Internet of Things - Architecture

IoT-A

Deliverable D1.3 – Updated reference model for IoT v1.5

Project acronym:	IoT-A
Project full title:	The Internet of Things - Architecture
Grant agreement no.:	257521

Doc. Ref.:	D1.3	
Responsible Beneficiary :	FhG IML	
Editor(s):	Andreas Nettsträter (FhG IML)	
List of contributors:	Martin Bauer (NEC), Mathieu Boussard (ALBLF), Nicola Bui (CFR), Francois Carrez (UniS), Pierpaolo Giacomin (HEU), Stephan Haller (SAP), Edward Ho (HSG), Christine Jardak (SIEMENS), Jourik De Loof (ALUBE), Carsten Magerkurth (SAP), Stefan Meissner (UniS), Andreas Nettsträter (FhG IML), Alexis Olivereau (CEA), Alexandru Serbanati (CATTID), Matthias Thoma (SAP), Joachim W. Walewski (SIEMENS)	
Reviewer(s):	Alessandro Bassi (HEU), Pierpaolo Giacomin (HEU)	
Contractual Delivery Date:	30.06.2012	
Actual Delivery Date:	16.07.2012	
Status:	Final	
Version and date	*Changes*	*Reviewers / Editors*
V1, 2012-07-16	Version 1: D1.3	Andreas Nettsträter (FhG IML)

Project co-funded by the European Commission within the Seventh Framework Programme (2007-2013)

	Dissemination level	PU
PU ~~PP~~ ~~RE~~ ~~CO~~	Public ~~Restricted to other programme participants (including the Commission Services)~~ ~~Restricted to a group specified by the Consortium (including the Commission Services)~~ ~~Confidential, only for members of the Consortium (including the Commission Services)~~	

Overview of the IoT-A project partners

Acronym	Full name	Country
ALBLF	Alcatel-Lucent Bell Labs France	FR
ALU BE	Alcatel-Lucent Bell N.V.	BE
CATTID	Università Sapienza di Roma	IT
CEA	Commissariat à l'Energie Atomique	FR
CFR	Consorzio Ferrara Ricerche	IT
CSE	Creative Systems Engineering	GR
FhG IML	Fraunhofer Institute for Materialflow and Logistics	DE
HEU	Hitachi Europe Ltd.	GB
HSG	University of St. Gallen	CH
IBM	IBM Research GmbH	CH
NEC	NEC Europe Ltd.	GB
NXP BE	NXP Semiconductors Belgium N.V.	BE
NXP DE	NXP Semiconductors Germany GmbH	DE
SAP	SAP AG	DE
SIEMENS	Siemens AG	DE
TID	Telefonica Investigacion y Desarrollo SA Unipersonal	ES
UniP	Universita degli studi di Padova	IT
UniS	University of Surrey	GB
UniWue	University of Wuerzburg	DE
VDI/VDE-IT	VDI/VDE Innovation + Technik GmbH	DE

Table of content

Overview of the IoT-A project partners ..... - 3 -
Table of content ..... - 5 -
List of abbreviations ..... - 9 -
Table of figures ..... - 11 -
1 Executive Summary ..... - 15 -
1.1 Objectives ..... - 17 -
1.2 Document structure ..... - 17 -
1.3 Project-internal inputs ..... - 18 -
2 Introduction ..... - 19 -
2.1 Usage of architectural reference models ..... - 21 -
2.1.1 Cognitive aid ..... - 21 -
2.1.2 IoT-A Reference Model as a common grounding ..... - 22 -
2.1.3 Generation of architectures ..... - 22 -
2.1.4 Identifying differences ..... - 22 -
2.1.5 Benchmarking ..... - 22 -
2.2 Process and Methodology ..... - 22 -
2.2.1 Introduction ..... - 22 -
2.2.2 Reference model and reference architecture ..... - 23 -
2.2.3 Actions and inputs ..... - 24 -
2.2.4 Overall process ..... - 25 -
2.3 Business Scenarios ..... - 31 -
2.3.1 Rationale and Introduction ..... - 31 -
2.3.2 Fields of Application ..... - 33 -
2.3.3 Business Case Methodology ..... - 36 -
2.3.4 Retail Business Case ..... - 37 -
3 Reference model ..... - 41 -
3.1 Interaction of all sub-models ..... - 41 -
3.2 Domain Model ..... - 42 -

3.2.1 Definition and Purpose ..... - 42 -
3.2.2 Main abstractions and relationships ..... - 42 -
3.2.3 Detailed explanations and related concepts ..... - 46 -
3.3 Information model ..... - 49 -
3.3.1 Relation of Information Model to Domain Model ..... - 51 -
3.3.2 Data in IoT systems ..... - 53 -
3.3.3 Other information-related models in IoT-A ..... - 53 -
3.4 Functional model ..... - 53 -
3.4.1 Functional decomposition ..... - 53 -
3.4.2 Functional Model Diagram ..... - 54 -
3.5 Communication model ..... - 59 -
3.5.1 Communication stack ..... - 59 -
3.5.2 Actors in IoT communication ..... - 61 -
3.5.3 Channel model for IoT communication ..... - 61 -
3.5.4 IoT Communication model as seen from the application level ..... - 63 -
3.6 Trust, Security and Privacy ..... - 65 -
3.6.1 Trust ..... - 66 -
3.6.2 Security ..... - 68 -
4 Reference architecture ..... - 71 -
4.1 Short definition of views and perspectives ..... - 71 -
4.2 Views ..... - 71 -
4.2.1 Usage for the IoT-A Reference Architecture ..... - 72 -
4.2.2 Functional ..... - 72 -
4.2.3 Information ..... - 90 -
4.2.4 Deployment & Operation ..... - 93 -
4.3 Perspectives ..... - 96 -
4.3.1 Evolution and interoperability ..... - 97 -
4.3.2 Performance and scalability ..... - 97 -
4.3.3 Trust, Security and privacy ..... - 98 -
4.3.4 Availability and resilience ..... - 101 -

