who followed a traditional monitoring without IoT devices (control group) and (2) 100 patients with devices (experimental group), being these last ones the ones using the INTER-Health solution (Fig. 12).

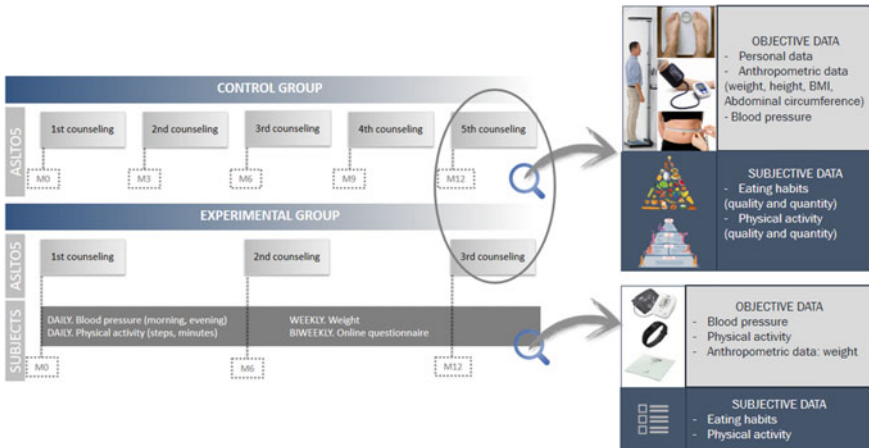


Fig. 12 INTER-Health in brief

Control group (CG) consisted on visits to the hospital each 3 months. HP interviewed patients to gather information related to their nutritional and physical activity habits (check-ups). Furthermore, anthropometric values were measured. It is worthy to mention that the results of these interviews were collected by hand (on paper) before INTER-Health solution.

Experimental group (EG) attended a first nutritional counselling at hospital premises, similar to the one done for the control group, and then each 6 months. The patient was assigned to a prevention program, which included a kit of health devices (weight scale, an activity bracelet, and when the patient is at risk of suffering high blood pressure, a sphygmomanometer). Thus, the health professional helped the patient to install a mobile app in his/her smartphone in order to get measurements from the kit and a questionnaire on eating habits and physical activity practice. Once at home, (s)he followed the prevention program by measuring the indicated anthropometric values.

All the data is sent from the mobile app to the server in ASLTO5, where HPs have access to all data.

4.1 Inclusion and Exclusion Criteria

Subjects had to meet inclusion criteria to be part of the Experimental Group. They had to be healthy patients greater than 18 years old; and the possibility to be available for at least 1 year to complete the whole piloting phase.

As exclusion criteria, subjects with food behavioural and/or serious psychiatric disorders, with cardiovascular diseases, tumours, diabetes and/or hypertension, and who required a unique appointment to have a dietetic or food advice, were discarded.

Subjects considered eligible were included in the non-invasive and health risk-free study. The decision to include a subject into the EG or CG depended on the presence or absence of a “Technological Prerequisite” that was to have an Android smartphone compatible with a concrete Bluetooth version. In which case, the absence of the “Technological Prerequisite” determined the inclusion of the subject in the CG, against those subjects with present prerequisite in their smartphone meant the inclusion of them into the EG.

In order to avoid drop out of the patients, eHealth solution has been designed to aid them by means of reminders and motivational messages.

4.2 Key Performance Indicators

4.2.1 Health Indicators

The indicators used in the study to evaluate the effectiveness of the Experimental Nutritional counselling are the following:

Body Mass Index (BMI). BMI is an objective measure that allows assessing the health state of subjects (underweight, normal weight, overweight, Level 1, Level 2 and 3rd level obesity), allowing making the diagnosis of overweight and obesity. This indicator can be used to monitor over time the health status of individuals assisted; it helps to HPs to state if the patient has maintained, improved or worsened his/her health status.

The effectiveness of the experiment can be measured by evaluating whether the treatment received by the group of persons assisted with Experimental Nutritional counselling (EG) compared to the one received by the group of persons assisted with the traditional nutritional counselling (control group) produced greater number of “successes”. Considering “Success”:

- In normal weight, maintaining weight at 6 months, in the period from the visit to the next control that takes place six months after the visit;
- In overweight patients, a decrease in six months by at least 5% of the BMI in the period following the Visit to the next control that takes place six months after the visit;
- In 1st level and 2nd level obese patients, a decrease in 12 months at least 5% of the BMI in the period following the Visit to the next control that takes place 12 months after the visit;
- In 3rd level obese patients, a decrease in 12 months at least 10% of BMI in the period following the Visit to the next control that takes place 12 months after the visit.

Waist Circumference. It is another method to diagnose overweight and obesity, values greater than 94 cm in men and 80 cm in women are considered high risk factor level to develop cardiovascular diseases. This indicator can be used to monitor over time the health status of individuals assisted at the Nutritional Outpatient.

BMI and CV related chronic risk pathology. The correlation between BMI and CV gives a greater indication of the subject’s health and in particular about the risk factors associated with overweight and obesity such as type 2 diabetes, hypertension, cardiovascular diseases, and stroke.

Physical activity reported by the subject during the visit to the Nutritional Outpatient is a subjective measure, that when it is not correct (physical inactivity) becomes a common risk factor at the basis of major chronic diseases. So the amount (hours/daily and hours/week) and the type of physical activity practiced (no type of activity; light activity, moderate and intense) related to the subject, it will become an indicator that can allow the medical staff to control during the various checks whether this lifestyle is maintained, improved or worsened. In particular, for the subjects of the “Experimental Group” the physical activity will be used as an indicator of steps number and physical activity duration recorded by wearable mobile devices referring to the 10,000 steps to be performed on a daily and 150 min a week.

Eating habits reported by the subject during the visit at the Nutritional Outpatient is a subjective measure, which is a lifestyle that when it is not correct (incorrect diet

and high-calorie) becomes a common and modifiable risk factor at the basis of major chronic diseases. So eating habits such as quality of foods (various food groups), amount of food consumed daily/weekly and daily frequency of consumption of the main meals (breakfast, lunch, dinner and snacks) related to the subject will become an indicator that may allow health staff to verify, during the various checks, if that lifestyle is maintained, improved or worsened.

Drop-out rate of the group of persons assisted with the experimental nutritional counselling (Experimental Group) compared to the one received by the group of persons assisted with the traditional nutritional counselling (control group), can express the effectiveness of the trial. We will use two quantitative indicators of Drop-out:

- Absolute quantity Drop-Out Indicator = N° of users leaving for study/12 months;
- Relative quantity Drop-Out indicator = N° of users leaving by choice from the study/12 months/number of followers.

Number of patients connected to INTER-Health

INTER-Health pilot is tested with at least with 100 real patients. KPI value is obtained by observing the number of patients registered in the health platform that actually use the mobile application.

Results

100 subjects (77 females and 23 males) were recruited in the **Control Group**. The average age of the subjects recruited is 47 years:

- 1% of patients were in a condition of underweight with BMI $<18.5 \text{ kg/m}^2$,
- 12% of patients were in normal weight with a body mass index of between 18.5 and 24.9 kg/m^2 ;
- 33% of subjects were overweight with BMI between 24.9 and 29.9 kg/m^2 ;
- 54% of subjects were obese with a Body Mass Index greater than 30 kg/m^2 .

The level of risk of developing cardiovascular disease in relation to waist circumference values (CV) was 14% of subjects moderately increased and 74% significantly increased.

Subjects presenting a risk (from high to extremely high) of developing chronic-degenerative diseases in relation to the values of BMI and CV were 86% of the total sample observed.

44% of subjects reported daily physical activity lasting at least 30–60 min, while 17% of subjects last longer than 60 min.

9% of subjects reported to practice structured physical activity for a duration of at least 30–60 min, while 25% of subjects last longer than 60 min.

Regarding eating habits:

- 88% of subjects reported consuming the 3 main meals,
- 67% consuming at least two portions of vegetables a day,
- 53% consuming 2/3 portions of fruit a day.

100 patients were recruited for the **Experimental Group** of which 67 females (67%) and 33 males (33%). 100 scales, 100 step bracelets and 25 sphygmomanometers were delivered. The average age of the subjects recruited is 46 years.

- 38% of patients were in normal weight with a BMI between 18.5 and 24.9 kg/m²;
- 30% of the subjects were overweight with a BMI of between 24.9 and 29.9 kg/m²;
- 32% of subjects were obese with a BMI greater than 30 kg/m².

25 subjects presented normal-high blood pressure values (systolic blood pressure ≥ 130 and/or diastolic blood pressure ≥ 85) and were then equipped with a sphygmomanometer for the provision of remote care at their homes.

The level of risk of developing cardiovascular diseases in relation to waist circumference values (CV) was 21% of subjects moderately increased and 62% significantly increased.

Subjects presenting a risk (from high to extremely high) of developing chronic-degenerative diseases in relation to the values of BMI and CV were 69% of the total sample observed.

22% of subjects reported daily physical activity lasting at least 30–60 min, while 11% of subjects last longer than 60 min.

35% of subjects reported to practice structured physical activity for a duration of at least 30–60 min, while 17% of subjects last longer than 60 min.

Regarding eating habits, 93% of subjects reported consuming the three main meals, 51% consuming at least two portions of vegetables a day, 36% consuming 2/3 portions of fruit a day.

The subjects recruited for the Control Group who continued the experimentation for a period of about 1 year were 43 of which 34 females (79%) and 9 males (21%). The Control Group presents a 57% drop out rate in 12 months.

At the end of the T12 experimentation, 26% of subjects are in normal weight with a body mass index (BMI) between 18.5 and 24.9 kg/m²; 28% of overweight subjects with a BMI of between 24.9 and 29.9 kg/m²; 46% obese subjects with a BMI greater than 30 kg/m². The state of health of the sample observed, based on the IMC, has improved.

The level of risk of developing cardiovascular diseases in relation to waist circumference values (CV) results for 16% of subjects moderately increased and for 63% significantly increased.

Subjects presenting a risk (from high to extremely high) of developing chronic-degenerative diseases in relation to the values of BMI and CV are 79% of the total sample observed. 53% of subjects reported daily physical activity lasting at least 30–60 min, while 16% of subjects last longer than 60 min.

19% of subjects reported to practice structured physical activity for a duration of at least 30–60 min, while 19% of subjects last longer than 60 min.

As for eating habits, 98% of subjects reported consuming the three main meals, 79% consuming at least two portions of vegetables a day, 77% consuming 2/3 portions of fruit a day.

The subjects recruited for the Experimental Group who continued the experimentation for a period of about 1 year were 88 of which 59 females (67%) and 29 males (33%).

The Experimental Group has a 12% dropout rate in 12 months.

At the end of the T12 trial, it results that 32% of subjects are in normal weight with a body mass index (BMI) between 18.5 and 24.9 kg/m²; 42% of overweight subjects with a Body Mass Index of between 24.9 and 29.9 kg/m²; 26% obese subjects with a Body Mass Index greater than 30 kg/m². The state of health of the sample observed on the basis of the IMC is therefore improved.

The level of risk of developing cardiovascular diseases in relation to waist circumference values (CV) results for 29% of subjects moderately increased and for 56% significantly increased.

Subjects presenting a risk (from high to extremely high) of developing chronic-degenerative diseases in relation to the values of BMI and CV are 72% of the total sample observed.

43% of subjects reported daily physical activity lasting at least 30–60 min, while 29% of subjects last longer than 60 min.

45% of subjects reported to practice structured physical activity for a duration of at least 30–60 min, while 17% of subjects last longer than 60 min.

Regarding the eating habits, 99% of the subjects reported to consume the three main meals, 56% to consume at least two portions of vegetables a day, 68% to consume 2/3 portions of fruit a day.

In conclusion, the pilot was carried out successfully, most of the participants improved their health conditions, preventing the risk of suffering for chronic diseases [29].

4.2.2 Technical Indicators

Performance of the Professional Web Tool

This KPI measures the technical performance of the pilot system as perceived by professional users (PU). The responsiveness of the Professional Web Tool (PWT) is measured indirectly through the analysis of system log files. Parameters such as speed of SQL queries execution or HTTP response times is considered.

PWT has been developed following Model-View-Controller architectural pattern. PWT performance refers to the time that an action takes since a query is launched until the result is shown to the PU, then invested time is registered in the system.

PWT is divided into controllers. Each controller has defined different actions (methods). When an action is triggered, the controller executes a query in the database. Then it is prepared a model based on the obtained result. Finally, the model is sent to the view, which generates the HTML code to show the result to the PU.

The list of actions taken into account are the following:

- Login. PU login into the PWT

- `getPatientsList`. PU accesses to the Patient list screen
- `PatientsFolderGet`. PU accesses to the folder of a specific patient
- `AddCheckUpGet`. PU creates a new check up for a patient
- `AddCheckUpPost`. PU saves the data added to a new patient's check up
- `ViewCheckUp`. PU consults the data of an existing check up
- `EditCheckUpGet`. PU edits the data of an existing check up
- `EditCheckUpPost`. PU saves the modifications done to an existing check up
- `PrintCheckUp`. PU prints the data of an existing check up
- `viewQuestionnaires`. PU consults historic data of questionnaires reported by a patient
- `viewPreventionProgram`. PU consults the prevention program defined for a patient
- `viewWeightChart`. PU consults historical weight data of a patient
- `viewPhysicalActivityChart`. PU consults historical physical activity data of a patient
- `ViewBloodPressureChart`. PU consults historical blood pressure data of a patient
- `Logout`. PU logout

The final KPI value is the average of the total time of actions divided into the number of actions.

Body Cloud mobile app usage

It is important to measure the total amount of time spent by a patient in each screen of the mobile app. It indicates how long takes for a patient to use the app. Being more than 10 min per functionality and day may be that the adherence to the app is good but not much user friendly as expected.

Value of this KPI is obtained by addition of time spent in each screen of the app. Measured in the app itself. Measurement per day and functionality.

Professional Web Tool application usage

As in the case of the patient, the time spent by the health professionals in the PWT is also important to measure the adherence to the tool. Time spent by a PU in a patient counselling and without using INTER-Health solution is around 90 min.

Value of this KPI is obtained by addition of time spent in each screen of the app. Measured in the app itself and per patient.

In INTER-Health, patients are split in experimental and control group. Patients in the experimental group are using BC mobile app with the medical sensors, which implies that every day the patients send data to the PWT and have counselling each six months. Instead, in the control group, the patients visit doctors every three months and do not have any associated app neither medical sensors.

The time of usage of the tool may vary depending on the group a patient belongs. It is not the same when a PU is checking the profile of a patient or counselling her/him, either when the face-to-face visits are dilated in time or are the unique feedback from the patient.

It is easy to determine the time spent in the PWT, by using the list of actions described in “**Performance of the Professional Web Tool**”, when a PU is actively

working. However, there are moments where the PU interviews the patient or introduces data that is not evident how to quantify this time. For that reason, we introduce Process Mining techniques to recognize the different procedures followed by PUs.

The KPI is understood as the time that a professional dedicates to a patient during a counselling, where is not possible to do more than one counselling per day and patient. A counselling is described as a face-to-face visit of a patient to the hospital, where the professional interviews and checks the progress of that patient.

Process Mining allows identifying workflows followed by PUs and inferring the total time spent. The final KPI value is the median of all value obtained.

Results

Name	Metric	Target (T)	Current KPI value
Number of patients connected to INTER-Health	Number of patients	100	102
Performance of the Professional Web Tool	Seconds	< 5 s	0.07 s
Body Cloud mobile app usage	Minutes	> 10 min	10 min 40 s
Professional Web Tool application usage	Minutes	> 60 min	68.82 min

5 Conclusions

Integrated care is a new approach followed in health, which main objective is to provide a holistic service to patients. It implies to share data from different sources. In the health environment, there are many different actors, and usually each with its own independent system. The interoperability among these systems is necessary, although this should be done in a secure and robust way.

The goal of INTER-Health pilot was to demonstrate the need for a system that allows the exchange of data and messages among the different actors of a health operator. In this case, there is only one actor, with the need of integrating different e-Health solutions, but it can be extended with the integration of new ones (IoT platforms, e-Health applications/services, medical devices, etc.) to enhance the current portfolio offered by ASLTO5, demonstrating the success of INTER-IoT in a health environment.

References

1. Suzman, R., Beard, J.R., Boerma, T., Chatterji, S.: Health in an ageing world-what do we know? *The Lancet* **385**(9967), 484–486 (2015)
2. Fortino, G., Trunfio, P. (eds.): *Internet of Things Based on Smart Objects, Technology, Middleware and Applications*. Springer (2014)
3. Fortino, G., Gravina, R., Galzarano, S.: *Wearable computing: from modeling to implementation of wearable systems and body sensor networks* (2018)

4. Gravina, R., Fortino, G.: Wearable body sensor networks: state-of-the-art and research directions. *IEEE Sens. J.* (2020)
5. Li, S., Da Li, X., Zhao, S.: The Internet of Things: a survey. *Inf. Syst. Front.* **17**(2), 243–259 (2015)
6. Yin, Y., Zeng, Y., Chen, X., Fan, Y.: The Internet of Things in healthcare: an overview. *J. Ind. Inf. Integr.* **1**, 3–13 (2016)
7. Yacchirema, D., Sarabia-Jácome, D., Palau, C.E., Esteve, M.: System for monitoring and supporting the treatment of sleep apnea using IoT and big data. *Pervasive Mob. Comput.* **50**, 25–40 (2018)
8. Cebul, R.D., Rebitzer, J.B., Taylor, L.J., Votruba, M.E.: Organizational fragmentation and care quality in the US healthcare system. *J. Econ. Perspect.* **22**(4), 93–113 (2008)
9. Iroju, O., Soriyan, A., Gambo, I., Olaleke, J.: Interoperability in healthcare: benefits, challenges and resolutions. *Int. J. Innov. Appl. Stud.* **3**(1), 262–270 (2013)
10. Yacchirema, D., Gonzalez-Usach, R., Esteve, M., Palau, C.E.: Interoperability of IoT platforms applied to the transport and logistics domain. In: *Transport Arena Research Conference 2018*, TRA, Austria (2018)
11. Giménez, P., Molina, B., Palau, C.E., Esteve, M.: SWE simulation and testing for the IoT. In: *2013 IEEE International Conference on Systems, Man, and Cybernetics*, pp. 356–361 (2013)
12. World Health Organization: Preventing chronic diseases: a vital investment. https://www.who.int/chp/chronic_disease_report/en/. Accessed Dec 2015
13. Global status report on noncommunicable diseases 2010. https://www.who.int/nmh/publications/ncd_report2010/en/. Accessed Oct 2015
14. Regione Piemonte. Azienda sanitaria locale TO5. <https://www.aslto5.piemonte.it/>
15. Aloï, G., Caliciuri, G., Fortino, G., Gravina, R., Pace, P., Russo, W., Savaglio, C.: Enabling IoT interoperability through opportunistic smartphone-based mobile gateways. *J. Netw. Comput. Appl.* **81**, 74–84 (2017)
16. Pace, P., Gravina, R., Aloï, G., Fortino, G., Fides-Valero, K., Ibáñez-Sánchez, G., Traver, V., Palau, C.E., Yacchirema, D.C.: IoT platforms interoperability for active and assisted living healthcare services support. In: *Global Internet of Things Summit, GIOTS 2017*, Geneva, Switzerland, 6–9 June 2017, pp. 1–6. IEEE (2017)
17. UniversAAL IoT: Home. <https://www.universaal.info/>
18. Ram, R., Furfari, F., Girolami, M., Ibáñez-Sánchez, G., Lázaro-Ramos, J.-P., Mayer, C., Prazak-Aram, B., Zentek, T.: UniversAAL: provisioning platform for AAL services. In: *Ambient Intelligence-Software and Applications*, pp. 105–112. Springer (2013)
19. Yacchirema, D.C., Sarabia-Jácome, D., Palau, C.E., Esteve, M.: A smart system for sleep monitoring by integrating IoT with big data analytics. *IEEE Access* **6**, 35988–36001 (2018)
20. Make it ReAAL: Home. <http://www.cip-reaal.eu/home/>
21. Fortino, G., Parisi, D., Pirrone, V., Di Fatta, G.: BodyCloud: a SaaS approach for community body sensor networks. *Future Gener. Comput. Syst.* **35**, 62–79 (2014)
22. Wang, Z., Donghui, W., Gravina, R., Fortino, G., Jiang, Y., Tang, K.: Kernel fusion based extreme learning machine for cross-location activity recognition. *Inf. Fusion* **37**, 1–9 (2017)
23. Fortino, G., Garro, A., Russo, W.: Achieving mobile agent systems interoperability through software layering. *Inf. Softw. Technol.* **50**(4), 322–341 (2008)
24. Fortino, G., Guerrieri, A., Bellifemine, F.L., Giannantonio, R.: SPINE2: developing BSN applications on heterogeneous sensor nodes. In: *IEEE Fourth International Symposium on Industrial Embedded Systems, SIES 2009*, Ecole Polytechnique Federale de Lausanne, Switzerland, 8–10 July 2009, pp. 128–131. IEEE (2009)
25. Iyengar, S., Bonda, F.T., Gravina, R., Guerrieri, A., Fortino, G., Sangiovanni-Vincentelli, A.L.: A framework for creating healthcare monitoring applications using wireless body sensor networks. In: *Panchanathan, S., Gupta, S. (eds.) 3rd International ICST Conference on Body Area Networks, BODYNETS 2008*, Tempe, Arizona, USA, 13–15 Mar 2008, p. 8. ICST (2008)
26. Bellifemine, F.L., Fortino, G., Guerrieri, A., Giannantonio, R.: Platform-independent development of collaborative wireless body sensor network applications: SPINE2. In: *Proceedings of the IEEE International Conference on Systems, Man and Cybernetics*, San Antonio, TX, USA, 11–14 Oct 2009, pp. 3144–3150. IEEE (2009)

27. Pace, P., Aloï, G., Caliciuri, G., Gravina, R., Savaglio, C., Fortino, G., Ibáñez-Sánchez, G., Fides-Valero, A., Bayo-Monton, J., Uberti, M., et al.: Inter-health: an interoperable IoT solution for active and assisted living healthcare services. In: 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), pp. 81–86. IEEE (2019)
28. European Network of Research Ethics Committees. EUREC—Italy. <http://www.eurecnet.org/information/italy.html>
29. Gulino, M., Maggi, C., Costa, A., Mortara, M., De Luca, I., Minutolo, M., Uberti, M., Bernini, L., Corona, M., Maio, F., et al.: Mobile health: studio pilota sul “monitoraggio decentralizzato ed in mobilità degli stili di vita” nell’ambito del progetto europeo “interoperabilità di piattaforme eterogenee iot-inter-iot”

INTER-LogP: INTER-IoT for Smart Port Transportation



Pablo Giménez, Miguel Llop, Joan Meseguer, Fernando Martin,
and Antonio Broseta

Abstract In the transport chain there are involved several public and private companies including ports, terminals, hauliers companies, shipping lines, freight forwarders, customs, etc. The transport activity requires the exchange of data and documentation between these companies. However, each of them has its own system and it is not easy to share data. This data can be used to provide better services to their clients. This chapter presents the demonstration of the INTER-IoT components in a real environment in the port and logistic domain. Three of the main actors in the transport chain are sharing data in the port of Valencia. With the data provided, we defined three different scenarios focused on access control and traffic, dynamic lighting, and wind gusts detection. With this pilot we have shown the benefits of sharing data in the port and logistic sector using INTER-IoT.

1 Introduction

Port environments are usually a technological level lower than other industrial sectors [1, 2]. That is why they are large areas with a huge potential for deploying IoT platforms [3, 4]. However when such platforms are deployed are typically not interoperable and are managed and used by the operator stakeholder (e.g. shipping line or container terminal operator). Increasing ability to track the location of smart objects has enabled the monitoring of traffic flows (save time and reduce congestion) as well as the provision of new location-based services (LBS), have enhanced the effectiveness and interoperability of smart transportation management systems (STMS). Real-world application scenarios are needed to derive requirements for software architecture and functionalities of future-generation STMS in the context of IoT. The deployment of IoT technologies can provide future STMS with huge volumes of real-time data that need to be linked, aggregated, communicated, anal-

P. Giménez (✉) · M. Llop · J. Meseguer
Fundación Valenciaport, Valencia, Spain
e-mail: pgimenez@fundacion.valenciaport.com

F. Martin · A. Broseta
CSP Iberian Valencia Terminal, Valencia, Spain

© Springer Nature Switzerland AG 2021
C. E. Palau (eds.), *Interoperability of Heterogeneous IoT Platforms*, Internet of Things,
https://doi.org/10.1007/978-3-030-82446-4_9

ysed, and interpreted [5]. However, being individual, self-contained services, these offerings fall short in terms of their seamless integration into the network of business roles conjointly performing the activities required to execute the business processes or fulfil customer requirements, e.g., interoperability or communication needs [6].

Typical logistic operations within a port can be enhanced due to interoperability, e.g. parking space availability, container status, availability of unloading/loading capabilities at the container terminals, ETAs (Estimated Time of Arrival), and remaining driving times. Currently, these operations have to be performed independently and with a complete lack of interoperability between components and platforms. [7] Additionally, the increase in utilization of vehicle/infrastructure electronics and communications raises security and privacy issues that, if left unaddressed, could jeopardise the wider deployment of IoT platforms. Security in a transport context seeks to prevent acts of unlawful interference against freight, goods or the transport infrastructure. There are several mechanisms, architectures, procedures and protocols in the state of the art to achieve security and privacy in services related with transportation at national and international level that need to be considered in case IoT interoperability [8, 9].

INTER-IoT framework applied in the port sector, INTER-LogP, is designed and built to specifically accommodate the communication and processing needs of trucks, containers and sensors. This use case test M2M communications of objects in a way that it can be monitored without any human intervention by different IoT architectures [3, 10, 11], generating a progress beyond the state of the art in the information management within the port premises and the business process. This facilitate and optimize transport operations and intermodal changes from origin to destination through different logistics nodes and substituting papers and barcodes with automatic M2M communications between objects.

2 Main Actors

The goal of INTER-LogP pilot is to demonstrate the need for a system that allows the exchange of data and messages among the different actors of the port community. In this case, as can be seen in Fig. 1, there are three main actors: the port, the terminal and the haulier company. INTER-IoT has to provide interoperability between the IoT platforms of the port and the terminal, and give access to other devices from other companies, like trucks.

Both the port and the terminal have a large number of sensors and devices that produce large amounts of data, which can be interesting for other entities. Furthermore, they need data from other companies to provide a better services to their clients [12, 13].

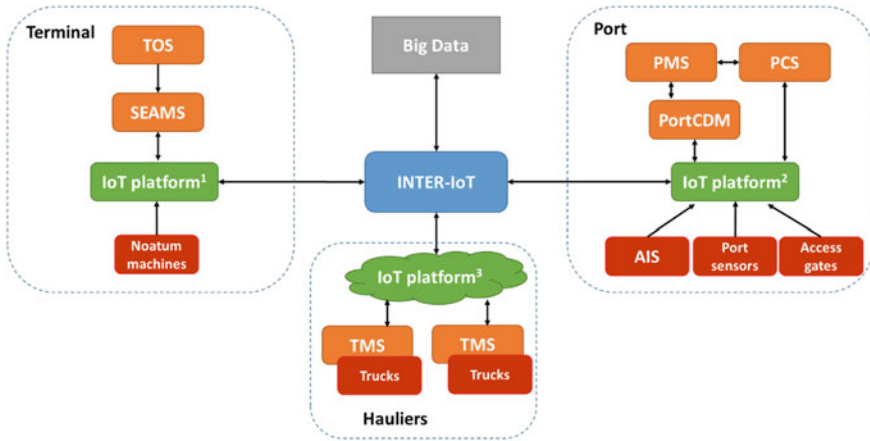


Fig. 1 INTER-LogP high-level pilot design

2.1 Port Authority

The port authority has a large number of sensors distributed throughout the port that provides data for management and operation. Most of that data is confidential, but other can be shared, adding value to other companies.

The architecture for providing interoperability from the legacy infrastructure is the one that can be seen in Fig. 2. Currently, the port authority has a common database where all the data is stored coming from different systems (in red). It uses WSO2 to provide an IoT architecture in two ways: data in real time through the Message broker and historic data through the Data services server and Enterprise Service Bus [14].

Because the port has its own platform, the integration with the INTER-IoT is done through the middleware. It needs a bridge in the middleware layer in order to interoperate with order platforms. The integration of new devices, such as the light controllers, is done using the INTER-IoT gateway to connect them with the IoT platform.

The port authority of Valencia has a virtual machine in which its IoT platform is deployed (in green). The platform has an API, which provide services to interoperate with it. The port bridge developed and deployed in INTER-MW, is using this API to subscribe to observations and act on the sensors.

The IoT platform is based on WSO2, an open source service-oriented architecture (SOA) middleware. It is designed with independent components, so it can be adapted for a lean targeted solution to enterprise applications. WSO2 products use Java technology and are built on top of WSO2 Carbon, a SOA middleware platform. Carbon makes use of Apache Axis2 and encapsulates SOA functionality such as data services, business process management, ESB routing/transformation, rules, security, throttling, caching, logging and monitoring [15].

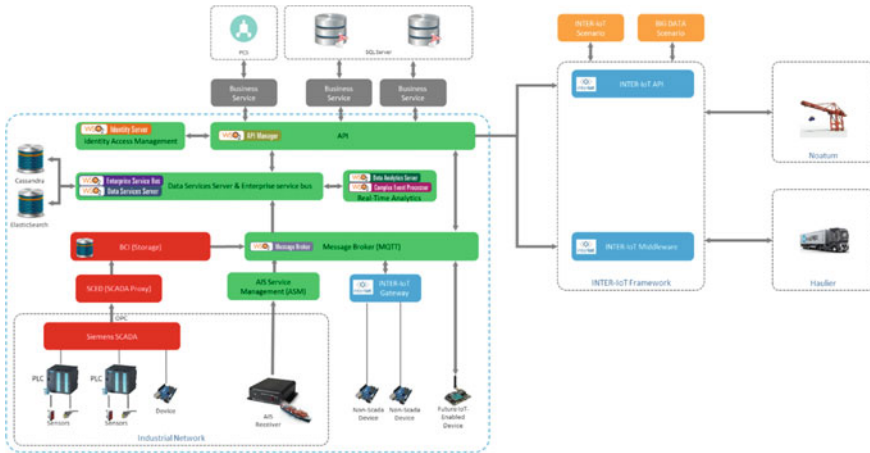


Fig. 2 Port IoT platform and integration

Not all components are used as stand-alone implementations. Many of them are used to supplement the capabilities or add functionality to an implementation of the Enterprise Service Bus. The main WSO2 components deployed in the IoT platform are:

Integration

- Enterprise Service Bus: Allows developers to connect and manage systems and software in accordance with SOA Governance principles.
- Data Services Server: Provides a Web service interface for data stores.
- Message Broker: Translates, validates and routes messages between systems.

API Management

- API Manager: API management platform for creating, deploying and managing APIs to expose data and functionality of backend systems.

Identity Management and Security

- Identity Server: Connects and manages multiple identities across applications, APIs, the cloud, mobile, and Internet of Things devices.

Management and Governance

- App Manager: Facilitates the process of creating, deploying and managing applications.

Analytics

- **Data Analytics Server:** Real-time, batch, interactive and predictive analytics using enterprise data.
- **Complex Event Processor:** Real-time event processing and detection. Identify patterns from multiple data sources, analyse their impacts. Uses WSO2 Siddhi and Apache Storm.

2.2 Container Terminal

NOATUM is the biggest container terminal in the port of Valencia and in the Mediterranean. The correct management of resources in a container terminal implies the monetarization of all the machinery in the yard, to be able to manage the resources properly. For that reason, in the NOATUM terminal each of the machines (vehicles, cranes, etc.) provide massive data about up to 80 sensors per machine each second.

There are around 300 monitored devices such as: STS (Sea-To-Shore) cranes, RTG (Rubbed-Tyred Gantry) cranes, Reachstackers, ECH (Empty Container Handler), TT (Terminal Tractor) and dynamic lighting on lamp posts.

So far, data were polled every second from around 200 machines and inserted into an SQL Server relational database, arising obvious scalability issues. With the deployment of an IoT platform, now data is sent from the machinery to the IoT Platform in two ways. Legacy sensors are collected once per second and inserted in the IoT Platform. New IoT devices are configured to send directly through MQTT or REST interfaces real-time data [16, 17]. In addition, the data is stored in a non-relational database, providing faster access to information. This architecture can be seen in the Fig. 3.

As in the case of the port, the IoT platform of the terminal is integrated with INTER-IoT through the middleware layer and the API layer, so a specific bridge was developed.

The container terminal has its own server with its IoT platform. They are mainly interested in knowing the estimated time of arrival of the truck to the terminal to be able to manage its yard resources. Furthermore, the terminal gives access to other companies to some of their own data, such as the entry and exit of trucks by their gates.

2.3 Haulier Company

Haulier companies have a large fleet of trucks, which access the port daily. These companies usually have fleet management systems in order to receive data from the truck. However, they are not exchanging any data with the port or the terminal, only the proper documentation.

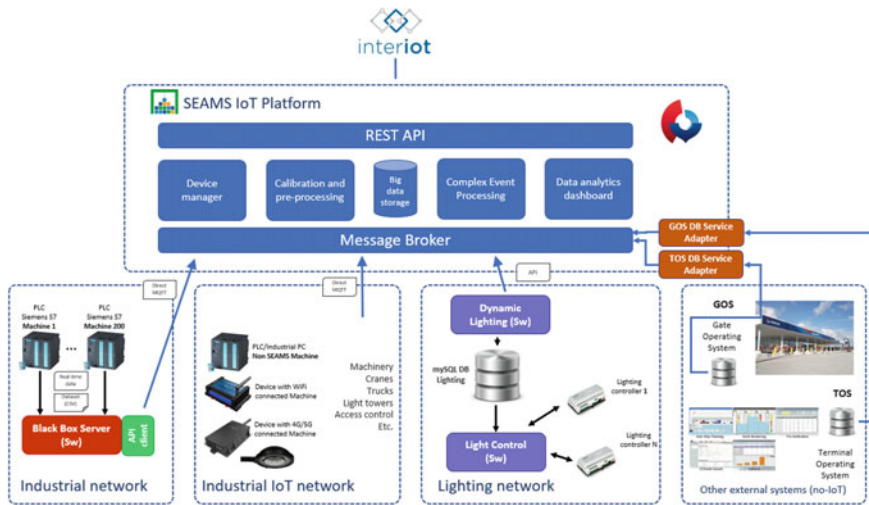


Fig. 3 Terminal IoT platform and integration

For the pilot we have designed an alternative system where each truck has a mobile app (MyDriving) installed in a mobile or a tablet that acts as a bridge between the vehicle and the IoT platform of the haulier company. All the devices in the truck and the driver send the data to the IoT platform through the movie app via Bluetooth.