5 Best practices ..... - 103 -
5.1 Overview ..... - 103 -
5.2 Usage of the IoT Reference Model ..... - 103 -
5.2.1 Guidelines for using the IoT Domain Model ..... - 103 -
5.2.2 Examples for IoT Domain Model objects ..... - 113 -
5.3 Usage of the IoT Reference Architecture ..... - 117 -
5.3.1 Design choices ..... - 117 -
5.3.2 Risk analysis ..... - 132 -
6 Conclusions and Outlook ..... - 143 -
References ..... - 145 -
Appendix ..... - 151 -
A Terminology ..... - 151 -
B Requirements ..... - 161 -
B.1 Requirements Gathering Methodology ..... - 161 -
B.1.1 Gathering external requirements from stakeholders ..... - 161 -
B.1.2 Gathering internal requirements ..... - 163 -
B.1.3 Unification process ..... - 163 -
B.2 Unified requirements list ..... - 163 -
C Use cases, sequence charts and interfaces ..... - 197 -
C.1 IoT Business Process Management and Service Organisation ..... - 197 -
C.1.1 IoT Business Process Management ..... - 197 -
C.1.2 Service Organisation ..... - 199 -
C.2 IoT Services ..... - 203 -
C.2.1 IoT Service Resolution functional component ..... - 203 -
C.3 Virtual Entity (VE) ..... - 223 -
C.3.1 Virtual Entity Resolution functional component ..... - 223 -
C.3.2 Virtual Entity and IoT Service Monitoring functional component ..... - 239 -
C.4 Security ..... - 247 -
C.4.1 IoT Service Resolution functional component ..... - 247 -

List of abbreviations

a.k.a.	Also Known as
API	Application-programming interface
ARM	Architectural Reference Model
AuthN	Authentication
AuthS	Authorisation
BC	Business Case
BPM	Business Process Management
BSN	Body Sensor Network
CA	Certification Authority
CCTV	Closed-Circuit TeleVision
CD	Constrained Device
CFG	Communication Functionality Group
CIM	Common Information Model
CO	Carbone monoxide
CP	Control Point
D*n.m*	IoT-A deliverable *n.m*
DNS	Domain Name System
DoS	Denial of Service
DP	Data Processor
DS	Data Sink
EMR	Electronic Medical Record
EPC	Electronic Product Code
EPCIS	Electronic Product Code Information Services
ERP	Enterprise Resource Planning
FC	Functional Component
FG	Functionality Group
GPS	Global Positioning System
GW	Gateway
ICT	Information and Communication Technology
ID	Identity
IoT	Internet of Things
IoT-A	Internet of Things - Architecture
ISO	International Organization for Standardization
IT	Information Technology
KEM	Key Exchange and key Management
LED	Light-Emitting Diode
M2M	Machine-to-Machine
MDA	Model Driven Architecture
MDE	Model Driven Engineering

NFC	Near Field Communication
NTC	Constrained network
NTU	Unconstrained network
OASIS	Organization for the Advancement of Structured Information Standards
OMG	Object Management Group
OPEX	OPerational EXpenditure
ONS	Object Naming Service
OS	Operating System
OSI	Open System Interconnection
OWL	Web Ontology Language
PE	Physical Entity
QoS	Quality of Service
QR	Quick Response
R&D	Research and Development
RDF	Resource Description Framework
RDFa	RDF-in-attributes
RF	Radio Frequency
RFID	Radio-Frequency IDentification
RS	Resolution Server
S&AN	Sensor and Actuator Networks
SO	Service Organisation
SW*n*	Stakeholder workshop *n*
TRA	Trust and reputation
UNI.*n*	UNIfied requirement, number *n*
USDL	Unified Service Description Language
VE	Virtual Entity
VE-ID	Virtual Entity IDentifier
WP	Work Package
WSN	Wireless Sensor Network
WS&AN	Wireless Sensor & Actuator Network

Table of figures

Figure 1: IoT-A architectural reference model building blocks. - 16 -

Figure 2: The IoT-A Tree - 20 -

Figure 3: Relationship between a reference architecture, architectures, and actual systems (adapted from Mueller [1]). - 24 -

Figure 4: Relation of an architectural reference model, best practice, and concrete architectures. - 24 -

Figure 5: High-level taxonomy of the IoT-Reference-Model and IoT-Reference-Architecture dependencies and model influences. - 25 -

Figure 6: Dynamic view of the IoT-A ARM process. - 26 -

Figure 7: Process for the generation of concrete architectures. - 28 -

Figure 8: Generalised architecture approach according to the Model-Driven-Architecture methodology, a.k.a. Model-Driven Engineering [2]. - 29 -

Figure 9: Relation of the Best-Practice-driven derivation of concrete architectures from an architectural reference model and the derivation of implementations from said concrete architecture. This Figure is a composite of Figure 4 and Figure 8. - 29 -

Figure 10: Interaction of all sub-models - 41 -

Figure 11: Basic abstraction of an IoT interaction. - 43 -

Figure 12: The IoT Domain Model - 46 -

Figure 13: Information Model - 50 -

Figure 14: Relation between Domain Model and Information Model - 52 -

Figure15: Functional Model - 55 -

Figure 16: IoT Service and Virtual Entity abstraction levels - 58 -

Figure 17: IoT communication stack. - 60 -

Figure 18: Schematic diagram of a general communication system. - 61 -

Figure 19: Channel model for the current Internet. - 62 -

Figure 20: IoT channel for a single constrained network. - 62 -

Figure 21: IoT channel for communication over two constrained networks. - 62 -

Figure 22: IoT channel for communication constrained to unconstrained networks. - 63 -

Figure 23: IoT channel for communication over two constrained networks intermediated by the Internet. - 63 -

Figure 24: Communications in the IoT domain model from an application point of view. AppNode: application node; GW: gateway; CP: control point; DS: data sink. - 64 -

Figure 25: Security features and general layering. Some architectures can exhibit a slightly different approach, depending on the actual implementation. For example, some optional components might not have been implemented while some features could have been implemented in a cross-layered approach. ..... - 68 -

Figure 26: Providing the best security features for the lower layers in each IoT domain by introducing Gateways with adaptive functions aimed to provide scalability functions (including security scalability). NTC: Constrained Device Network; NTU: Unconstrained Device Network. CDSecFeat: implementation of security feature for the constrained device leverages the extension of the functionalities of gateway devices. ..... - 69 -

Figure 27: Functional view process ..... - 72 -

Figure 28**:** Functional View ..... - 74 -

Figure 29: Example for flat entity type model ..... - 90 -

Figure 30: Example for hierarchical entity type model ..... - 91 -

Figure 31 Domain model groups ..... - 93 -

Figure 32: Various deployment configurations of devices, resources, and services. ..... - 104 -

Figure 33: Data-base pattern as an example for an Augmented Entity. ..... - 106 -