Although there are big transport companies, there are other that do not have the necessary human resources have and maintain servers in their premises. That is the reason why the haulier company in the pilot has an Azure IoT platform in the cloud [18], where their trucks send all the data from the truck. These data is accessible to other companies as long as they are authorized and certain conditions are met, such as being inside the port area.

The main Azure modules used in IoT platform are:

- The IoT Hub is a managed service, hosted in the cloud that acts as a central message hub for bi-directional communication between INTER-IoT and MyDriving.
- Azure App Service is the service that enables you to build and manage your devices and applications.
- MyDriving is a mobile app that allow you to send data form your car to the IoT platform.

3 Use Cases

During the project, we have defined three scenarios where the INTER-LogP architecture has been tested. These scenarios demonstrate some of the products developed during the project in some relevant systems in the port. These scenarios are:

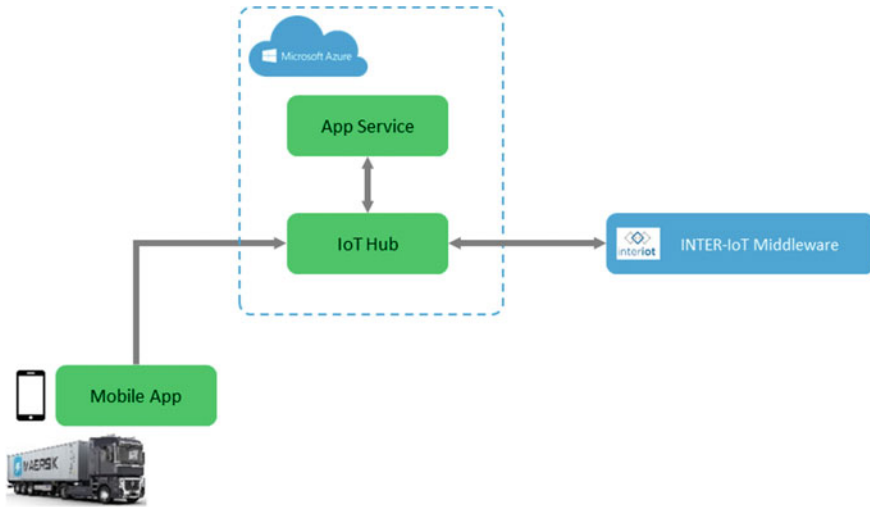


Fig. 4 Haulier company IoT platform and integration

- Pilot IoT access control, traffic and operational assistance
- Pilot Dynamic lighting
- Pilot Wind gusts detection

Furthermore, other scenarios were defined in collaboration with the third parties of INTER-IoT (Fig. 4).

3.1 Scenario IoT Access Control, Traffic and Operational Assistance

The main objective in this scenario is a service to monitor the gate access in order to assist the operations at the port. Several systems are able to identify trucks and drivers using different devices. This information can be shared under certain predefined rules through interoperability between the IoT platforms involved. This information can be used to monitor the truck inside the port by the Port Authority platform (security and safety purposes) and to manage more efficiently resources in the terminal. This can also allow to avoid queues in the access gates to the port and the terminal [19].

The main benefits we can get from this scenario is to obtain data regarding queues, congestion and temporary distribution of traffic, to manage efficiently the resources. Other important data is the position of the trucks while they are inside the port facilities, for safety and security. All these data can be shared between the port authority and the port terminals to improve the operation (Fig. 5).

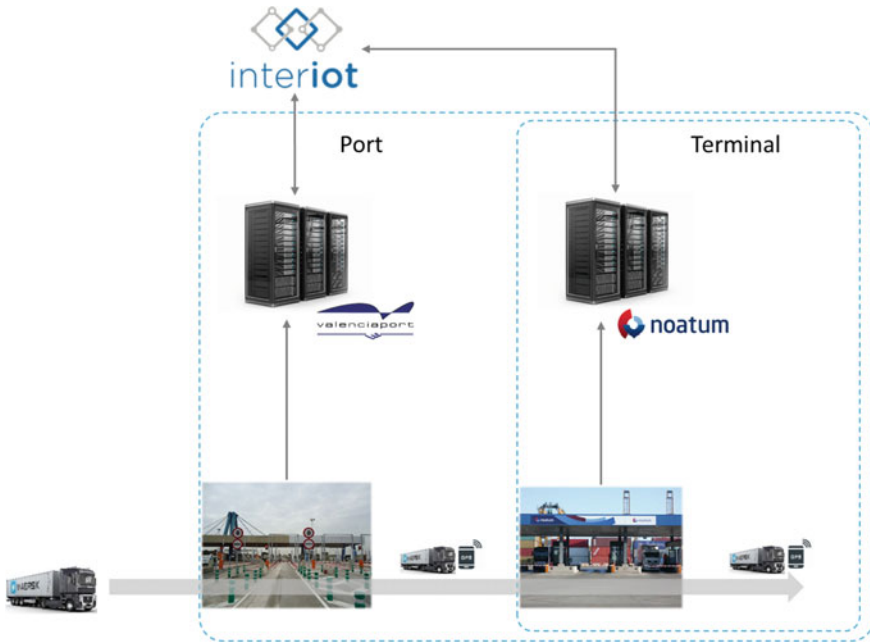


Fig. 5 High-level view of gate access scenario

3.2 Scenario Dynamic Lighting

The goal of this scenario is expand the smart illumination (Dynamic Illumination) in the yard of NOATUM for the rail yard. This area is not lit properly due some administrative issues.

In this case, the element that are triggering the activation of the lights is the NOATUM's personnel and machinery, but the lighting posts that are of the port authority of Valencia. So it is needed an exchange of data between both companies to illuminate it properly during the operation.

As can be seen in the Fig. 6, currently the rail yard (green area) is dimly lit with the container yard lighting posts. The objective is replace the road lighting posts (blue area) with low consumption lights and a dynamic lighting system, which receive data from the terminal to change the degree of illumination.

The opposite situation occurs on the railway switchgears at the Noatum's entrance (pink area). This area is not illuminated due to the little activity that is carried out. In this case, the lighting posts are of NOATUM but the operators are from the port authority (Fig. 7).

The dynamic lighting system is based in the GPS position of the NOATUM port equipment and long range PIR Sensors (presence sensors).



Fig. 6 Representation of the dynamic lighting scenario

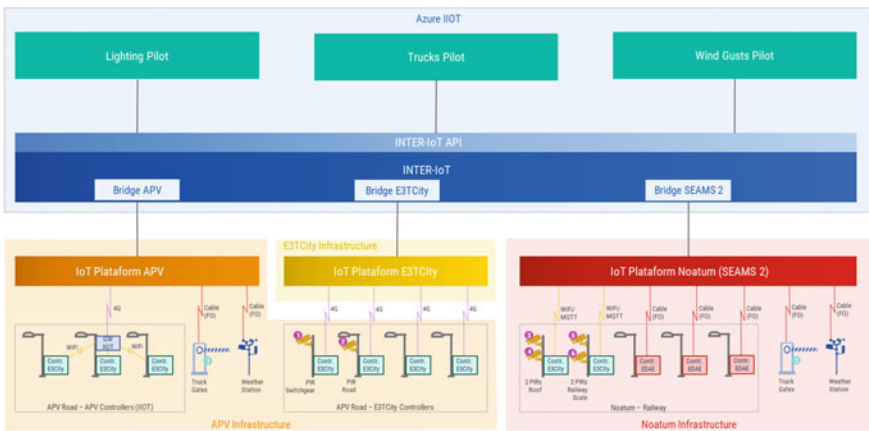


Fig. 7 High-level view of dynamic lighting scenario

The main benefits we can get from this scenario is an energy saving due to the adaptation of lighting to traffic and operation, and an improvement of the safety and security in the railway infrastructure due a better illumination.

This scenario was deployed with two different approaches. Firstly, a scenario was deployed only with the participation of the INTER-IoT partners. Then, there was a second version in the INTER-DOMAIN pilots, which integrates the technology of the third party involved in the Open call, called E3TCity.

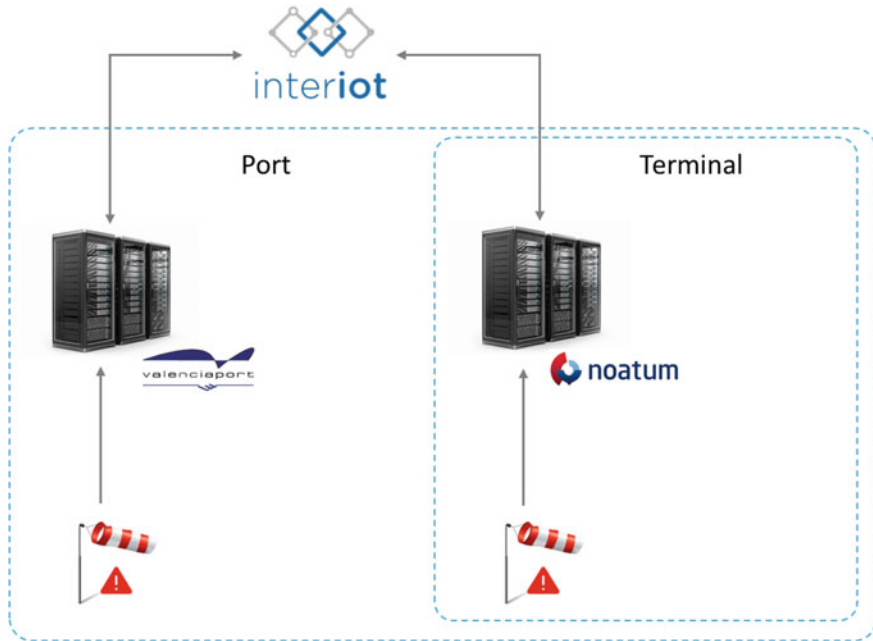


Fig. 8 High-level view of gate access scenario

3.3 Scenario Wind Gusts Detection

Currently, both the port authority and each of the container terminals in the port have anemometers to detect wind gusts. In situations where the wind speed exceeds a threshold, operations must be stopped for safety reasons. However, each terminal can only measure information in its premises, so that there is not data of surrounding areas, making impossible to predict when the stronger wind gusts can be expected. This makes the first detection of strong wind a risky situation for the operators, since the hazard is only detected when there is already active. If they were able to receive this information before, they would stop the operation in a safer way (Fig. 8).

The main benefit we can get from this scenario is improve operational safety at terminals, enabling an early awareness system that could end up in less accidents due to environmental phenomena.

4 Pilot Design

The initial high-level pilot design in Fig. 1 can be focused with the information from previous sections. In Fig. 9, the three IoT platforms are represented with the sensors that are sharing with companies. In the middle of the figure is INTER-IoT with the

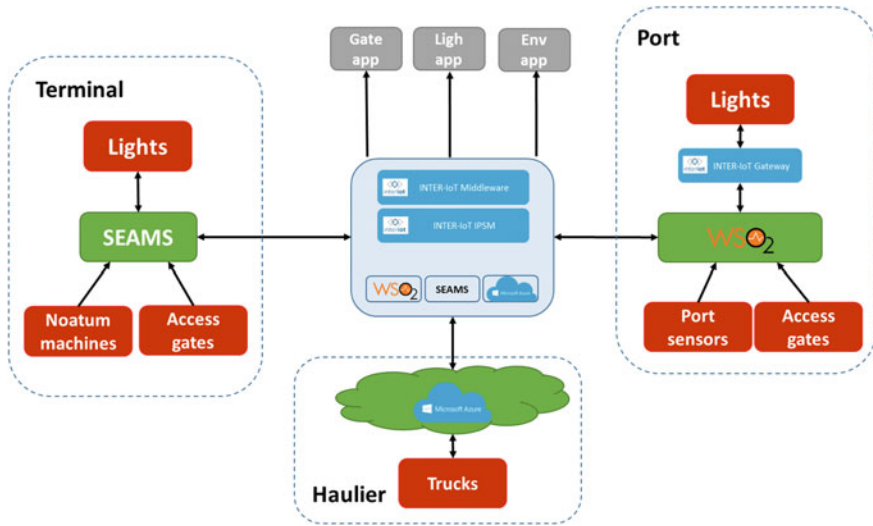


Fig. 9 INTER-LogP pilot design

main components used: INTER-MW, IPSM and the three bridges. There is an extra INTER-IoT product in the port environment. The lights and PIRs data is sent through the Gateway.

4.1 Ontology

Each IoT platform has its own ontology defined according to their operations. Port and terminal have at least three different set of data, which is used in the INTER-IoT pilots: gate access, environmental, and lighting. NOATUM also includes the machinery as a set of data. Finally, the haulier company includes data from the trucks and containers. For each one of them, we have defined a message and they are included in the ontologies.

4.2 Data Services

The data coming from the three IoT platforms is accessible in two different ways, depending on the type of data.

To access real-time data, a platform or application can subscribe to different devices through the MW. If the client has the proper permissions, it is subscribed to the platform broker to receive the data [20].

In the case of historical data, there are different APIs providing access to the data stored in the database. The port and the terminal IoT platform have an API manager where all the published APIs can be found.

4.3 Equipment

The INTER-LogP pilots reuse the existing systems and sensors in the port and the NOATUM terminal as much as possible. However, the pilots required the deployment of some additional equipment.

The main equipment included in INTER-LogP are servers for hosting the port and the terminal IoT platforms. The servers are connected to the different port legacy systems needed to collect the data.

However, in the case of the lighting scenario new equipment had to be deployed in order to monitor the two areas involved in the scenario: the rail yard in NOATUM and the railway switchgear area, which are shown in the Fig. 10.

The new equipment distributed between the port and the terminal is:

- 28 light bulbs
- 35 controllers for lights and PIRs
- 9 PIRs
- 4 routers
- 3 Wi-Fi access points
- 3 INTER-IoT gateways

In Fig. 7 is shown how this equipment is connected and controlled through the different IoT platforms.

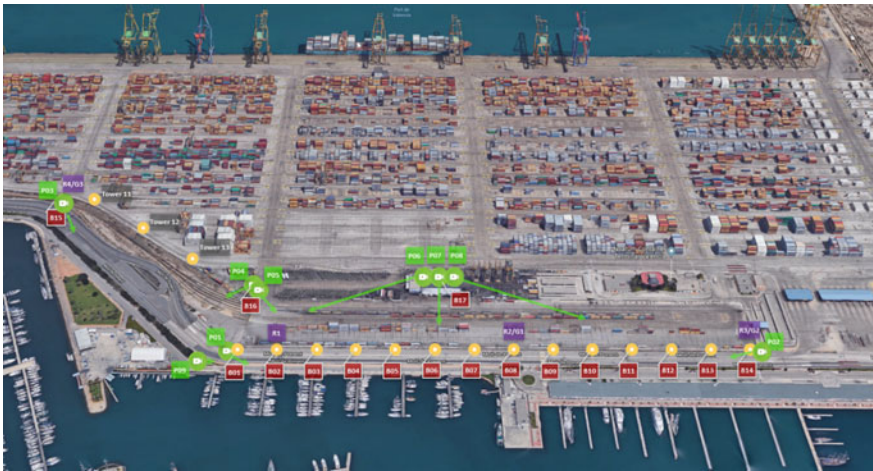


Fig. 10 Lighting scenario installation map

5 Pilot Execution

The first stage in the pilot execution was the deployment and configuration of the IoT platforms in each environment. After that, all the legacy systems and new devices were integrated and tested. These integration test were made following the FAT (Factory Acceptance Test) defined, in a controlled environment.

Then, we started to work in a real environment following the SAT (Site Acceptance Test). In this stage, we started to work with the real time data needed in the three scenarios.

Finally, we developed and tested the three scenarios, which are described below.

5.1 *Scenario IoT Access Control, Traffic and Operational Assistance*

The main objective of this scenario is verify the integration of all the components in the IoT access control, traffic and operational assistance scenario in the port and provide a tool to analyse real time data.

The process that follows the scenario is the following:

1. The truck continuously sends information to the haulier company. This information includes the position.
2. Upon arrival, the truck is detected by the port gates system and the associated data is sent to the port authority IoT platform.
3. The port IoT platform publishes the data to all the entities that are allowed to receive this data.
4. From this moment, the haulier IoT company starts to share the position of the truck with the port and the terminal.
5. When the truck is detected by the NOATUM gates system, the data is sent to the terminal IoT platform.
6. The terminal IoT platform publishes the data to all the entities that are allowed to receive this data.
7. All the data is gathered, analysed and represented in a dashboard owned by the port authority (Fig. 11).

All the data gather by the application is shown to the port authority operator to analyse the real time situation. In the dashboard, he can see the number of trucks and container accessing and leaving the port, and some figures with average time in the port and the time spent getting to the terminal. The last one is the most interesting, as it combines data from the port and the terminal to represent the time needed to arrive to the terminal [21] (Fig. 12).