Figure 34: Smart-object pattern. UAV: unmanned aerial vehicle. ..... - 107 -

Figure 35: Multiple Virtual Entities (data-base entries) for a single car. ..... - 108 -

Figure 36: Exemplary modelling of a smart phone that is used as tracking device. ..... - 109 -

Figure 37: Domain model instantiation for a M2M communication scenario ..... - 110 -

Figure 38: M2M communication. ..... - 111 -

Figure 39: Shipping box containing multiple packets. The VE-to-PE mapping is exemplified by paper tags. ..... - 112 -

Figure 40: Domain modelling of a typical EPC-based RFID scenario (pallet containing cases). .. - 112 -

Figure 41: Growth fruit sensor [3]. ..... - 114 -

Figure 42: Interacting services for a home-patient monitoring scenario. ..... - 116 -

Figure 43: Telos ultra-low power wireless module. ..... - 117 -

Figure 44: Evolution of requirements lists towards unified requirements list in D1.3 ..... - 161 -

Figure 45: Development process for Requirements ..... - 162 -

Figure 46: Further development of requirements and validation approach ..... - 162 -

Figure 47: Business Process Execution. ..... - 199 -

Figure 48: Service Organization. ..... - 201 -

Figure 49: Orchestrate Service. ..... - 202 -

Figure 50: Decompose Composite Service. - 203 -
Figure 51: Use case IoT Service Resolution. - 207 -
Figure 52: Resolve Service Identifier to URL. - 208 -
Figure 53: Subscribe Resolution of Service Identifier to URL - 209 -
Figure 54: Lookup Service Description based on Service Identifier. - 210 -
Figure 55 Subscribe Look-up of Service Description based on Service Identifier - 211 -
Figure 56: Discover Service based on Service Specification. - 212 -
Figure 57 Subscribe Discovery of Service Descriptions based on Service Specification - 213 -
Figure 58: Insert Service Description. - 214 -
Figure 59: Update Service Description. - 214 -
Figure 60: Delete Service Description. - 215 -
Figure 61: Virtual Entity Resolution. - 226 -
Figure 62: Look up Associations based on VE-ID and VEServiceSpecification. - 227 -
Figure 63 Subscribe Look-up of Associations for VE Identifier and VE Service Specification - 228 -
Figure 64: Discover Associations based on VE Specifications and VEServiceSpecifications. - 229 -
Figure 65 Subscribe Discovery of Associations based on VE Specification and VE Service Specification - 230 -
Figure 66: Insert Association. - 231 -
Figure 67: Update Associations. - 232 -
Figure 68: Delete Association. - 233 -
Figure 69: Virtual Entity & IoT Service Monitoring. - 240 -
Figure 70: Assert Static VE-IoT Service Association. - 241 -
Figure 71: Discover Dynamic Associations between VEs and Services. - 242 -
Figure 72: Monitor and Update Existing Dynamic Associations. - 243 -
Figure 73: Monitor and Delete Existing Dynamic Associations. - 244 -
Figure 74: Secure discovery of IoT services. - 248 -
Figure 75: Secure Direct Discovery of IoT Services. - 249 -
Figure 76: Restricted discovery. - 251 -
Figure 77: Restricted Lookup. - 252 -

1 Executive Summary

Today, "Internet of Things" (IoT) is used as a catchphrase by many sources. This expression encompasses a galaxy of solutions somehow related to the world of intercommunicating and smart objects. These solutions show little or no interoperability capabilities as usually they are developed for specific challenges in mind, following specific requirements. Moreover, as the IoT umbrella covers totally different application fields, development cycles and technologies used vary enormously, thus implementing vertical solutions that can be labelled as "INTRAnet of Things", rather than "INTERnet of Things". For instance, in some fields such as manufacturing and logistics, communication and tagging solutions are well established as they provide a clear business benefit in terms of asset tracking and supply-chain management. However, the same solutions do not apply for other fields such as domestics, where business synergies could provide services with clear added-value benefits.

While quite logical at this point, on the long run we believe that this situation is unsustainable. As in the networking field, where several solutions emerged at his infancy to leave place to a common model, the TCP/IP protocol suite, the emergence of a common reference model for the IoT domain and the identification of reference architectures can lead to a faster, more focused development and an exponential increase of IoT-related solutions. These solutions can provide a strategic advantage to mature economies, as new business models can leverage those technological solutions providing room for economic development.

Leaving aside business considerations, and considering only the technical point of view, the existing solutions do not address the scalability requirements of a future IoT, both in terms of communication between and the manageability of devices. Additionally, as the IoT domain comprises several different governance models, which are often incompatible. This leads to a situation where privacy and security are treated on a per-case and per-legislation basis, retro-fitting solutions to existing designs, and this severely hampers portability, interoperability and deployment.

In our vision of the Internet of Things, the interoperability of solutions at the communication level, as well as at the service level, has to be ensured across various platforms.

This motivates, first, the creation of a **Reference Model** for the IoT domain in order to promote a common understanding.

Second, businesses that want to create their own compliant IoT solutions should be supported by a **Reference Architecture** that describes essential building blocks as well as design choices to deal with conflicting requirements regarding functionality, performance, deployment and security. Interfaces should be standardised, best practices in terms of functionality and information usage need to be provided.

The central choice of the IoT-A project was to base its work on the current state of the art, rather than using a clean-slate approach. Due to this choice, common traits are derived to form the base line of the ***Architectural Reference Model*** **(ARM)**. This has the major advantage of ensuring backward-compatibility of the model and also the adoption of established, working solutions to various aspects of the IoT. With the help of end users, organised into a stakeholders group, new requirements for IoT have been collected and introduced in the main model building process. This work was conducted according to established architecture methodology. defined in Section 2.2.

Figure 1 shows an overview of the process we used for defining the different parts that constitute the IoT Architectural Reference Model (ARM). Notice that definitions of terms such as reference architecture, etc. can be found in an external glossary [4] Starting with existing architectures and solutions, generic baseline requirements can be extracted and used as an input to the design. The IoT-A ARM consists of four parts:

- The **vision** summarises the rationale for providing an architectural reference model for the IoT. At the same time it discusses underlying assumptions, such as motivations. It also discusses how the architectural reference model can be used, the methodology applied to the architecture modelling, and the business scenarios and stakeholders addressed. The vision is described in Section 1.

- **Business scenarios & stakeholders** are the drivers of the architecture work. With the knowledge of businesses aspirations, a holistic view of IoT architectures can be derived. Furthermore, a concrete instance of the reference architecture can be validated against selected business scenarios. A stakeholder analysis contributes to understanding which aspects of the architectural reference model need to be described for the different stakeholders and their concerns. More information on the Business Scenarios & Stakeholders is provided in Section 2.2. According to common usage, this part constitutes a subset of the vision [5]

- The **IoT Reference Model** provides the highest abstraction level for the definition of the IoT-A Architectural Reference Model. It promotes a common understanding of the IoT domain. The description of the IoT Reference Model includes a general discourse on the IoT domain, an IoT Domain Model as a top-level description, an IoT Information Model explaining how IoT knowledge is going to be modelled, and an IoT Communication Model in order to understand specifics about communication between many heterogeneous IoT devices and the Internet as a whole. The definition of the IoT Reference Model is conforming to the OASIS reference model definition [6]. A detailed description of the IoT Reference Model is provided in Section 2.

- The **IoT Reference Architecture** is the reference for building compliant IoT architectures. As such, it provides views and perspectives on different architectural aspects that are of concern to stakeholders of the IoT. The terms view and perspectives are used according to the general literature and standards [7],[8]. Definitions of these terms are also provided in Section 0. The creation of the IoT Reference Architecture focuses on abstract sets of mechanisms rather than concrete application architectures.

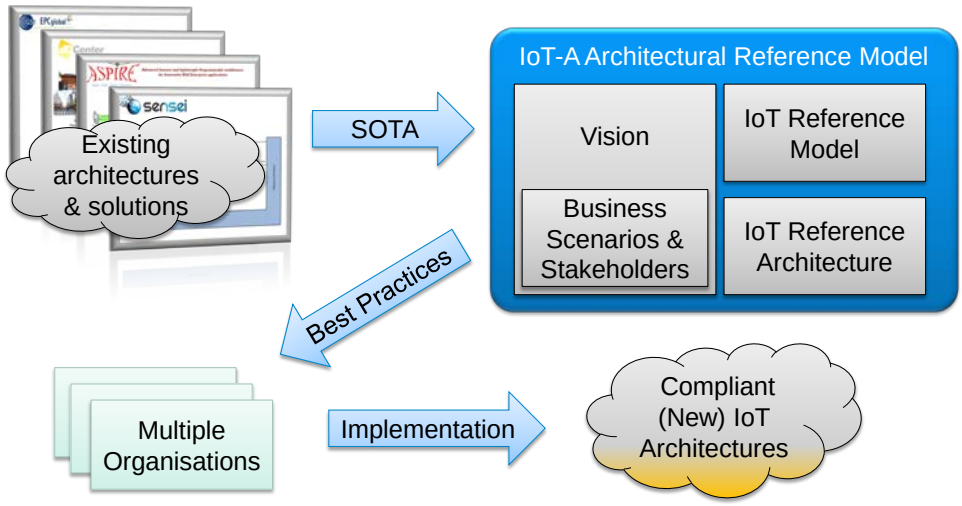

Figure 1: IoT-A architectural reference model building blocks.

To organisations, an important aspect is the compliance of their technologies with standards and best practices, so that interoperability across organisations is ensured. If such compliance is given, an ecosystem forms, in which every stakeholder can create new businesses that "interoperate" with already existing businesses. The IoT-A ARM provides best practices to the organisations so that they can create compliant IoT architectures in different application domains. Those IoT architectures are instances from the Reference Architectures with some architectural choices (called later on *Design Choices*) like considering strong Real/Time or choosing strong security features, etc. They consist of special "flavours" of the IoT Reference Architecture. Where application domains are overlapping, the compliance to the IoT Reference Architecture ensures the interoperability of solutions and allows the formation of new synergies across those domains.

The rest of this section organises as follows. In Section 1.1 we remind shortly the objectives of D1.3, emphasising on the improvements it brings to D1.2 Then in Section 1.2 we outline the structure of the document. Finally Section 1.3 gives some hints about the various project input documents used for writing this deliverable.

1.1 Objectives

D1.3 is the Second public version of the Architectural Reference Model. It leverages on D1.2 and an intermediary internal report IR1.4 release internally earlier this year.

While the general objective of D1.3 is strictly the same than D1.2 i.e. describing thoroughly an Architectural Reference Model for IoT, this version of the ARM brings to the audience critical improvements to its previous version, as it is summarised below:

- All feedback received internally from IoT-A and externally from the Stakeholders was taken into account in order to improve the document and in order to make sure that the IoT-A architecture work will eventually meet expectations from the external users;
- Introduction of news views and perspectives as only the Functional Decomposition view and Security Perspective were touched in D1.2. D1.3 comes with the Deployment & Operation and Information views and with the Evolution & Interoperability, Performance & Scalability and Availability & Resilience perspectives. It will be shown in the document how the Design Choices applied at the view levels impact the various quality properties attached to the system architecture materialised by the four perspectives introduced above;
- First version of Best Practices and associated Design Choices which are a first step towards an aided architecture design for concrete system architects;
- Improvement of the soundness of the whole ARM approach, emphasizing the logical links existing between the various models of the Reference models and the views and perspectives of the Reference Architecture.

1.2 Document structure

The table of content of D1.3 does not strictly follows the one of D1.2 as we can see below:

Section 1 gives a short and general introduction to the document and shows how it positions it-self with regards to D1.2.

Section 2 gives a more complete introduction to the IOT-A vision and philosophy. It provides the reader with some elements of discourse and general concerns about what is the ARM, how it was elaborated (elements of methodology), how it can be used (usage, benefits of using it) and where it potentially can apply (business scenarios, field of application) and envisioned impacts.

Section 3 gives the full detail about the updated Reference Model, starting with an informal discourse about the IOT domain, emphasising specific challenges coming with the IOT field and giving explanations about how the sub-models interact. Then this section introduces the Domain model. The rest of section is dedicated to the Communication model, Information model, Functional model and model relating to Trust, Security and Privacy.

Section 4 is dedicated to the Reference Architecture (which found its grounding in Section 3). Following the element of methodology explained in Section 2 and 3.1, this section provides a set of Views (functional decomposition, Information, Deployment & Operation) and Perspectives (Security & Privacy, Evolution & Interoperability, Performance & Scalability and Availability & Resilience).