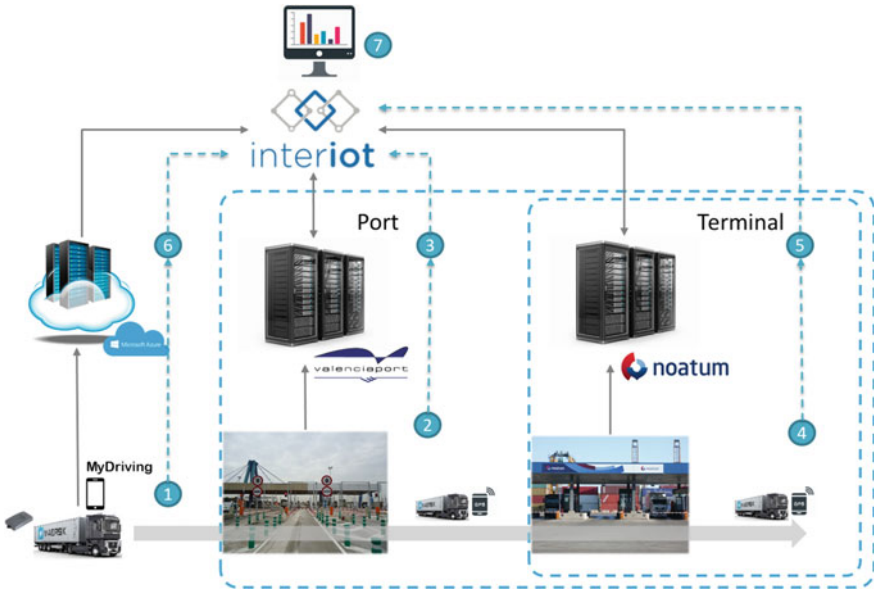


Fig. 11 Gate access scenario process

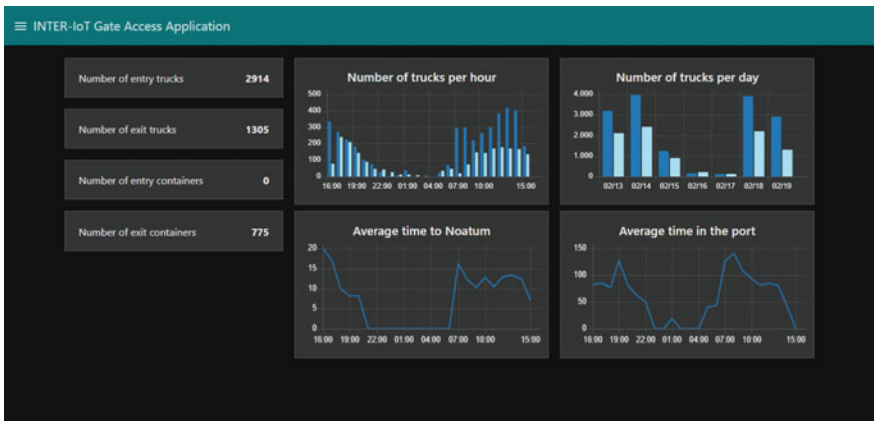


Fig. 12 Gate access scenario dashboard

5.2 Scenario Dynamic Lighting

The main objective of this scenario is verify the integration of all the components in the dynamic lighting scenario, and to develop a smart illumination (dynamic Illumination) system for the rail yard in the yard of NOATUM.



Fig. 13 Dynamic lighting scenario process 1

The process that follows the first situation is the following:

1. Activity is detected in the train yard (train, truck, or person) by PIRs located on the left corner (truck access) or on top of the building.
2. Upon detection, the associated data is sent to the terminal IoT platform to be processed. The light posts that depend on the terminal raise their light intensity.
3. At the same time, the data is published through INTER-IoT and, hence, shared with the port.
4. The two platforms managing some of the port streetlights receive the data and raise the light intensity (Fig. 13).

The process that follows the second situation is the following:

1. Activity is detected in the railway switchgear (train or person) by PIRs located at the beginning or at the end of the area.
2. Upon detection, the data is sent to the port IoT platform to be processed.
3. At the same time the data is published though INTER-IoT and, hence, shared with the terminal.
4. The terminal IoT platform receives the data and raises the light intensity in the area (Fig. 14).

The terminal yard operator has a tool where he can see all the devices in the terminal. He is able to filter the dynamic lighting system and see the level of light of each of the lamppost. Clicking in each of the devices it is possible to see the parameters of the device (Fig. 15).



Fig. 14 Dynamic lighting scenario process 2



Fig. 15 Dynamic lighting dashboard

5.3 Scenario Wind Gusts Detection

The main objective of this scenario is verify the integration of all the components in the wind gust detection scenario and to provide a service to share wind gust data to improve the safety.

The process that follows the scenario is the following:

1. The port weather stations detect wind data that is stored in the port IoT platform.
2. When the wind gust exceeds a threshold, the event is published through INTER-IoT.

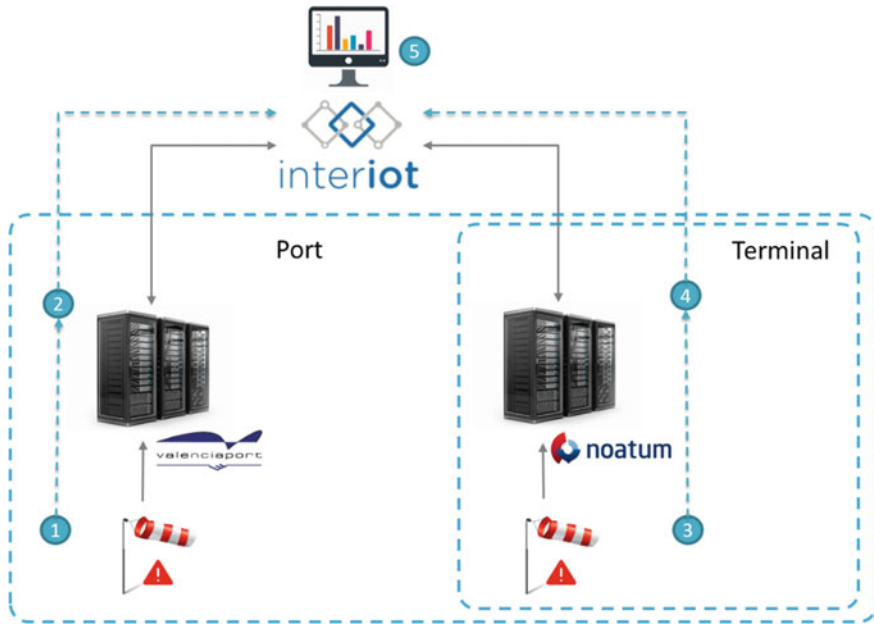


Fig. 16 Wind gusts detection pilot process

3. In the same way, NOATUM has its own weather station that are storing metrological data.
4. When a dangerous wind gust is detected, is also published.
5. All the data is gathered, analysed and represented in a dashboard (Fig. 16).

Port and terminal safety operator have a tool to get real time environmental data. This dashboard generates alarms when the wind gust exceeds a threshold, and he can stop the activity to avoid dangerous situations. There are different views with all the devices deployed in the three ports, figures with historical data, and alarm management (Fig. 17).

6 KPIs

During the pilot, we were able to gather some data, which was analysed to assess the performance of systems and the INTER-IoT components. Furthermore, in the validation plan, some indicators were defined to represent the results.

The list of defined KPIs are:

In the table we can see the number of scenarios deployed in the port and the number of devices connected, including the contributions from the third parties. Furthermore, all the technical indicators have been achieved (Table 1).

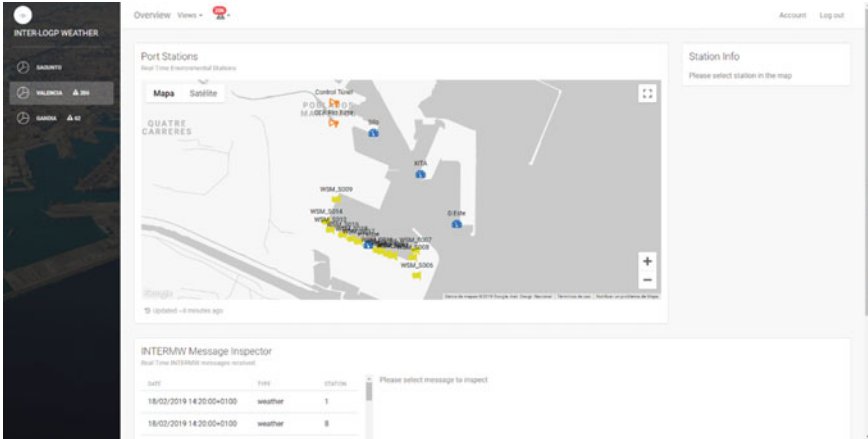


Fig. 17 Wind gust dashboard

Table 1 List of KPIs defined in the evaluation plan

KPI id	Name	Description	Metric	Target	KPI value
KPI.2.01	Port scenarios	Scenarios defined and deployed to test different developments	Number	4	$3+(1+10) = 14$
KPI.2.03	Number of objects connected to INTER-LogP	Number of devices and sensors connected to the different IoT platforms in INTER-LogP	Number	250	$61+182+2+(10+14) = 269$
KPI.2.04	Accuracy ETA vs ATA	Measurement about the accuracy between estimated time of arrival, versus actual time of arrival	Minutes	5	7.3
KPI.2.05	Activity detected in the railway area	Percentage of trains correctly detected by system (Arrival and Departure trains).	%	0.8	0.85
KPI.2.06	Trucks detected by system	Percentage of trucks correctly detected by system	%	0.8	0.81
KPI.2.07	Global events detected by system	Percentage of events correctly detected by system	%	0.8	0.83

7 Privacy and Security

7.1 Privacy and Confidential

The objective of this project is interoperability, so the exchange of data between components and platforms is mandatory. However, in the port sector this data could be sensitive to the owner's company, so agreements are necessary among the organizations to use the data only for the agreed purpose. Moreover, the information should not be shared with anyone else [22].

The data used in the INTER-LogP pilot is not critical or sensitive, oppositely to the health pilot, but it is confidential. The access has to be secure as this data could be used by competitors for market positioning.

Furthermore, all the INTER-IoT developments and platforms must adapt to the General Data Protection Regulation (GDPR) legislation that becomes enforceable on May 2018 [23]. For this reason, the data owner is always capable to give consent, see who is accessing their data and revoke the access permissions if necessary.

One of the lessons learnt regarding data sharing in a port environment is that several companies are reluctant to share their data. If these companies are not in the consortium agreement and they do not see a clear benefit to them, they do not want to participate [24, 25].

7.2 Security

In addition to privacy and confidentiality, all these processes require a high level of security. The security must be guaranteed in communications and in each of the intermediate components. This must be done through the use of secure and encrypted communication channels and with high security in the middleware [26].

In the port IoT platform there is an Identity Access management module that manages the security. A token is necessary to access the data through any of the APIs or the real-time data in the broker. The protocol used for this purpose is OAuth version 2.0. In fact, the client has to go through two steps before getting a valid token: first, it must get the client id and the secret key, and second, the client uses the client id and secret key to request a token using the OAuth2 Client Credentials Grant type.

Apart from OAuth2 access protocol, each client is required to establish a connection using SSL/TSL encryption protocol. Clients need a CA signed certificate in order to connect with the port IoT platform and thus exchange information encrypted by both sides.

8 Conclusions

In recent years, there is a need to share real-time data between different companies in order to offer new services to their clients. In the port environment, there are many different companies, each with its own independent system. Nowadays they only exchange some logistic documentation and not sensor data. This new exchange of data should be done in a secure and robust way.

The goal of INTER-LogP pilot was to demonstrate the need for a system that allows the exchange of data and messages among the different actors of the port community. In this case, there are three main actors: the port, the terminal and the haulier company. INTER-IoT provide interoperability between the IoT platforms of the port and the terminal, and give access to other devices from other companies, like trucks. We have demonstrated in three different scenarios the benefits of data exchange between companies in the port sector.

References

1. Carlan, V., Sys, C., Vanelslander, T. and Roumboutsos, A.: Digital innovation in the port sector: Barriers and facilitators. *Compet. Regul. Netw. Ind.*, 71–3 (2017)
2. DP World/The Economist Intelligence Unit: A turning point: The potential role of ICT innovations in ports and logistics. DP World, s.l. (2015)
3. Fortino, Giancarlo, Savaglio, Claudio, Spezzano, Giandomenico, Zhou, MengChu: Internet of Things as System of Systems: A Review of Methodologies, Frameworks, Platforms, and Tools. *IEEE Trans. Syst. Man Cybern. Syst.* **51**(1), 223–236 (2021)
4. Vermesan, O., Friess, P. (eds.): *Digitising the Industry Internet of Things Connecting the Physical*. River Publishers, Digital and Virtual Worlds (2016)
5. Fiosina, J., Fiosins, M., Müller, J.: Big Data Processing and Mining for the future ICT-based Smart Transportation Management System. *Jurnal Teknologi* **63**(3) (2013)
6. Giménez, P., Molina, B., Calvo-Gallego, J., Esteve, M., Palau, C.E.: I3WSN: Industrial intelligent wireless sensor networks for indoor environments. *Comput. Ind.* **65**(1), 187–199 (2014)
7. Sarabia-Jácome, D., Palau, C.E., Esteve, M., Boronat, F.: Seaport Data Space for Improving Logistic Maritime Operations. *IEEE Access* **8**, 4372–4382 (2020)
8. Fortino, G., et al.: Towards multi-layer interoperability of heterogeneous IoT platforms: the INTER-IoT approach. In: *Integration, Interconnection, and Interoperability of IoT Systems*, vol. 199–232. Springer (2018)
9. Broring, A., Zappa, A., Vermesan, O., Främling, K., Zaslavsky, A., Gonzalez-Usach, R., Szmeja, P., Palau, C., Jacoby, M., Zarko, I.P., Soursos, S., Schmitt, C., Plociennik, M., Krco, S., Georgoulas, S., Larizgoitia, I., Gligoric, N., García-Castro, R., Serena, F., Orav, V.: *Advancing IoT Platform Interoperability*, River Publishers (2018)
10. Fortino, Giancarlo, Russo, Wilma, Savaglio, Claudio, Shen, Weiming, Zhou, Mengchu: Agent-Oriented Cooperative Smart Objects: From IoT System Design to Implementation. *IEEE Trans. Syst. Man Cybern. Syst.* **48**(11), 1939–1956 (2018)
11. Claudio Savaglio, Maria Ganzha, Marcin Paprzycki, Costin Badica, Mirjana Ivanovic, Giancarlo Fortino: Agent-based Internet of Things: State-of-the-art and research challenges. *Future Gener. Comput. Syst.* **102**: 1038–1053 (2020)
12. Celtinkaya, B., Cuthbertson, R. et al.: *Sustainable Supply Chain Management, Practical Ideas For Moving Towards Best Practice*, Springer, p. 264 (2011)

13. Rodrigue, J.-P.: The geography of port terminal automation, PortEconomics website, <https://www.porteconomics.eu/the-geography-of-port-terminal-automation/>. Accessed Apr 2019
14. Fremantle, Paul. A reference architecture for the Internet of Things. WSO2 White paper, 2015
15. Fremantle, P., Aziz, B., Kopecký, J., Scott, P.: Federated identity and access management for the Internet of Things. In: International Workshop on Secure Internet of Things. Wroclaw, Poland **2014**, 10–17 (2014). <https://doi.org/10.1109/SIoT.2014.8>
16. Soni, D., Makwana, A.: A survey on mqtt: a protocol of Internet of Things (IoT). In: International Conference On Telecommunication, Power Analysis And Computing Techniques (ICTPACT-2017), vol. 20 (2017.)
17. Richardson, L.: RUBY, Sam. RESTful web services. O'Reilly Media, Inc. (2008)
18. Bansal, N.: Microsoft Azure IoT Platform. Designing Internet of Things Solutions with Microsoft Azure, pp. 33–48. Apress, Berkeley, CA (2020)
19. Yacchirema, D., Gonzalez-Usach, R., Esteve, M., Palau, C.E.: Interoperability of IoT Platforms applied to the transport and logistics domain. In: Proceedings of Transport Arena Research Conference 2018 (April 2018)
20. Fortino, G., Liotta, A., Palau, C., Gravina, R., Manso, M. (eds.) Integration, Interconnection, and Interoperability of IoT Systems, Springer (February 2017)
21. Belsa, A., Sarabia-Jacome, D., Palau, C.E., Esteve, M.: Flow-based programming interoperability solution for IoT platform applications. In: 2018 IEEE International Conference on Cloud Engineering (IC2E), pp. 304-309, Orlando (FL) (February 2018)
22. Pileggi, S.F., Palau, C.E., Esteve, M.: building semantic sensor web: knowledge and interoperability. In: Proceedings of the International Workshop on Semantic Sensor Web, Vol. 1: SSW, (IC3K 2010), pp. 15.22 (April 2010)
23. Seo, J., Kim, K., Park, M., Park, M., Lee, K.: An Analysis of Economic Impact on IoT Industry Under GDPR. Mobile Information Systems (2018)
24. Gizelis, C., Mavroeidakos, T., Marinakis, A., Litke, A., Moulos, V.: Towards a Smart Port: The Role of the Telecom Industry (2020)
25. Cavanillas, M., Edward Curry, J., Wahlster, W.: New Horizons for a Data-Driven Economy: A Roadmap for Usage and Exploitation of Big Data in Europe. Springer Nature (2016)
26. Hiro Gabriel Cerqueira, F., Timoteo de Sousa Junior, R.: Security analysis of a proposed Internet of Things middleware. Clust. Comput. **20.1**, 651–660 (2017)
27. Fortino, Giancarlo, Garro, Alfredo, Russo, Wilma: Achieving Mobile Agent Systems interoperability through software layering. Inf. Softw. Technol. **50**(4), 322–341 (2008)

IoT Ecosystem Building



Regel Gonzalez-Usach, Carlos E. Palau, Miguel A. Llorente, Roel Vossen, Rafael Vaño, and Joao Pita

Abstract A main obstacle for building the emerging IoT ecosystem is the current lack of standardized infrastructure and thus interoperability, hampering IoT platforms to inter-operate among them and causing a massive proliferation of vertical silos. Since interoperability is a critical feature for the creation of IoT ecosystems, the novel open interoperability solutions provided by the H2020 INTER-IoT project can very significantly promote the creation and expansion of IoT interoperable ecosystems, especially across domains. The road towards INTER-IoT ecosystem building is described in this chapter, noting the different IoT solutions and domains already addressed.