Section 5 is dedicated to Best Practices and Design Choices and can be considered as a Cookbook for the IOT application architects to use. It gives indications and modelling rules on how to use the IoT-A RM, all illustrated with concrete examples (Sub-section 5.2). Then this section explains how to use the IoT-A Reference Architecture. For that purpose it provides the architects with a large number of Design Choices that can be used by an architects to build up a concrete architecture, depending on various aspects and properties of the targeted system (Sub-section 5.3.1). This sections also provides a Risk Analysis that guides the architects in making their IOT-system secured.

Then follow some appendixes; Appendix A gives a glossary of terms along with their definitions. Appendix B is about Requirements (Requirement Methodology (B1) and list of unified requirements (B2)); Appendix C gives Use cases with associated sequences diagrams and interfaces that correspond to the functional groups identified in Section 4.

1.3 Project-internal inputs

This document draws heavily on the following public IoT-A deliverables and internal reports:

- D6.1, which contains a summary of the IoT-A requirements-engineering process and a first list of requirements inferred from stakeholder aspirations provided during the first IoT-A stakeholder workshop in Paris in October 2010 [9]. This requirements list was analysed and views and perspectives were assigned to all requirements. The list of unified requirements can be found in Appendix B.

- D1.1, which contains a summary of the state of the art of IoT-related architectures, service interfaces, communication layers, resolution infrastructures, and hardware [10]. Each of the aforementioned topics is divided into input gathered from standardisation, commercial applications, and EU and other research projects. This document was used for the inference of technical requirements pertaining to the IoT architectural reference model.

- IR1.4 (Confidential and Internal): builds upon D1.2 taking into account all feedback received internally from the WPs and externally during the Stakeholders workshops.

- IR2.1, IR3.1, IR4.1: which contain the detailed feedback from those work package w.r.t. the first IOT Reference Architecture document D1.2.

- IR6.1: which contains the feedback collected during the second stakeholder workshop in Barcelona (during the IoT Week'2011)

Furthermore, as already mentioned, IoT-A provides a web page on which all the IoT terminology (see Appendix A) that is used in this deliverable (and will be used in forthcoming IoT-A deliverables) is listed [4].

2 Introduction

A commonly observed trend in the Internet of Things (IoT) domain is the emergence of a variety of communication solutions targeted at specific application domains. Many popular “umbrella” topics like Smart Cities pull a large number of specific domains of applications like Transportation, Energy, Environment, Assisted Living, most of time pre-fixed with “Smart” in order to emphasise the fact they embed a sort of intelligence and global awareness… This new breed of application exploits the full potential of IoT related technologies, however unfortunately, the resulting applications appear as vertical silos only, meaning specific applications with specific architectures, with little place left for inter-system communication and inter-operation. Actually that is where the real issue stands: the smartness of those new applications can only reach its pinnacle whenever full collaboration between those vertical silos can be achieved.

If we consider also the fact that IoT related technologies come with a high level of heterogeneity, with specific protocols developed with specific applications in mind, it results that the IoT landscape nowadays appears as highly fragmented. Many IoT-enabled solutions exist with recognised benefits in terms of business and social impact, however they form what we could call a set of **Intra**nets of things, not an **Inter**net of things!

In the vision of the Internet of things IoT-A wants to promote, high level of interoperability needs to be reached at the communication level as well as at the service and even knowledge levels across different platforms established on a common grounding. The IoT-A project reckons that achieving those goals comes in two steps, first of all in establishing a common understanding of the IoT domain (hereafter called Reference Model), and second in providing to IoT system developers a common foundation for establishing the IoT system architecture (hereafter called Reference Architecture).

While existing literature like [11], [12] and [13] (to name just a few) provide methodologies for dealing with system architectures (hereafter called Concrete Architectures) based on Views and Perspectives for instance (those concepts are developed in further sections of this document), establishing a reference architecture is a totally different business, at least as far as describing Views and Perspectives is concerned as we will see in the rest of this document.

A *Reference Architecture* (RA) can be visualised therefore as the *matrix* that eventually gives birth ideally to all concrete architectures. For establishing such a matrix, based on a strong and exhaustive analysis of the State of the Art, we need to envisage the super-set of all possible functionalities, mechanisms and protocols that can be used for building such concrete architecture and to show how interconnections could take place between selected ones (as no concrete system is likely about to use all of the functional possibilities). Giving such a foundation along with a set of design-choices, based on the characterisation of the targeted system w.r.t. various dimensions (like distribution, security, real-time, semantics,…) it becomes possible for a system architect to select the protocols, functional components, architectural options, needed to build their IoT systems. The main aim of IoT-A can be explained using the pictorial representation shown in Figure 2 below.

Figure 2: The IOT-A Tree

As in any metaphoric representation, this tree does not claim to be fully consistent in its depiction it should therefore not be taken too strictly: on the one hand, the roots of this tree are spanning across a selected set of communication protocols (6lowpan, Zigbee, IPv6,…) and device technologies (sensors, actuators, tags,..) while on the other hand the flowers/leaves of the tree represents the whole set of IoT applications that can be built from the sap (information/knowledge) coming from the roots. The trunk of the tree is of the utmost importance here, beyond the fact it represents the IoT-A project. This trunk represent the Architectural Reference Model (which means here Reference Model + Reference Architecture a.k.a. ARM), the set of models, guidelines, best practices, views and perspectives that can be used for building fully interoperable IoT Concrete architecture (and therefore systems). In this tree, we aim at selecting a minimal set of interoperable technologies (the roots) and proposing the potentially necessarily set of enablers or building blocks etc…(the trunk) that enable the creation of a maximal set of interoperable IoT systems (the leaves).

The deliverable D1.2, which was released one year ago approximately, was presented to a large audience during the IoT week'2011 in Barcelona. As a result we received a large number of comments, the majority of them being taken into account already in this new version of the ARM. D1.3 also adds critical improvements to D1.2 within the various models of the Reference Model as well as at the Reference Architecture level, adding more views, identifying and describing Best Practice and Design Choices and their potential impacts on quality properties of the targeted system (perspectives).

The ultimate aim of the Reference Architecture work is to make sure that concrete system designers will eventually use it. High attention is therefore paid to ensuring the soundness of our work. In particular this version of the ARM aims at making more explicit the various links existing between the various models, views and perspectives, so that the final users finds a logic in the process of using the ARM.

The rest of these sections give more details on the architectural process and methodology. In Section 2.1 different benefits of having an ARM are explained. Some possible usages that can be made of this model and reference architecture are also introduced. Section 2.2 gives some explanations about the methodology used for building ARM. Finally, Section 2.2 introduces the relevant business scenarios and stakeholders and gives a hint at the fields of application where IOT technologies potentially apply.

The content of the document then reads as follows: Section 3 and 4 are respectively presenting the updated Reference Model and Reference Architecture; Section 5 is dedicated to Best Practice and Design Choices that helps a system designer to select the necessary components needed for their concrete architectures. Section 6 finally draws conclusions and further steps leading to the next release of the IoT-A ARM. The annexes give respectively an update of the terminology, an update of requirements used to drive the ARM work and also a set of Use Cases/Sequence Charts and Interfaces relating directly to the RA.

2.1 Usage of architectural reference models

This section provides a non-exclusive list of the beneficial uses of the IoT-A ARM.

2.1.1 Cognitive aid

When it comes to product development and other activities, an architectural reference model is of fourfold use.

First, it aids in guiding discussions, since it provides a language everyone involved can use, and which is intimately linked to the architecture, the system, the usage domain, etc.

Second, the high-level view provided in such a model is of high educational value, since it provides an abstract but also rich view of the domain. Such a view can help people new to the field with understanding the particularities and intricacies of IoT.

Third, the ARM can assist IoT project leaders in planning the work at hand and the teams needed. For instance, the Functionality Groups identified in the Functional View of the IoT system can also be understood as a list of independent teams working on an IoT system implementation.

Fourth, the ARM aids in identifying independent building blocks for IoT systems. This constitutes very valuable information when dealing with questions like system modularity, processor architectures, third-vendor options, re-use of already developed components, etc.

2.1.2 IoT-A Reference Model as a common grounding

Establishing a common grounding for a field is not an easy task. In order to be effective, such a grounding has to capture as many pertinent vantage points as possible. Establishing the common grounding encompasses the definition of IoT entities and describing their basic interactions and relationships with each other. The Architecture Reference Model is providing exactly such a common grounding for the IoT field. Any party envisaging to develop an IoT system that is IoT-A compatible must build on the common concepts provided in the IoT Reference Model.

2.1.3 Generation of architectures

Another benefit is the use of the IoT ARM for the generations of compliant architectures for specific systems. This is done by providing best practices for the translation of the ARM into concrete architectures. The benefit of such a generation scheme for IoT architectures is not only the automatism of this process, and thus the saved R&D efforts, but that the generated architecture will intrinsically provide interoperability of the derived IoT systems [14], [15].

2.1.4 Identifying differences

When using the aforementioned system-generation tools, which are based on the IoT-A ARM, any differences in the derived architectures can be attributed to the particularities of the pertinent use case [14]. When applying the IoT ARM, predictions of system complexity, etc. are available for the system parts to be implemented. That makes judging the overall implementation effort for use case implementation easier, and some projects that might not have been realised due to uncertainties in the project plan might become possible. The overall implementation effort is most certainly less than developing an architecture without the help of an architectural reference model.

2.1.5 Benchmarking

Another important use is benchmarking. For example, NASA used a reference architecture of its new exploration vehicle for better benchmarking tenders it was going to receive during a public bidding process [16]. While the reference model prescribes the language to be used in the systems/architectures to be assessed, the reference architecture states the minimum (functional) requirement on the systems/architectures. By standardising the description and also the ordering and delineation of system components and aspects, it also provides a high level of transparency and inherent comparability to the benchmarking process.

2.2 Process and Methodology

2.2.1 Introduction

This section provides a meta-perspective of IoT-A process, i.e. a look at how the IoT ARM model was derived. First, we need to understand why the derived reference architecture needs to be accompanied by a reference model, before we discuss how the parts of the IoT ARM have been developed.

Through the development of an architecture, a solution to a pre-defined goal is found. The development and description of architectures in turn is a modelling exercise. It is important to point out that the modelling itself does not happen in a vacuum, but rests on a thorough understanding of the domain modelled. In other words, any architecture development is contingent on one's understanding of the domain in question. The same is true for a generalisation of this process, i.e. the derivation of reference architectures. Thus, reference architectures, as the one presented in this deliverable, also have to be based on a detailed understanding of the domain in question. This understanding is commonly provided in the form of a reference model.

The above discourse motivates why the reference architecture presented in this deliverable is preceded by a thorough discussion of the IoT domain in the form of a reference model. However, this high-level view does not explain how one derives either. What is needed here are both a process and a methodology for deriving the parts of the ARM. The process describes what steps need to be undertaken during the derivation of the architectural reference model, and the methodology describes how these steps are achieved. In other words, the methodology describes, how to identify the tasks attached to each development step, and how and in which order to conduct these steps. Both the process and the methodology description are provided in this section.

The remainder of the text in this section is organised as follows. To start with, we provide a short discussion of the particularities of reference architectures and how they relate to concrete architectures, and also how they relate to reference models. This information enables us to discuss what high-level actions and input is needed for the derivation of an ARM, and what input is needed in order to guide the transformation of the reference architecture into use-case- and application-specific architectures, also called concrete architectures in the following. With this knowledge at hand, we dive into the details of the development process. First, we restate the goals of IoT-A and how we translated them into a step-by-step process. Next, we discuss the methodologies available for conducting each step. As it turns out, there is no standardised methodology for the derivation of ARMs. In order to overcome this lack of ARM methodology, we assessed the well-equipped toolboxes for the development of use-case- and application-specific architectures instead. Since these methods intrinsically rely on the specifity of the pertinent use cases and application scenarios, it is found that the method considered, for instance model-driven engineering, cannot always be applied one to one. This section concludes with a detailed discussion of our requirements process, which is at the heart of our entire architecture process.

2.2.2 Reference model and reference architecture

Reference models and reference architectures provide a description of greater abstraction than what is inherent to actual systems and applications. They are more abstract than system architectures that have been designed for a particular application with particular constraints and choices. From the literature, we can extrapolate the dependencies of reference architecture, architectures, and actual systems (see Figure 3) [1]. Architectures do help in designing, engineering, building, and testing actual systems. At the same time, understanding system constraints better can provide input to the architecture design, and in turn this allows identifying future opportunities. The structure of the architecture can be made explicit through an architecture description, or it is implicit through the system itself. By extracting essentials of existing architectures, like mechanisms or usage of standards, a reference architecture can be defined. Guidance in form of best practices can be associated to a reference architecture in order to derive use-case-specific architectures from the reference architecture (see Figure 4). Such guidance can, for instance, make new architectures and systems compliant to each other. These general architecture dependencies apply to the modelling of the IoT domain as well.