1 Introduction

Internet of Things (IoT) is a recent paradigm that is deeply transforming and revolutionizing our world [1]. Most organizations today are under a digital transformation, changing their business model through the use of IoT technologies and as a result, providing new critical value-producing opportunities and revenues. Many sectors in our modern world will face or are already facing a transformation through the IoT that will open up a new world of possibilities. For example, cities can take an immense advantage of IoT by adopting the smart city paradigm, which requires the interconnection and coordination of many systems. Also, industry is enormously benefiting from the Industry 4.0 paradigm, on which IoT has a critical role as a technology enabler.

Smart Digitalization involves creating, nurturing and maintaining of a vast ecosystem comprising devices, technology infrastructures, markets and industries world-

R. Gonzalez-Usach (✉) · C. E. Palau · M. A. Llorente · R. Vaño
Universitat Politècnica de València, Camino de Vera, 46022 Valencia, Spain
e-mail: regonus@dcom.upv.es

R. Vossen
Neways Technologies B.V., EP Son, The Netherlands

J. Pita
XLAB doo, Pot za Brdom 100, SI-1000 Ljubljana, Slovenia

© Springer Nature Switzerland AG 2021

C. E. Palau (eds.), *Interoperability of Heterogeneous IoT Platforms*, Internet of Things, https://doi.org/10.1007/978-3-030-82446-4_10

wide. It is not easy to achieve and present many challenges to overcome. A main obstacle for building the emerging IoT ecosystem is certainly the lack of standardized infrastructure that enables IoT systems to be interoperated across different industry verticals [2–4]. IoT systems are typically locked as vertical silos of information and associated services, constrained to a single platform and unable to interact with other verticals due to the differences between them regarding standards, formats, communication procedures, information models and semantics [5]. Horizontal integration is complex, and this fact hinders the creation of fruitful ecosystems, reduces the benefits from the use of the IoT paradigm and hampers the incipient evolution of IoT (e.g. Ambient Intelligent Environments, natural transparent human-oriented interfaces, integration with machine learning mechanisms, blockchain security and Artificial intelligence) [6].

In order to enable IoT horizontal integration, the Inter-IoT framework offers a valuable set of tools capable of overcoming the barriers of the inherent lack of interoperability in IoT at many different levels (i.e. device, network, middleware, data, semantics, application and services) and boost the disappearance of vertical silos. Those assets are an open cross-layer framework, an associated methodology and tools for enabling voluntary interoperability among heterogeneous IoT platforms. As a result, INTER-IoT interoperability solutions allow effective and efficient development of adaptive, smart IoT applications and services, atop different heterogeneous IoT platforms, spanning single and/or multiple application domains and creating significant added value. This interoperability approach is general-purpose and can be applied to any application domain and across domains, in which there is a need to interconnect IoT systems already deployed or add new ones [5].

The use of the INTER-IoT interoperability solution has been explored and applied for building thriving domain agnostic IoT ecosystems. First, the development of the INTER-IoT pilots allowed to set the basis for ecosystem building in and across the domains of e-health, transport and logistics. Second, the launch of the project Open Call supported by the European Commission represented a key action to obtain the collaboration of key stakeholders of the IoT landscape, improve and validate the INTER-IoT framework, and boost the creation and enlargement of interoperable ecosystems. Third, the adoption of the INTER-IoT solutions in other innovative projects such as H2020 PIXEL and H2020 LSP ACTIVAGE enabled the creation of new ecosystems in new domains. Moreover, INTER-IoT benefited from the support and promotion of the IoT-EPI initiative for promoting ecosystem building across Europe, and from specific actions that allow to attract stakeholders and adopters (i.e. dissemination and communication activities and the INTER-IoT open source release).

This set of actions that lead to the creation and enlargement of an IoT domain-agnostic ecosystem are described in this chapter. In particular, this document is mainly focused on the Open Call impact enabling the creation of ecosystems on different domains. Pilots, already described in previous chapters, are indirectly referred through their relation to some open call projects.

2 INTER-IoT Ecosystem Creation

INTER-IoT has offered the latest solutions in the area of interoperability and service development support, addressing major interoperability needs of IoT platforms and IoT ecosystems [6]. IoT platforms require interoperability in multiple layers. IoT connectivity is the first step in making the Smart Objects collect valuable information and exploit the vast potential of IoT. It involves linking to sensors, actuators, Bluetooth, NFC, other low power short range communications, 4G, 5G, syntactic transformation of data formats, protocol conversion, and even pre-processing of data in order to deliver the IoT information collected by the Smart Objects through the Internet to upper systems such as IoT platforms or Cloud services. The access to IoT middlewares and APIs from IoT platforms, and the possibility of adapting the data formats and semantics of shared key information to the receiver's models in real time, will allow the possibility of sharing key information in real time and having all of them accessible in a single access point in order to create new applications and services. Key aspects to achieve across platforms in order to foster an interoperable ecosystem are semantic interoperability, resource access and resource discovery. Another layer in which interoperability and ecosystem growth is needed, is cross-platform service composition as IoT platforms provide native services that are not exploited externally. Additionally, new functions for the creation and management of Software Defined Networks (SDN) are useful to prevent the network congestion due to the exponential increase of data on the Internet, providing flexibility to the networks in case of peaks of traffic, and prepare the ground for 5G.

The outcomes of the Inter-IoT project are a set of interoperability tools that can provide tight interoperability at each layer of IoT platforms and systems (i.e. INTER-Layer). Those tools can be used as a whole or separately. This framework can be used through INTER-FW, a software that provides a usable visual interface for launching this tools in a virtualized way, providing instantaneous deployment regardless the underlying technologies and operating system, monitoring of connected IoT platforms through INTER-IoT solutions, and an integrated API of all INTER-IoT layer solutions, which simplifies the access for developers and users. Moreover, INTER-IoT also provides a guiding methodology to easily implement interoperability solutions.

In particular, the INTER-IoT assets for layered interoperability are:

- **at device level**, a smart gateway (i.e. INTER-IoT Gateway) [6] allows to connect very dispair and heterogeneous IoT devices, with constrained resources and very limited memory and wireless connectivity. These type of devices are the ones that typically constitute IoT networks. Inter-IoT smart gateway connects those devices with the Internet, performing operations of data format adaptation, pre-processing and protocol conversion. The gateway is decoupled in a real and a virtual part, and the latter performs network functions and is capable to connect to Software Defined Networks (SDN) and support Network Function Virtualization (NFV) functionality, **at a network level**. This smart gateway allows the seamless inclusion

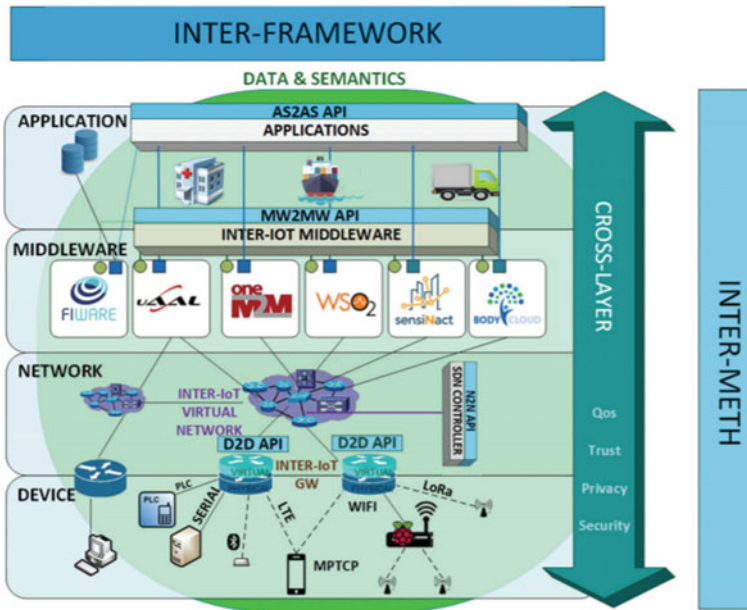


Fig. 1 INTER-IoT layered interoperability solutions

of new IoT devices and their interoperation with already existing heterogeneous ones, allowing a fast growth of smart objects ecosystems (Fig. 1).

- **at middleware level**, the INTER-MW engine [7], allows to connect seamlessly to any IoT platforms and allows communication on top, or between them, despite of the different communication interfaces, standards, data format and information model. The combination of INTER-MW with the Inter Platform Semantic Mediator (IPSM) component [6] from the semantic layer provides real-time data format and semantic information model adaptation of the data from platforms, translating in real-time the information from the sender's semantics into the receivers' semantics, in order to make information understandable by the receiver, and thus providing syntactic and semantic interoperability. INTER-MW provides access to connected resources and performs resource discovery.
- **at semantic level**, as mentioned, the IPSM [8] which is able to provide semantic interoperability by performing real-time semantic translations between connected entities. It can be used separately from INTER-MW, and it employs a central ontology specifically designed for IoT in the INTER-IoT project, called GOIoTP,¹ which extends W3C SSN/SOSA.²

¹ <https://inter-iot.github.io/ontology/>.

² <https://www.w3.org/TR/vocab-ssn/>.

- **at application level**, the AS2AS graphic tool [6] allows to easily combine application and services from different IoT platforms, creating sequences of service composition and service execution in cascade on-demand.
- **at all layers** (i.e. cross-layer), it is aimed to guarantee non-functional aspects that must be present across every layer: trust, security, privacy, and quality of service.

Those INTER-IoT interoperability solutions can very significantly boost the creation and growth of IoT ecosystems, by providing horizontal integration of vertical isolated systems and IoT interoperability at all levels. The ecosystems would be domain-agnostic, as the INTER-IoT technology can be applied to any domain, or across domains, in which there is a need of interoperability [9].

It should be noted that the openness of the solutions is also a key enabler of ecosystem building, as this feature incentivizes the adoption, support and future extension, attracting developers and stakeholders, as well as an open-source community.

3 INTER-IoT Open Call

In a continuously evolving IoT landscape with new protocols, platforms, communication and network technologies, ontologies, there is a need for flexibility, constant update and openness. Partnering with the latest cutting-edge technology vendors and developers represents an important means to accomplish this objective. To collaborate or associate with the right partners is essential to develop better solutions and enlarge the IoT ecosystem with crucial support and promotion.

One important step towards IoT ecosystem creation in the INTER-IoT project was the launch of the INTER-IoT Open Call to incentivize external stakeholders to use INTER-IoT interoperability tools and create new technology solutions based on the INTER-IoT framework.

The Open Call has been an important means towards exploitation and validation of the Inter-IoT solutions: tools (INTER-Layer), framework (INTER-FW), methodology (INTER-METH) and developed technologies. Moreover, through the Open Call the INTER-IoT ecosystem widened by involving new partners and enhancing the offerings of the interoperability solutions. Indeed, the Open Call represented a key tool to impact and attract new stakeholders around an Inter-IoT interoperable global ecosystem, across many different application domains, while consolidating the stakeholders' relationships.

This call was open to individual European Small and Medium Enterprises (SMEs), Universities and Research Center Organizations (RTOs) that could contribute to the INTER-IoT paradigm.

This action allowed the evolution of the INTER-IoT products (i.e. INTER-Layer, INTER-FW and INTER-METH) to match the needs of proposers, but at the same time evolve their products in order to add new interoperability features. Moreover, this action allowed to test, validate and improve INTER-IoT components and their associated methodology, and add new scenarios, platforms and components on which

achieve interoperability. The proposals accepted validated the Inter-IoT solutions at different application domains.

The Inter-IoT Open Call was conducted following the basic principles that govern European Commission calls provided in the “Guidance note on financial support to third parties under H2020”: excellence of selected proposals, transparency, fairness and impartiality, confidentiality, efficiency and speed of evaluation.

The activities eligible to receive funding from the European Commission involved the integration of new components for a specific layer solution (i.e. device, network, middleware, application, data and semantics) that improve or extend the interoperability solution functionality and thereby INTER-FW, the integration of new IoT platforms in the INTER-IoT ecosystems and their application on the project pilots, and the enhancement of aspects of INTER-FW and INTER-METH. The detailed list of eligible activities for Open Call participants, as listed in the Grant Agreement between the European Commission and the INTER-IoT Consortium, are:

- Design, implementation and integration of interoperable device layer components for INTER-FW. Device layer components should be based on different low-level communication standards (e.g. Zigbee, 6LowPan, WIFI, Bluetooth, IEEE 802.15.4, NFC, etc.) or on ad hoc proprietary device solutions.
- Design, implementation and integration of interoperable networking layer components for INTER-FW. Networking layer components should be based on different standards higher-level communication standards (e.g. TCP/IP, HTTP, CoAP, etc.) or on ad hoc proprietary networking solutions.
- Design, implementation and integration of interoperable middleware layer components for INTER-FW. Middleware layer components need to deal with the different middleware services such as discovery, management, querying, coordination and interaction.
- Design, implementation and integration of interoperable application service components for INTER-FW. Application service layer components should exploit major standards e.g. HTTP, SOA, or REST as well as proprietary solutions.
- Design, implementation and integration of interoperable data and semantics layer components for INTER-FW. Specifically semantics layer components have to deal with heterogeneous IoT ontology matching.
- Design, implementation and integration of virtualization mechanism for smart objects and platform of smart objects for INTER-FW, including context-aware mechanisms and transfer of virtual objects between servers and cloud platforms.
- Design, implementation and integration of cloud support mechanisms to be integrated in INTER-FW, including support for different services, inter cloud mechanisms applied to IoT and support for virtualization.
- Design and implementation of new components of the INTER-METH tool for supporting integration among IoT platforms. Extension of the INTER-METH tool is needed to include the support for other IoT platforms to be made interoperable.
- Development of services/applications on top of the proposed use cases (INTER-Health and INTER-LogP) by (re)using the INTER-API to obtain a novel INTER-IoT ecosystem, representing the INTER-DOMAIN use case.

- Integration of IoT platforms in the three addressed use cases (INTER-Health, INTER-LogP, INTER-DOMAIN), following the INTER-METH methodology and the employing the INTER-IoT associated tools.

Next, it is presented the summary of the selected stakeholders that received funding from the European Commission and the different contributions that they proposed and provided. There were 2 large proposals, with a major budget and very ambitious objectives:

- **Integrating sensiNact platform with INTER-IoT-Framework**—CEA, Commissariat à l'énergie atomique et aux énergies alternatives (France/RTO)
- **INTER-OM2M**—VUB, Vrije Universiteit Brussel (Belgium, university).

As potential solutions that relied on a medium budget, 10 different small proposals were accepted:

- **INTER-HARE platform: Integration of multiband IoT technologies**—Universitat Pompeu Fabra (Spain/University)
- **Mission Critical operations based on IoT analytics (MiCrOBioTA)**—Nemergent Solutions S.R.L. (Spain/SME)
- **Interoperable Situation-Aware IoT-Based Early Warning System**—University of Twente (The Netherlands/University)
- **SENSHOOK**—Irideon S.L. (Spain/SME)
- **SOFOS: A software-defined end-to-end IoT gateway with virtualization**—INFOLYSIS P.C. (Greece/SME)
- **E3Tcity Smart City Platform and Devices Integration**—E3Tcity (Spain/SME)
- **ACHILLES: Access Control and authentication delegation for interoperable IoT**—AUEB—Athens University of Economics and Business Research Center (Greece/University)
- **INTER-HINC: Interoperability through Harmonizing IoT, Network Functions and Clouds**—TU Wien, Vienna University of Technology (Austria/University)
- **A Semantic Middleware for the information synchronization of the IoT devices**—ITIA-CNR, Institute of Industrial Technologies and Automation National Research Council (Italy/RTO)
- **SecurIoTy—security for the IoT**: AvailabilityPlus GmbH (Germany/SME).

As it can be appreciated in Fig. 2, the different proposals contributed to diverse aspects and areas of the INTER-IoT set of interoperability solutions. The majority of Open Call partners contributed with integrations on the different layer solutions of INTER-Layer.

Most Open Call contributions were related with the middleware layer (CEA, VUB, ITIA-CNR, e3tcity, TU Wien). This layer represents the level of IoT systems on which IoT information generated by Smart Objects flows across large streams of data in IoT platforms. CEA and VUB integrated IoT platforms with Inter-IoT by extending the Inter-MW solution with 2 new platform bridges. This way, both sensiNact and a smart city OneM2M platforms were able to seamlessly interoperate with other IoT

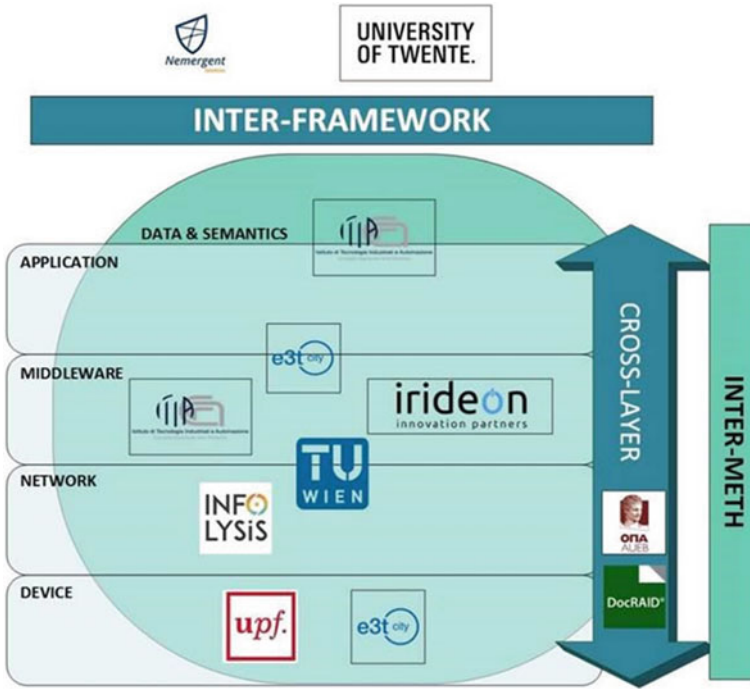


Fig. 2 Applications and contributions to different INTER-IOt interoperability solutions

platforms and systems connected to the Inter-MW solution, instead of being isolated platforms unable to communicate with others. The bridges will also enable the future connection of any instance of these type of platforms (sensiNact and OneM2M) to Inter-MW. In particular, the sensiNact bridge was reused in other H2020 project, ACTIVAGE, providing interoperability and numerous interoperability-related benefits to an Active and Healthy Ageing growing ecosystem.