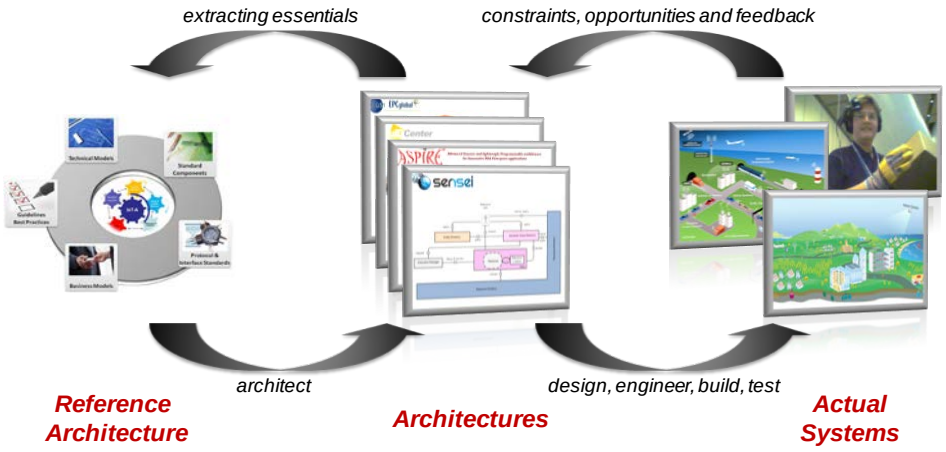

Figure 3: Relationship between a reference architecture, architectures, and actual systems (adapted from Mueller [1]).

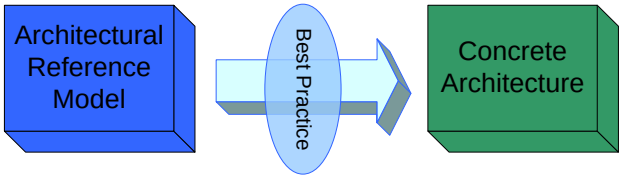

Figure 4: Relation of an architectural reference model, best practice, and concrete architectures.

While the model presented in Figure 3 stops at the reference architecture, the IoT-A architectural reference model goes one step beyond and also defines a reference model. As already discussed earlier, a reference model provides the grounding for a common understanding of the IoT domain by modelling its concepts and their relationships. A detailed description of the IoT Reference Model can be found in Section 2.

2.2.3 Actions and inputs

In the previous section we discussed how reference architectures relate to architectures and real systems. In order to derive such a reference architecture and the reference model upon which the reference architecture builds, one needs to understand better how they relate to each other and to external input.

A high-level taxonomy of how we understand the reference architecture process is depicted in Figure 5. Such a taxonomy already provides us with a high-level perspective of actions and inputs needed for developing an ARM for IoT. As discussed earlier, the IoT Reference Model provides guidance for the description of the IoT Reference Architecture. The Best Practice guides the derivation of IoT-A-compliant domain-specific concrete architectures from the reference architecture.

Essential inputs for the definition of the IoT Reference Model are stakeholder concerns, business scenarios, and existing architectures. It is important to create a common understanding of the IoT domain from the different inputs. This is mainly a modelling exercise, during which experts have to work together and extract the main concepts and their relations of the IoT domain from available knowledge.

Furthermore, business scenarios, existing architectures, and stakeholder concerns can be transformed into application-specific requirements. When extrapolated, these requirements lead to a set of unified requirements. Unified requirements in turn steer the definition of the IoT Reference Architecture.

Within the ARM, the IoT Reference Model guides the definition of the IoT Reference Architecture, creating dependencies between the Reference Architecture and the Reference Model; once a change is proposed in the Reference Model a clear chain of dependencies can be followed and lead to subsequent changes within the Reference Architecture. By so doing, an overall consistency of the IoT-A ARM is maintained.

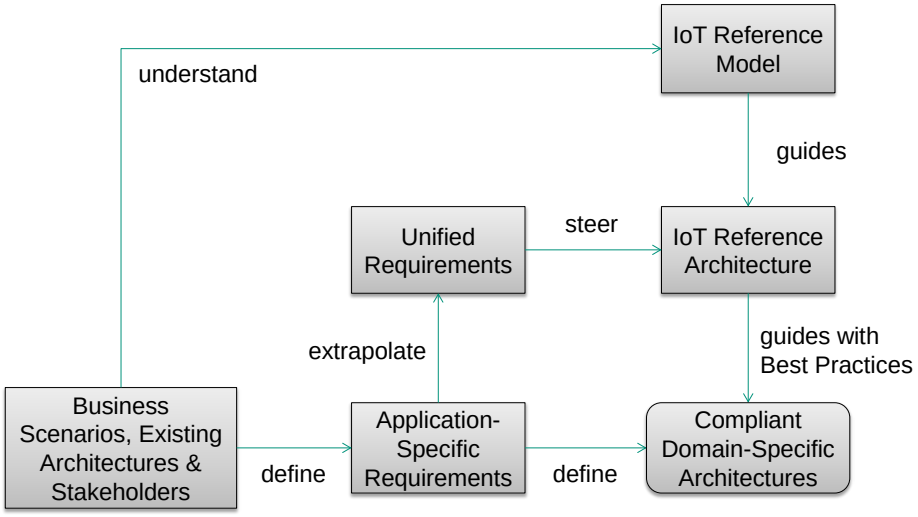

Figure 5: High-level taxonomy of the IoT-Reference-Model and IoT-Reference-Architecture dependencies and model influences.

As one can see, this high-level taxonomy already identifies high-level actions for the derivation of the ARM and for domain-specific architectures ("understand", "define", etc.). However this view is still too abstract for being of use in the day-to-day development work of the project. What is needed is a detailed architecture process that identifies individual tasks within the development process, that provides insight in the dependencies of said tasks, and that provides a dynamic model of the development process itself (viz. what step follows after the next).

2.2.4 Overall process

2.2.4.1 ARM development

A process-based view of the ARM derivation is shown in Figure 6.