E3tcity provided a Smart City platform with associated applications, which was integrated with the interoperability middleware and application solutions (Inter-MW and AS2AS). And differently, ITIA-CNR developed a Semantic Middleware (i.e. a semantic IoT platform) for the synchronization of IoT devices, which included the semantic dimension and involved the use of the Inter-IOt ontology and the Inter-MW solution.

Regarding network interoperability, the Open Call partner INFOLYSIS provided an important solution: a fully virtual smart gateway capable of supporting functions of network virtualization (SOFOS, a software-defined end-to-end IoT gateway with virtualization). Moreover, INFOLYSIS improved the INTER-IOt virtual gateway providing additional SDN and FNV functionality via the addition of new modules on top.

On the device layer, e3tcity aimed to integrate Smart Devices from a real Smart City using the Inter-IoT Smart Gateway and the e3tcity IoT platform. Also, UPF focused on the integration of multiband IoT technologies belonging to the device level in their INTER-HARE solution, which contributed to improve the Inter-IoT Smart Gateway functionality. Moreover, Irideon provided SENSHOOK, an IoT system that incorporates a smart solution for capturing mosquitoes, tested in the port of Valencia, one of the most important hubs in the world and a critical point for the entry of invasive species. This IoT system integrated the INTER-IoT Gateway as a middle element in their solution that allowed to collect data from the IoT Mosquito Trap system and deliver it to the SensHook monitoring IoT platform. A critical cross-layer feature for IoT systems is security, which must be provided across all INTER-Layer solutions, and to systems connected to INTER-Layer. Both AvailabilityPlus and AUES provided security mechanisms for IoT that enhanced the security provided through the use of INTER-Layer solutions.

Independently of Inter-IoT independent layer solutions, the University of Twente and Nemergent opted for feeding their full IoT systems with IoT data via the INTER-IoT solution. This data came from platforms connected to the INTER-IoT interoperability solution, and it was retrieved by Twente and Nemergent systems through the use of the INTER-IoT API (Inter-API) provided by INTER-FW. The University of Twente developed an Interoperable Early Warning System based on the monitoring of IoT sensing devices which provided situation awareness and enabled prompt warning in case of abnormal situations that require external attention. On the other hand, Nemergent provided a solution for Mission Critical Operations based on IoT analytics, called MiCrOBioTA.

Finally, TU Wien provided a solution that covered all INTER-IoT layers solutions in addition to enhance the INTER-FW, in order to provide Interoperability through Harmonizing IoT, Network Functions and Clouds (INTER-HINC). This solution allowed an easy launch and management of all INTER-Layer solutions, and it was integrated in INTER-FW, significantly improving the INTER-IoT framework functionality.

From an overall perspective, the Open Call projects a set fully exploit the functionality of the diverse Inter-IoT interoperability solutions and contribute in a well-balance manner to the whole areas of the Inter-IoT framework and the expansion of the ecosystem in different domains: Smart City, Port environment, AHA, Sanitary Plague Smart Prevention, e-Health, Critical Operation. Moreover, they contributed to the development of IoT cutting-edge technology such as the reception and interoperability of the latest multiband technologies or the creation of a novel semantic middleware. Furthermore, the availability of more and new data, thanks to the interoperability solutions, stimulates the creation of new opportunities, services and products.

In the following subsections, each of the Open Call projects contribution for the IoT ecosystem, along with their integration with INTER-IoT, is described.

3.1 *E3Tcity Smart City Platform and Devices Integration*

E3Tcity platform³ is a Smart City platform in production stage used in more than 20 towns in Spain, which provides a wide variety of services spanning from public lighting control to mobility control solutions such as traffic, parking, crowd, traffic lights, irrigation and water quality, and heating, ventilation and air conditioning control. All those services are remotely controlled and monitored.

ET3city platform was integrated with INTER-IoT at middleware level, achieving interoperability with other entities connected and providing INTER-IoT a whole device-cloud-app solution that could be applied in IoT scenarios, such as INTER-LogP port pilot. For this integration, it was necessary to develop an interoperability bridge for E3Tcity platforms that connects to Inter-MW.

In order to validate the integration of et3city with INTER-IoT, and to take advantage of the use of the platform resources in an INTER-IoT pilot, it was performed a technical feasibility assessment of the integration of E3TCITY devices the INTER-IoT architecture, via the use ModBus and MQTT standards, in the port of Valencia. The final result was the integration of the et3city solution in the lighting pilot of INTER-LogP, on which the platform interacted and interoperated seamlessly with several port management IoT platforms to create a smart lighting system, thanks to the interoperability provided by Inter-MW.

3.2 *SENSHOOK*

SENSHOOK is a smart solution provided for Plague Prevention and Disease Control via IoT technologies. Several mosquito species can bring lethal diseases to humans, and its entrance in the country must be controlled and monitored for the sake of citizen safety. The port of Valencia is one of the most important hubs in the world and thus a critical point of entry of invasive species that must be monitored, according to the European Centre of Disease Control. For these reasons, it represents a key neuralgic center for applying the SENSHOOK solution.

Non-native species cost the EU €12 billion per year in damage and control costs. In the last decades several species of disease carrying mosquitoes have invaded Europe through the transport of goods, international travel and climate change. SensHook potentially reduces inspection costs and improves surveillance programs.

The system was deployed as a set of observation static IoT nodes in a critical point of the port of Valencia. These nodes were composed of a Smart Mosquito Trap capable of mimicking human body signals (scent and respiration) and of automatically counting captured mosquitoes, identify the gender and the species. The information collected by each node was sent to a central server.

³ <https://partners.telefonica.com/es/buscador-partners/e3tcity-sl-8b6e9021-6344-e711-8166-3863bb34ed80>.

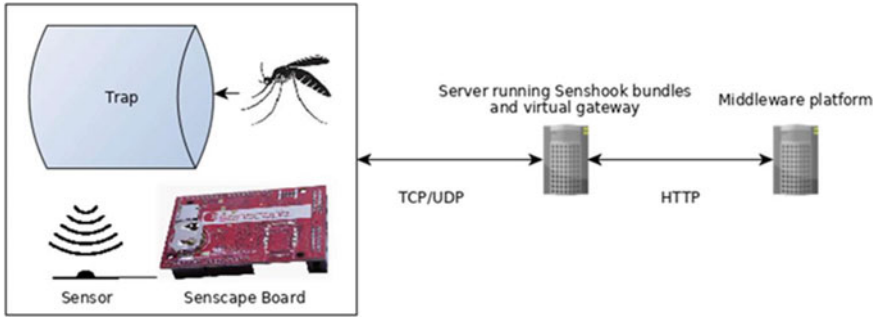


Fig. 3 SENSHOOK system overview

The following diagram provides a high-level overview of the system. Every time a mosquito enters in the trap it becomes detected by the sensor. This sensor is connected to a Senscape board which sends the information to a server running SensHook and to the INTER-IoT virtual gateway. The middleware platform can then retrieve the gathered information, sent by the INTER-IoT Virtual Gateway. The communication between the trap and the monitoring IoT platform is bidirectional (Fig. 3).

The pilot started with preliminary tests in the zoo of Barcelona in May-June 2018 and lasted until October 2018, covering exactly the period of the year when disease-vector mosquitoes are active and must be critically monitored.

Irideon S.L.⁴ contributed to the INTER-IoT project by providing a new open tool for the INTER-LAYER building block (a virtual gateway connector for enabling the connection with the trap). This contribution will allow the evolution of products based on INTER-IoT, and at the same time will allow Irideon S.L. to evolve SENSHOOK products in order to add new interoperability features.

3.3 *MiCrOBioTA: Mission Critical Operations Based on IoT Analytics*

The “Mission Critical operations based on IoT analytics” (MiCrOBioTa) project, developed by Nemergen⁵, aims at exploiting the INTER-IoT platform as a means to gather information from heterogeneous sensors of diverse IoT platforms in a converged way. This way, the Nemergen Control Room engine receives IoT sensing information from a monitored environment, from IoT platforms connected to INTER-IoT (Inter-MW). This information will be used to perform Mission Critical Operation on those environments.

⁴ <http://irideon.eu>.

⁵ <https://nemergent-solutions.com/lang/es/>.

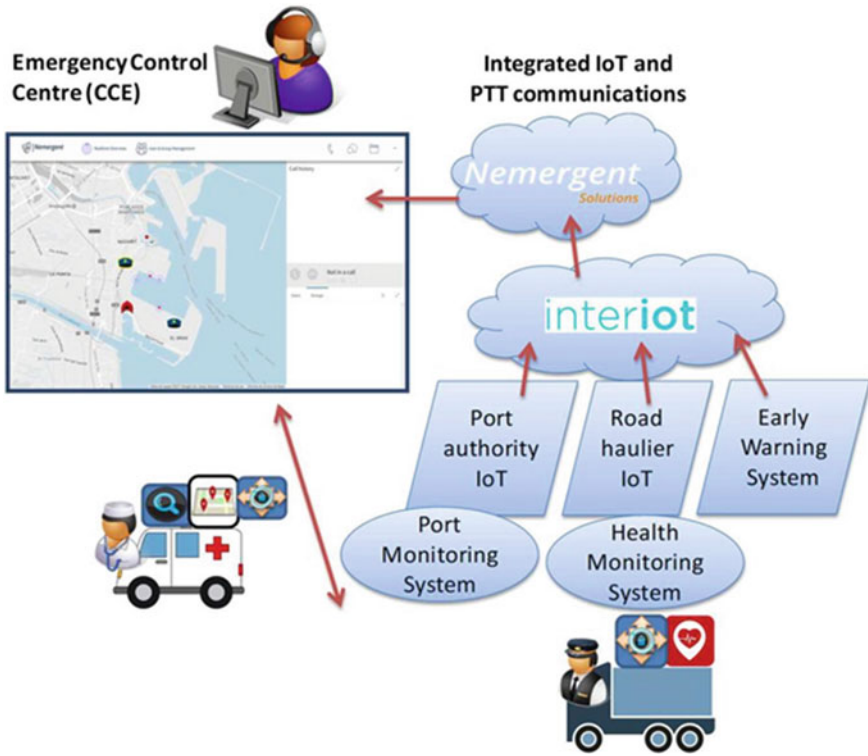


Fig. 4 MiCrOBioTa cross-domain use case employing interoperability solutions

As a result, Nemergent integrated a new IoT monitoring and analytics component in its product portfolio, and especially into the Nemergent Control Room application. Figure 4 illustrates the overall Nemergent mission critical applications framework and the specific scope of the work in MiCrOBioTa.

The benefits of the interoperable “Mission Critical operations based on IoT analytics” are unquestionable. A typical situation in mission critical operations support systems is to include information coming from specifically deployed devices to gather environmental measurements. Examples of these devices are temperature sensors, meteorological and hydrological probes, traffic monitoring cameras, etc. The Mission Critical IoT (MC-IoT) system, which includes a new monitoring and analytics component and an evolved Control Room interface tailored to the specific needs of the use case. In the case of a simulated crisis, significant information from on-body health-related sensors and port logistics devices will provide life-saving information to the mission critical operations support system. Besides, the available mission critical communications components can be used to demonstrate the crisis handling use case.

MiCrOBIoTa defined a cross-domain IoT scenario in the scope of mission critical operations. Moreover, it contribute to INTER-IoT activities related to the definition of the common semantics for specific mission critical scenarios. Also, for the integration with INTER-IoT, NEMERGENT developed a new component to cover the communication with the INTER-IoT platform. This component, namely “Nemergent MC-IoT monitoring and analytics”, interfaces with INTER-IoT API and with the Nemergent Control Room application. This connector implements all the necessary calls to the INTER-IoT API in order to gather the selected information with the corresponding message formats and data types.

The Nemergent Control Room application was adapted to perform basic IoT-related operations. Activities were monitored and controlled through the use of the evolved Nemergent Control Room. A service was created to perform specific actions in response to each of the messages. For example, temperature variation can be monitored, raising an alarm when it exceeds a predetermined threshold.

3.4 Interoperable Situation-Aware-IoT-Based Early Warning System

The University of Twente⁶ provided an Early Warning System (EWS) on top of an IoT platform (FIWARE⁷), which interoperates with other EWSs, emergency systems and emergency services [10]. The system coordinates emergency services based on IoT smart monitoring, alerting the involved parties (e.g. emergency command control, first responders and employees) when an accident occurs. The set of emergency-related services and the alert management are on top of different IoT platforms, and are coordinated through the IoT-based EWS, by enabling data exchange among them. Those platforms are very heterogeneous in terms of semantics and information models, and are not directly interoperable. To enable the sharing of information among them, IoT ontology translations are performed through the IPSM from INTER-Layer. This way, semantic interoperability is enabled across the different heterogeneous IoT systems, adapting the semantic model of the data to be understandable for the receiver. Several semantic models for disaster management were employed, which combined e-Health standards, context awareness and logistics. Notably, it was enabled of interoperability between the most important ontologies for IoT, the Smart Appliances REFERENCE ontology (SAREF⁸) and W3C Semantic Sensor Network Ontology (SSN⁹).

This EWS was implemented and validated in a cross-domain use case that combined the e-health, port logistics and transportation domains, allowing early warning in ships and boats in case of health emergency [11]. In the port of Valencia, on which

⁶ <https://www.utwente.nl/en/>.

⁷ <https://www.fiware.org>.

⁸ <https://saref.etsi.org>.

⁹ <https://www.w3.org/TR/vocab-ssn/>.

it was implemented this solution, transportation companies (haulers) and insurance companies could benefit from the IoT EWS by reducing disaster risks involving the employees and the goods being transported. This EWS project had significant impact on the IoT solutions industry, as it provides innovation by enabling the data exchange of different data formats from heterogeneous IoT platforms to detect and alert emergencies. Also, it represents a low-cost business model for logistics, healthcare and insurance companies.

Main technical achievements for the enablement of this novel EWS are the following:

- Semantic integration of a variety of data sources, avoiding a loss of semantics when multiple ontologies, standards and data models from different and overlapping domains are involved. considering their syntactic and semantic alignments.
- Processing in time- and safety-critical applications: provide the required performance for upstream data acquisition, emergency risk detection and message brokering, in terms of scalability and total transaction time.
- Data analysis for effective response: enable high quality situation awareness to avoid false positives and improve decision support based on emergency procedures.

3.5 INTER-HINC: Interoperability Through Harmonizing IoT, Network Functions and Clouds

INTER-HINC provided a framework for building “IoT Cloud systems” that link and use different IoT platforms. A main objective of the framework was to allow complete interoperability and integration between those platforms in the Cloud system and focusing on data integration and protocol integration via plug-ins and APIs (that includes INTER-API and Inter-MW). INTER-HINC relies on resources (offered by IoT, network functions and cloud providers) and software artefacts that can be instantiated [12]. This way, it makes use of network functions, cloud functions, edge computing brokers, edge computing workflows engines and domain and application specific services, as well as virtualized INTER-IoT solutions.

For creating this solution, TU Wien¹⁰ made use of INTER-IoT interoperability tools and INTER-API, creating a framework that allowed an instantaneous deployment of the layer solutions, regardless the underlying operating systems and technologies, through advanced virtualization techniques. By means of the INTER-IoT virtualized solutions, it was possible the linking of several platforms on the Cloud, providing interoperability across them at all levels (including data and protocol). This INTER-HINC framework was seamlessly integrated in INTER-FW, notably enhancing the INTER-IoT framework functionality [13].

¹⁰ <https://www.tuwien.at>.

3.6 A Semantic Interoperable Middleware for the Information Synchronization of the IoT Devices

ITIA-CNR developed one of the most ambitious Open Call IoT solutions: the creation of an interoperable semantic middleware for IoT (i.e. semantic IoT platform). From a semantic perspective, this middleware adopted the Global IoT Ontology (i.e. GOIoTP)¹¹ from INTER-IoT, and ensured by design that all exchanged information, including synchronization requests, is expressed under a semantic model. Very few IoT platforms adopt a complete semantic model for IoT information management, being this a very novel and interesting feature.

ITIA-CNR successfully developed a publish-subscribe middleware that combines Semantic-Web and agent-based technologies to provide services for events notification, and designed a mechanism with which devices can subscribe to the Semantic Middleware [14] to receive alerts related to the changes of the state of one or more elements of interest.

The integration of ITIA-CNR Semantic Middleware with INTER-IoT [15] was performed at two different levels. First, as a middleware or IoT platform it was integrated with Inter-MW (INTER-IoT interoperability solution at middleware level) through the creation of a specific Inter-MW bridge that connected both artifacts, and could be reused on any instance of this new middleware. This way it could interoperate seamlessly with other IoT platforms, receiving information from them in this new middleware's format, semantics and information model. From the other hand, from a semantic perspective, it followed the INTER-IoT ontology (i.e. GOIoTP), thus directly integrating with the INTER-IoT semantic model (Fig. 5).

3.7 INTER-HARE Platform: Integration of Multiband IoT Technologies

Inter-HARE [16] is focused on the integration of new multiband IoT technologies. This area is clearly related with the technologies employed in the IoT device layer, and the interoperability of Smart Gateways. Their solution integrated with the Inter-IoT Smart Gateway, the interoperability solution provided by Inter-IoT for the device layer that enables the communication of disjoint IoT devices with the IoT platform, employing very heterogeneous communication technologies and network protocols.

One of these network technologies is LPWAN (low-power wide-area network), a type of wireless telecommunication wide area network designed to allow long-range communications at a low bit rate among IoT Smart Devices (connected objects), such as sensors. The low power, low bit rate and intended use distinguish this type of network from a wireless WAN (wide area network) that is designed to connect users or businesses, and carry more data, using more power. LPWAN is a recent

¹¹ <https://inter-iot.github.io/ontology/>.

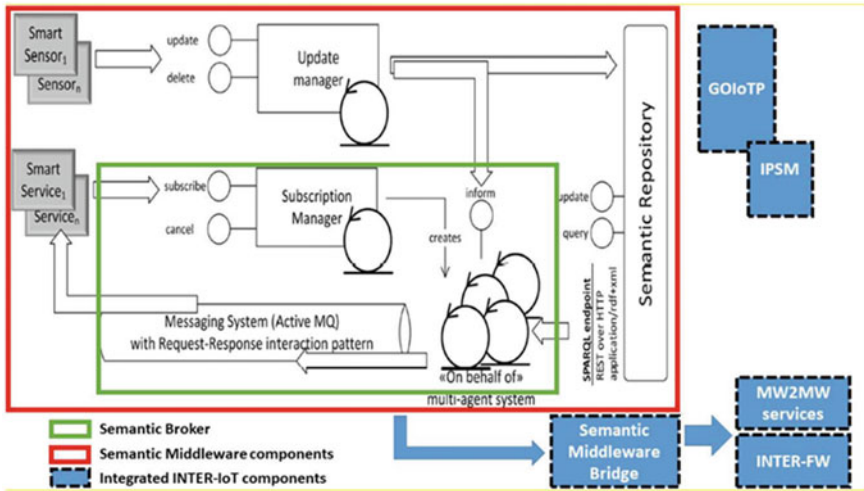


Fig. 5 Overall architecture of Semantic Middleware and its interaction with INTER-IoT solutions

technology IoT-oriented, as solves the connectivity problem of most IoT devices that typically have very limited energy and thus very reduced wireless distance reach. A LPWAN may be used to create a service or infrastructure allowing the owners of sensors to deploy them in the field without investing in gateway technology.

The INTER-HARE project is intended to design a new LPWAN technology flexible enough to transparently encompass both LPWAN devices and multiple so-called low-power local area networks (LPLANs) while ensuring overall system’s reliability. A cluster-tree network is created, where the LPWAN acts not only as data collector, but also as backhaul network for several LPLANs, as shown in Fig. 6, in which is possible to see the role of the Inter-IoT Gateway in the INTER-HARE set-up, managing their corresponding LPLAN, to which sensors connect, in a hierarchic way.

Communication within the LPWAN is based on the HARE protocol stack [17], ensuring transmission reliability, low energy consumption by adopting uplink multi-hop communication, self-organization, and resilience. Under these premises, LPWAN boundaries are extended beyond typical 868 MHz coverage range and easily integrate devices coming from adjacent/overlapping 2.4 GHz LPLANs. The use of separated frequency bands in overlapping networks results in an overall reduction of interferences. Lastly, thanks to the hierarchic system proposed, scalability is enforced by a management based on sub-networking techniques.

The INTER-HARE platform is conceived as an innovative evolution of HARE protocol stack and can be considered as a dynamic multiprotocol. INTER-HARE successfully guaranteed a transparent network interoperability and a unified, centralized, self-organized control of heterogeneous devices. Moreover, this project designed, developed and tested a novel end-to-end communication protocol, both from the LPWAN and from the LPLANs. For this aim, it was built a simulation

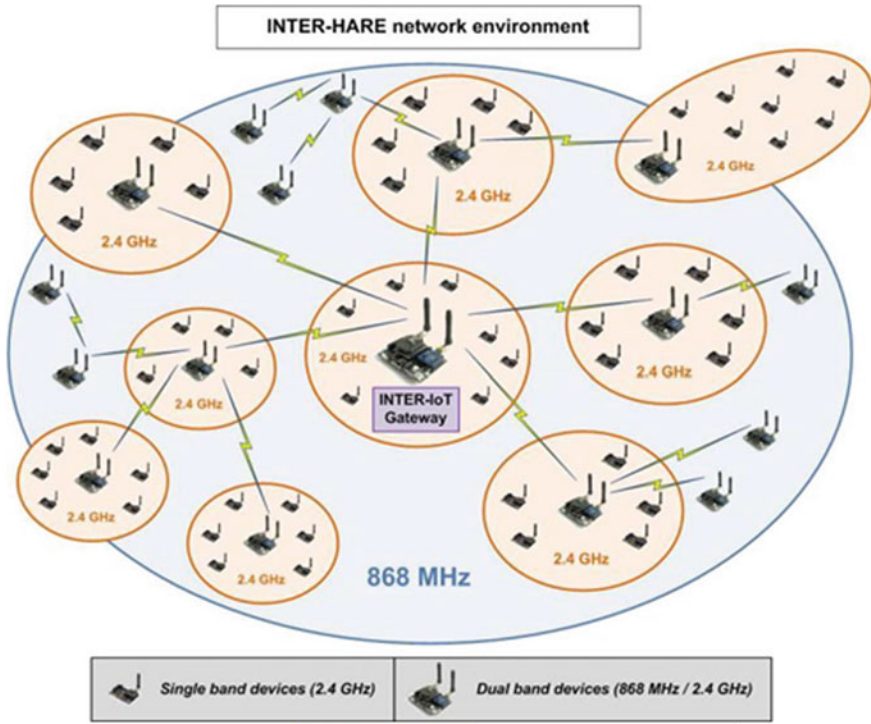


Fig. 6 INTER-HARE network environment

platform based on open-source technologies for preliminary testing of communication mechanisms, a preliminary testbed with heterogeneous devices in laboratory, and finally it was performed a pilot execution of the whole platform in a real IoT environment and use case. Statistics were collected by the Inter-IoT virtual gateway regarding network’s resilience and self-organization as well as communication performance, proving the success of this technology environment.

3.8 *SOFOS: A Software-Defined End-to-End Gateway with Virtualization*

The disruptive and accelerated arrival of the IoT implies that future networks need a new architecture to accommodate end-to-end IoT networking, dealing with: (i) the exponential massive increase in data generation, (ii) the problems related to the end-to-end IP networking of the resource-constrained IoT devices, (iii) the capacity mismatch between devices, and (iv) the rapid interaction between services and infrastructure.

Software defined networking (SDN) and network function virtualization (NFV) are two technologies that promise to cost-effectively provide the scale and versatility necessary for IoT services in order to address efficiently the aforementioned challenges. Moreover, given that SDN and NFV are considered fundamental components in the 5G landscape, since it is widely recognized that 5G networks will be software-driven and most components of future heterogeneous 5G architectures should be capable to support software-network technologies, both SDN and NFV are promising candidate technologies for a Software Defined Approach of end-to-end IoT Networking [18].

SOFOS [19] complemented the existing INTER-IoT framework with SDN and NFV functionalities towards a Software-defined end-to-end IoT infrastructure with IoT service chaining support. The main objective of SOFOS SDN/NFV-enabled framework is to enhance the interoperability of the INTER-IoT framework in order to facilitate the interoperable management of a large number of diverse smart objects that currently operate utilizing a variety of different IoT protocols (CoAP, MQTT, HTTP, etc.).

SOFOS added SDN/NFV Automation and Verification in IoT Infrastructures to relocate various IoT functions from HW appliances to Virtual Machines (VMs) (i.e. Virtual Network Functions—VNFs), to enhance the interoperability support of the INTER-IoT platform by deploying VNFs that map IoT protocols (such as CoAP, MQTT) to standard IP networking, to connect and chain the software-defined IoT functions (i.e. VNFs) together and to abstract plane by exploiting the SDN concept.

INTER-IoT infrastructure with the proposed advances can be enhanced by means of NFV with integration of SDN, making it more agile and introducing a high degree of automation in service delivery and operation—from dynamic IoT service parameter exposure and negotiation to resource allocation, service fulfilment, and assurance. SOFOS has deployed on top of INTER-IoT virtual gateway modules that provide SDN/NFV Automation in IoT Infrastructure, such as INFOLYSiS SDN/NFV Network Manager. By applying appropriate OPENFLOW commands, INFOLYSiS add-on steer the data traffic from the INTER-IoT Virtual Gateway to the various VNFs deployed, which is translated into a common protocol (e.g. HTTP). This way, the interoperability functions of INTER-IoT are enhanced, and the application layer becomes able to represent the received data in a unified way.

3.9 ACHILLES: Access Control and Authentication Delegation for Interoperable IoT

The ACHILLES solution [20] addressed an important security challenge in IoT, providing cross-layer security which can be applied to an IoT network (i.e. smart devices connected to a smart gateway), and it was integrated within the INTER-IoT virtual gateway.

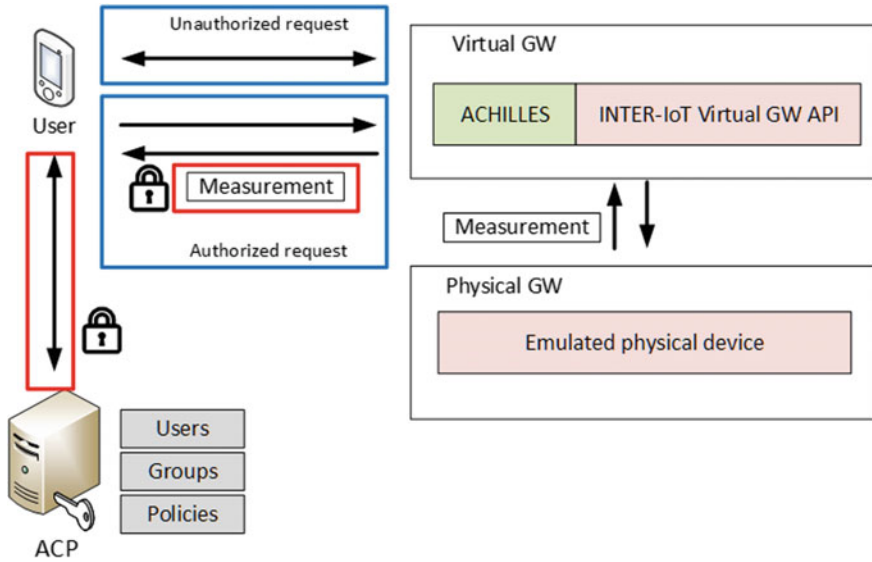


Fig. 7 ACHILLES security scheme for IoT systems

ACHILLES is focused on IoT access control and endpoint authentication, which is a current important challenge in IoT. Things are usually small devices with limited storage capacity, power, energy, and processing capabilities, in order to be inexpensive and practical. In many cases IoT devices are exposed to tampering, whereas in many application scenarios on which those devices are deployed, it is not easy to access them. IoT devices, such as sensors and actuators, usually are not able to perform heavy tasks, such as complex cryptographic operations. Storing user credentials or any other sensitive information in a Thing or Smart Object creates security risks, adds storage overhead, and makes security management an impossible task. When it comes to interoperable applications, Things (or even gateways) cannot interpret complex business roles and processes. Moreover, companies are not willing to share sensitive information about their users with a Thing (or a gateway), even if this information is required by an access control mechanism. Also, neither they want to invest in another security system.

The ACHILLES project overcomes these limitations by allowing the delegation of security operations to a third party, referred to as the Access Control Provider (ACP), which can be implemented by a trusted separate entity, or even the service provider itself. The ACHILLES concept is depicted in Fig. 7.

This solution is fully integrated with Inter-IoT as the virtual part of the Inter-IoT gateway has been extended to include ACHILLES API (as it can be seen in the figure below). The Achilles API is integrated as a part of the Virtual Gateway API.

This extension implements ACHILLES functionality and configuration files, and it is able to perform read and write calls to physical devices provided by the IoT

framework. It is possible to externally communicate as a client to the INTER-IoT gateway, through this API, and interact with an ACP, enhancing the security of IoT systems.

3.10 SecurityIoT: Security for the IoT

SecurityIoT is focused on providing higher security to IoT systems, through an advanced security mechanism that can be incorporated in the cross-layer of INTER-layer solution. In a complementary way to ACHILLES, which was focused on access control at device level using an authentication server, Security IoT is focused on a secure transmission of IoT information by means of an advanced encryption system. This type of transmission is related with the cloud, platform and application levels, and thus it is integrated as a cross-layer feature with the inter-MW and AS2AS INTER-IoT solutions. Concerns about information security are one of the main reasons for companies and private individuals not to adopt cloud services and to be skeptical about IoT systems. On top of concerns regarding physical- and cyber-attacks, international corporations additionally carry legal attacks in their threat model. However, cloud services are essential in improving efficiency and cost structures. SecurIoT addresses that gap by combining several security mechanisms to protect data and to address all relevant security dimensions such as confidentiality, integrity and availability. This solution uses DOCRAID¹² and CloudRAID¹³ fragmentation and encryption [21]. By these means, data is fragmented, the fragments are encrypted and they are redundantly distributed to multiple independent storages [21]. As SecurIoT is storage agnostic, meaning that the data fragments may be distributed across multiple jurisdictions, adding additional security. This way, SecurIoT solves security and compliance issues when sending and sharing data via public networks like the Internet and when storing data in cloud services, thus enabling companies to use cloud based IoT services, which they would not use without this type of protection. This security solution is a crypto proxy technology and as such it is transparent to users and does not affect user experience or end processes. The integration SecurIoT in platforms and applications is seamless and transparent, thus non-invasive, and this security mechanism can be operated off-premise, on-premise or in a hybrid manner. Regarding its integration with INTER-IoT, SecurityIoT was linked in a pilot with AS2AS and INTERMW components. AvailabilityPlus¹⁴ provided a web application that facilitates the access and management to application and middleware solutions in INTER-IoT, providing exchanged information making use of this underlying data encryption and fragmentation.

¹² <https://www.docraid.com/>.

¹³ <https://hpi.de/meinel/security-tech/secure-cloud/secure-cloud-storage.html>.

¹⁴ <http://www.availabilityplus.com/>.

3.11 *Integration of the SensiNact Platform with the INTER-IoT Framework*

sensiNact¹⁵ [22] is an open-source IoT platform based in the evolution of BUTLER,¹⁶ an IoT platform created by the H2020 BUTLER project with the endorsement of the ECLIPSE foundation.¹⁷ sensiNact is an edge IoT platform, that can be directly installed in gateways and performs network routing functions. The relevance and impact in the IoT landscape of sensiNact is significant, since it has been employed in numerous research project in Europe and Asia (i.e. Korea and Japan) on very diverse application domains from Smart Cities to Active and Healthy Aging.

CEA¹⁸ performed the integration of the sensiNact open IoT platform with the Inter-IoT framework at middleware level (i.e. platform level). This way, it was enabled seamless interoperability among sensiNact and any other platforms connected to INTER-IoT middleware solution. This integration required the creation of an interoperability bridge to connect any instance of sensiNact with INTER-MW. Moreover, it was also defined an IPSM semantic alignment to allow semantic interoperability across its connection with Inter-MW and other platforms, and the ability to publish information following the INTER-IoT ontology (GoIoT¹⁹) [2]. This integration was tested and validated with a large sensiNact deployment of traffic sensors in Japan.

The collaboration of CEA in INTER-IoT brought a relevant case of ecosystem building as the bridge was further reused in the Large Scale Pilot H2020 ACTIVAGE project, validating the extendibility and reuse of technology in different application domains. The bridge allowed to connect the sensiNact platforms of the Isère Smart Home large-scale pilot to an Active and Healthy Ageing (AHA)²⁰ [] ecosystem that employs the INTER-IoT solution for connecting platforms and enable semantic interoperability across them. The bridge allowed the interoperability of sensiNact with many other platforms connected in the ecosystem (e.g. universAAL,²¹ FIWARE,²² SOFIA2²³).

Moreover, from an ecosystem building perspective, it allowed to join forces between INTER-IoT and the sensiNact ecosystem for sustainable exploitation plans, foresee this exploitation within the ACTIVAGE project and new projects in perspective, and moreover collaborate with the ECLIPSE and the Urban Technology

¹⁵ <https://projects.eclipse.org/projects/technology.sensinact>.

¹⁶ <https://cordis.europa.eu/project/id/287901>.

¹⁷ <https://www.eclipse.org/org/foundation/>.

¹⁸ <https://www.cea.fr>.

¹⁹ <https://inter-iot.github.io/ontology/>.

²⁰ <https://www.who.int/westernpacific/news/q-a-detail/ageing-healthy-ageing-and-functional-ability#:~:text=Healthy>.

²¹ <https://www.universaal.info>.

²² www.fiware.org.

²³ <https://sofia2.readthedocs.io/en/latest/>.

Alliance.²⁴ This interoperability solution supported the development of a wide range of use cases and allowed application developers to produce new added value across multiple systems.

3.12 INTER-OM2M

OneM2M [2] is an ETSI standard that can be used as a reference architecture for IoT platforms [23], while OM2M²⁵ or Open Machine to Machine is an Eclipse open source project that developed an implementation of the oneM2M standard, providing the OM2M IoT platform.²⁶

Vrije Universiteit Brussel²⁷ (VUB) has a deployment for SmartCity managed by an open source OM2M platform with different sensors connected to provide vehicle location data and environmental information. This platform was integrated in the INTER-IoT solution via the development of an OM2M interoperability bridge and an IPSM semantic alignment that allows the seamless connection of this type of OM2M platform with other IoT platforms and systems, providing interoperability and thus common understanding of the information across them [24].

From an ecosystem perspective, VUB enlarged the INTER-IoT ecosystem and INTER-Layer solutions with the interoperability bridge specific to OneM2M standards, and endorsed the developments in a broader scenario. Moreover, VUB extended the INTER-IoT framework with respect to the application protocols HTTP, COAP and MQTT that are employed in VUB SmartCity OM2M[24] platform.

4 INTER-IoT Adoption in H2020 Projects

The INTER-IoT interoperability solutions were also successfully adopted in other H2020 European projects, such as PIXEL,²⁸ ACTIVAGE²⁹ and 5GENESIS.³⁰

H2020 PIXEL (Port IoT for Environment Leverage) [25] foresees the use of the Inter-MW solution for connecting platforms in port environments, enabling interoperability across them.

On the other hand, the H2020 LSP ACTIVAGE project [6] adopted INTER-IoT solutions for creating an Active and Healthy Ageing (AHA) interoperable ecosystem across multiple IoT platforms of Smart Homes deployment sites across Europe [26].

²⁴ <https://www.urbantechnologyalliance.org>.

²⁵ <https://www.eclipse.org/om2m/>.

²⁶ <https://wiki.eclipse.org/OM2M/Download>.

²⁷ <https://www.vub.be>.

²⁸ <https://pixel-ports.eu>.

²⁹ <https://www.activageproject.eu>.

³⁰ <https://5genesis.eu>.

The major aim of the ACTIVAGE project is the foundation of the first European AHA ecosystem, in order to address and provide IoT-based solutions to the critical societal problem of the steep increase of elderly population around the world, along with the caring needs that they need. This way, this AHA ecosystem provides smart solutions for improving the well-being and home safety of elders, as well as promoting an active and healthy lifestyle.

The combined use of Inter-MW and the IPSM in ACTIVAGE provided semantic interoperability across the whole AHA ecosystem [26]. This ecosystem it is composed by 12 AHA deployment sites from 12 different regions across Europe and several IoT systems from third parties. Each deployment site or system is managed by its own IoT platform. The interconnection and interoperability of those deployment sites and external systems was achieved through platform interoperability bridges connected to Inter-MW and the use of specific semantic alignments for the IPSM. As a result, new IoT platforms were included both in the INTER-IoT and ACTIVAGE ecosystem: IoTivity,³¹ SOFIA2,³² OpenIoT,³³ MC Cardio,³⁴ eukari;³⁵ and eukendu.³⁶ Moreover, the Inter-IoT ontology (GOIoT) became the core of the ACTIVAGE ontology. The integration actions on these AHA platforms validated the scalability of INTER-Layer and enlarged the INTER-IoT interoperability solution.

Focused on providing 5G solutions, the H2020 5GENESIS³⁷ project adopted the INTER-IoT smart gateway, capable of providing NFV/SDN functionality, in one of their pilots across Europe (i.e. Limassol pilot) [27].

Furthermore, alliances with stakeholders incentivize the adoption of INTER-IoT in future projects and endeavours. For example, the collaboration between INTER-IoT and sensiNact foresees a future participation in joint exploitation activities and in other projects within the sensiNact ecosystem. Moreover, it would promote collaborations with the Eclipse Foundation.³⁸

5 IoT-EPI

IoT-EPI³⁹ is a joint project initiative endorsed by the European Commission that aims to promote a vibrant and sustainable IoT ecosystem in Europe [28]. All Hori-

³¹ <https://iotivity.org>.

³² <https://sofia2.readthedocs.io/en/latest/>.

³³ <http://www.openiot.eu>.

³⁴ <http://EHR.it>.

³⁵ <https://www.activageproject.eu>.

³⁶ <https://www.activageproject.eu>.

³⁷ <https://5genesis.eu>.

³⁸ <https://www.eclipse.org>.

³⁹ <https://iot-epi.eu>.

Fig. 8 IoT-EPI initiative



zon 2020 research projects of the ICT30 cluster⁴⁰ participated in the IoT-EPI initiative (i.e. INTER-IoT,⁴¹ BigIoT,⁴² symbIoTe,⁴³ bIoTtope,⁴⁴ VICINITY,⁴⁵ Agile⁴⁶ and Tag-ItSmart⁴⁷) [29]. These RIA⁴⁸ projects under different approaches and scopes seek the common goal of improving horizontal interoperability and provide viable solutions for IoT platforms and connected devices to easily, safely, and reliably be integrated for a multiplicity of IoT applications. Each RIA project focuses on different nuances of interoperability or standardisation with respect to IoT platform solutions, gateways for IoT devices, and asset monitoring [29].

The initiative IoT-EPI developed a series of activities meant to foster and support the 7 RIA projects to build different IoT platform ecosystems. The IoT-EPI provided a common space to support the projects to find assistance to solve common problems and to identify and deal with shared threats and common opportunities. Partners of these European projects provided information and recommendations about their first-hand experience through questionnaires, roundtables, and workshops. Each project pursues a different goal, adopts different platforms, and has a different experience on establishing ecosystems across different domains, while shares a common overall objective in broad terms. Due to this fact the sharing of their experience to other RIA projects was highly valuable (Fig. 8).

Moreover, the IoT-EPI enabled the creation of a community for the seven RIA projects, and it was considered as an aggregation point thanks to the workshops, the

⁴⁰ <https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/ict-30-2017>.

⁴¹ <https://iot-epi.eu/project/inter-iot/>.

⁴² <https://iot-epi.eu/project/big-iot/>.

⁴³ <https://iot-epi.eu/project/symbiote/>.

⁴⁴ <https://iot-epi.eu/project/biotope/>.

⁴⁵ <https://iot-epi.eu/project/vicinity/>.

⁴⁶ <https://iot-epi.eu/project/agile/>.

⁴⁷ <https://iot-epi.eu/project/tagitsmart/>.

⁴⁸ <https://ec.europa.eu/info/funding-tenders/opportunities/portal/screen/opportunities/topic-details/ict-30-2017>.

roundtables, the meetups, and all the occasions for the different projects to have a communication shared space.

6 Conclusions

INTER-IoT has been part of a larger European initiative called IoT-EPI in which the key research leaders in IoT have provided ways of overcoming different interoperability problems between heterogeneous IoT systems.

The INTER-IoT initiative has provided a novel set of interoperability tools that enable interoperability at every layer of IoT systems, along with an associated framework and an implantation methodology. Those can facilitate the reuse and integration of existing and future IoT systems to obtain interoperable ecosystems of IoT platforms. The development of INTER-IoT has allowed stakeholders and developers to interact with different IoT platforms in a domain agnostic ecosystem. The creation of such ecosystem has addressed the needs of use cases and scenarios in which different IoT platforms are involved, and mainly in those in which more than one application domain is addressed.

Key aspects of INTER-IoT solutions that remarkably promote ecosystem building are the novel ability of providing seamless interoperability across any IoT systems, and the openness of the framework. Notably INTER-IoT provides semantic interoperability across heterogeneous IoT platforms that do not share common information models, data formats and semantics and employ different and heterogeneous communication interfaces.

Open interoperability delivers on the promise of open source software, enabling vendors and developers to interact and interoperate, without hindering with anyone's ability to compete by delivering a superior product and experience. INTER-IoT has been designed not to compete, but to contribute and collaborate with current growing ecosystems such as FIWARE, UniversAAL or Industrial Data Spaces.

In the absence of global IoT standards, the INTER-IoT interoperability solutions support and make it easy for any company to design IoT devices, smart objects or services and get them to market quickly to a wider client-base and to create new IoT interoperable ecosystems. In the long term, ability for multiple applications to connect to and interact with heterogeneous sensors, actuators, and controllers, thus making them interoperable, will become a huge enabler for new products and services.

INTER-IoT has created growing ecosystems with different entities making use of the different results provided by the project, mainly in the AHA, smart cities and transportation and logistics application domains. Other domains explored have been e-Health, Plague Control, Disease Prevention, Emergency Management and Industry. As the ecosystem of interoperable devices and services expands, so it will increase the value of building new devices for and applications working within this ecosystem. The data spaces ecosystem will provide a mean to exploit the interoperability tools developed in INTER-IoT and application market places will benefit from the results.

This emerging ecosystem is not owned by any business or entity, but rather it exists to enable many entities to pool their resources together to create larger opportunities for all.

References

1. Fortino, G., Savaglio, C., Spezzano, G., Zhou, M.C.: Internet of things as system of systems: a review of methodologies, frameworks, platforms, and tools. *IEEE Trans. Syst. Man Cybern. Syst.* **51**(1), 223–236 (2021)
2. Kim, J., Choi, S.-C., Yun, J., Lee, J.-W.: Towards the oneM2M standards for building IoT ecosystem: analysis, implementation and lessons. *Peer-to-Peer Netw. Appl.* **11**(1), 139–151 (2018)
3. Fortino, G., Savaglio, C., Palau, C.E., de Puga, J.S., Ghanza, M., Paprzycki, M., Montesinos, M., Liotta, A., Llop, M.: Towards multi-layer interoperability of heterogeneous IoT platforms: the INTER-IoT approach. *Internet of Things* 199–232 (2018)
4. Fortino, G., Palau, C.E., Guerrieri, A., Cuppens, N., Cuppens, F., Chaouchi, H., Gabillon, A. (eds.): *Interoperability, Safety and Security in IoT—Third International Conference, InterIoT 2017, and Fourth International Conference, SaSeIoT 2017, Valencia, Spain, November 6–7, 2017, Proceedings. Lecture Notes of the Institute for Computer Sciences, Social Informatics and Telecommunications Engineering*, vol. 242. Springer (2018)
5. Gonzalez-Usach, R., Yacchirema, D., Julian, M., Palau, C.E.: *Interoperability in IoT*. IGI, 2018. *Handbook of Research on Big Data and the IoT*
6. González-Usach, R., Palau, C., Julian, M., Belsa, A., Llorente, M. A., Montesinos, M., Ganzha, M., Wasielewska, K., Sala, P.: Use cases, applications and implementation aspects for IoT interoperability. In: *Next Generation Internet of Things: Distributed Intelligence at the Edge and Human Machine-to-Machine Cooperation*. River Publishers (2018)
7. Yacchirema, D., Gonzalez-Usach, R., Palau, C., Esteve, M., Montesinos, M., Llorente, M. A., Gimenez, P., Llop, M.: Interoperability of IoT platforms applied to the transport and logistics domain. In: *Transport Arena Research Conference 2018*, April 2018
8. Ganzha, M., Paprzycki, M., Pawłowski, W., Szymeja, P., Wasielewska, K.: Towards semantic interoperability between Internet of Things platforms. In: Gravina, R., Palau, C.E., Manso, M., Liotta, A., Fortino, G. (eds.) *Integration, Interconnection, and Interoperability of IoT Systems*, pp. 103–127. Springer International Publishing, Cham (2018)
9. Fortino, G., Garro, A., Russo, W.: Achieving mobile agent systems interoperability through software layering. *Inf. Softw. Technol.* **50**(4), 322–341 (2008)
10. Moreira, J., Daniele, L., Pires, L.F., van Sinderen, M., Wasielewska, K., Szymeja, P., Pawłowski, W., Ganzha, M., Paprzycki, M.: Towards IoT platforms’ integration. In: *SEMANTiCS conference 2017*, November 2017
11. Moreira, J., Pires, L.F., van Sinderen, M., Wieringa, R., Singh, P., Costa, P.D., Llop, M.: Improving the semantic interoperability of IoT early warning systems: the port of valencia use case. In: Popplewell, K., Thoben, K.-D., Knothe, T., Poler, R. (eds.) *Enterprise Interoperability VIII*, pp. 17–29. Springer International Publishing, Cham (2019)
12. Truong, H.-L., Gao, L., Hammerer, M.: Service architectures and dynamic solutions for interoperability of IoT, network functions and cloud resources. In: *Proceedings of the 12th European Conference on Software Architecture: Companion Proceedings, ECSA’18*, pp. 1–4, New York, NY, USA, September 2018. Association for Computing Machinery
13. Truong, H.-L.: Dynamic IoT data, protocol, and middleware interoperability with resource slice concepts and tools: tutorial. In: *Proceedings of the 8th International Conference on the Internet of Things*, pp. 1–4, Santa Barbara California USA, October 2018. ACM
14. Modoni, G.E., Caldarola, E.G., Sacco, M., Wasielewska, K., Ganzha, M., Paprzycki, M., Szymeja, P., Pawłowski, W., Palau, C.E., Solarz-Niesłuchowski, B.: Integrating the AAL

- CasAware platform within an IoT ecosystem, leveraging the INTER-IoT approach. In: Singh, P.K., Pawłowski, W., Tanwar, S., Kumar, N., Rodrigues, J.J.P.C., Obaidat, M.S. (eds.) *Proceedings of First International Conference on Computing, Communications, and Cyber-Security (IC4S 2019)*, pp. 197–212, Singapore. Springer Singapore (2020)
15. Modoni, G.E., Caldarola, Nicola E.G., Mincuzzi, N., Sacco, M., Wasielewska, K., Szmeja, P., Ganzha, M., Paprzycki, M., Pawłowski, W.: Integrating IoT platforms using the INTER-IoT approach: a case study of the CasAware project. *J. Ambient Intell. Smart Environ.* 1–18 (2020)
 16. Adame, T., Bel, A., Bellalta, B.: Increasing LPWAN scalability by means of concurrent multi-band IoT technologies: an industry 4.0 use case. *IEEE Access* 7, 46990–47010 (2019). Conference Name: IEEE Access
 17. Vázquez, T.A., Barrachina-Muñoz, S., Bellalta, B., Bel, A.: HARE: supporting efficient uplink multi-hop communications in self-organizing LPWANs. *Sensors* 18(2), 115 (2018)
 18. Nagarajan, G., Minu, R.I., Giannoccaro, I. (eds.): *Advances in computational intelligence and robotics*. In: IGI Global, *Edge Computing and Computational Intelligence Paradigms for the IoT* (2019)
 19. Koumaras, V., Kapari, M., Papaioannou, A., Theodoropoulos, G., Stergiou, I., Sakkas, C., Koumaras, H.: IoT Interoperability on Top of SDN/NFV-Enabled Networks, pp. 127–152. IGI Global Publisher (2019). ISBN: 9781522585558
 20. Fotiou, N., Polyzos, G.C.: Authentication and authorization for interoperable IoT architectures. In: Saracino, A., Mori, P. (eds.) *Emerging Technologies for Authorization and Authentication*. Lecture Notes in Computer Science, pp. 3–16. Springer International Publishing, Cham (2018)
 21. Sukmana, M.I.H., Torkura, K., Meinel, C., Graupner, H.: Redesign cloudRAID for flexible and secure enterprise file sharing over public cloud storage. In: *Proceedings of the 10th International Conference on Security of Information and Networks, SIN'17*, October 2017
 22. Gürgen, L., Munilla, C., Druilhe, R., Gandrille, E., do Nascimento, J.B.: sensiNact IoT platform as a service. In: El Fallah Seghrouchni, A., Ishikawa, F., Hérault, L., Tokuda, H. (eds.) *Enablers for Smart Cities*, pp. 127–147. Wiley, Hoboken, NJ, USA (2016)
 23. Xu, S.S., Chen, C., Chang, T.: Design of oneM2M-based fog computing architecture. *IEEE Internet of Things J.* 6(6), 9464–9474 (2019). Conference Name: IEEE Internet of Things Journal
 24. Thielemans, S., Sartori, B., Braeken, A., Steenhaut, K.: Integration of oneM2M in Inter-IoT's platform of platforms. In: *2019 IEEE International Symposium on Local and Metropolitan Area Networks (LANMAN)*, pp. 1–2, July 2019. ISSN: 1944-0375
 25. Lacalle, I., Llorente, M. A., Palau, C.E.: Towards environmental impact reduction leveraging IoT infrastructures: the PIXEL approach. In: Montella, R., Ciaramella, A., Fortino, G., Guerrieri, A., Liotta, A. (eds.) *Internet and Distributed Computing Systems*, pp. 33–45. Springer International Publishing, Cham (2019)
 26. Gonzalez-Usach, R., Julian, M., Esteve, M., Palau, C.E.: IoT semantic interoperability for active and healthy ageing. In: *SIoT 2020: Semantic IOT: Theory and Applications - Interoperability, Provenance and Beyond*. Studies in Computational Intelligence. Springer (2020)
 27. 5G-PPP. H2020 5GENESIS project, October 2020
 28. IoT-EPI. Steps towards the IoT platform ecosystem maturity (2017)
 29. Arne, B., Zappa, A., Vermesan, O., Frámling, K., Zaslavsky, A., Gonzalez-Usach, R., Szmeja, P., Palau, C., et al.: *Advanced IoT Platforms Interoperability*. River Publishers, The Netherlands (2018)