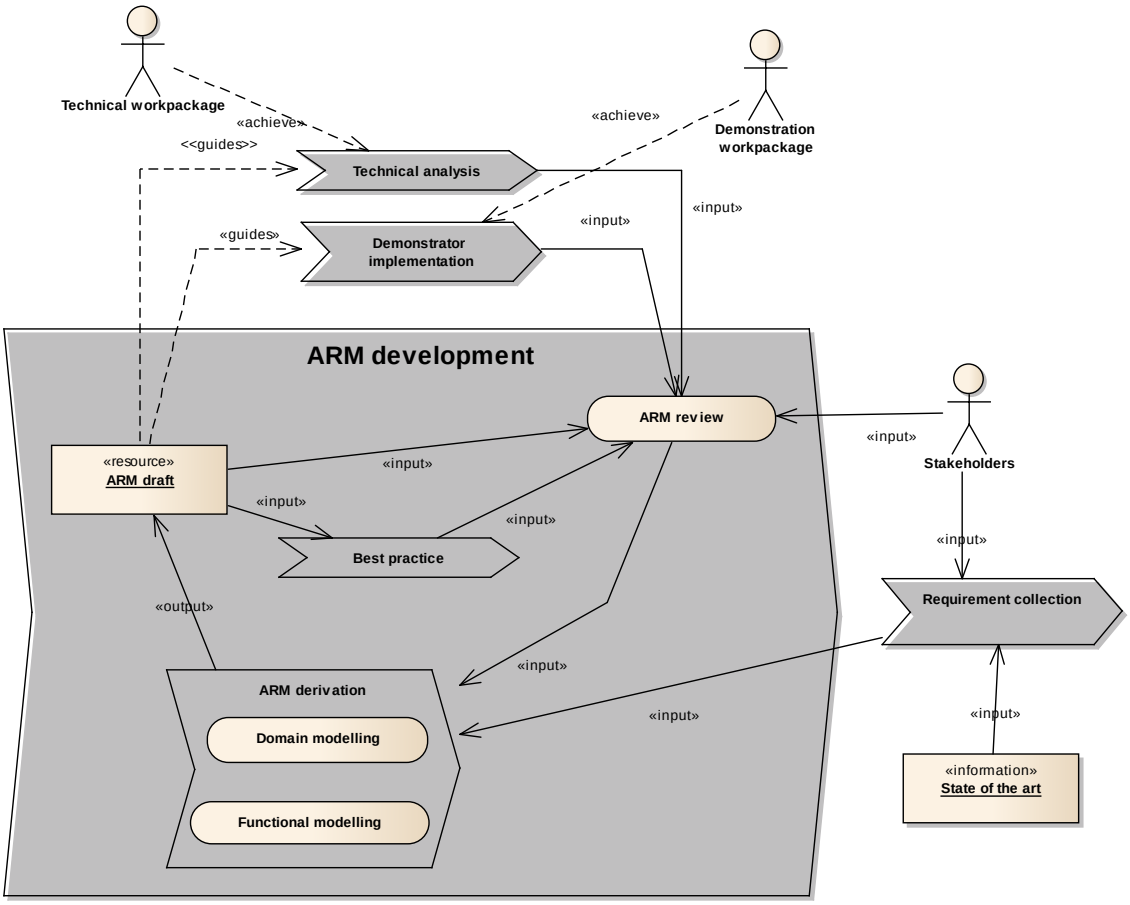

Figure 6: Dynamic view of the IoT-A ARM process.

The ARM development process consists of one main process, which is the ARM derivation. Within the ARM derivation two actions are worth mentioning, viz. the domain modelling, which results in the IoT Reference Model, and the functional modelling, which is the main contributor to the IoT Reference Architecture. This process receives input from the requirement-collection process, which in turn receives input from external stakeholders and the state-of-the-art surveys conducted during the early stages of IoT-A.

For a thorough explanation of the requirement-collection process we direct the reader to Annex B.

The work in the ARM-derivation process is described in our ARM drafts, viz. D1.2, D1.3, and D1.4. The final Version will be D1.5, while the D1.2 contained the initial ARM draft..

The ARM draft guides the set-up of the public use-case demonstrations as well as the work of the technical work packages within IoT-A (“technical analysis”).

The ARM draft is reviewed by the project’s external stakeholders, the demonstration activity, as well as the technical work packages. This review serves as input for a revision of the ARM. In other words, the IoT-A project follows the well-established spiral design and prototyping model [17]. The result from the first iteration of this development cycle is the current document, viz.

D1.3. Before the conclusion of the project two more iterations are planned, resulting in D1.4 and D1.5, respectively.

As discussed in Section 2.2.3, besides the architecture and domain analysis, we also provide the user of the ARM with best practices for deriving use-case- and application-specific architectures (see Figure 5). Besides being of benefit for the user of the ARM, this process has the side benefit of providing valuable feedback to the ARM derivation itself. When devising guidelines for translating the ARM into a specific architecture, potential gaps and inconsistencies are revealed. Also, the best practice exercise deepens our understanding of the IoT domain, and provides additional guidance on what aspects of the ARM need further enhancement. Last, but not least, studying the translation of ARM into specific architectures and vice versa provides a compelling validation of the usefulness of the ARM.

The spiral-model approach inherent in the ARM development process was chosen for the following virtues. First, each iteration increases the stability of the ARM. Second, due to its multi-step nature, the dissemination of the (embryonic) ARM starts early within the project. Thanks to early publication, corrective impulses from peers and external stakeholders are received early on in the development process and can thus positively influence both the applicability of the ARM as well as its acceptance. Third, this approach formalises and coordinates the interaction of the architecture activity within IoT-A with that of the other activities (technical analysis and demonstrator set up), which is expected to enhance the efficacy of this exchange.

2.2.4.2 Generation of architectures

So far we have only described the genesis of the IoT-A ARM, but not how its use for the generation of specific architectures actually works. While Figure 4 explains that the Best Practice also provided in this and the forthcoming documents accomplishes the transformation from the IoT ARM to a concrete architecture, the picture is actually more complicated than that.

When applying the ARM in the design of systems, it is likely that different architectures will result subject to the desired properties of the system. So, while one gets the impression from Figure 4 that the translation of the reference architecture into a concrete architecture is independent of the use case itself this is, in reality, not the case. Rather, Best Practice together with relies on a use-case description and requirements. This fact is reflected in Figure 7. The role of the ARM is to guide the architect through design choices at hand, and to provide her with best practices (sic!) and design patterns for those different choices. The ARM is not operating in a design vacuum but should be applied together with proven design-process practices.

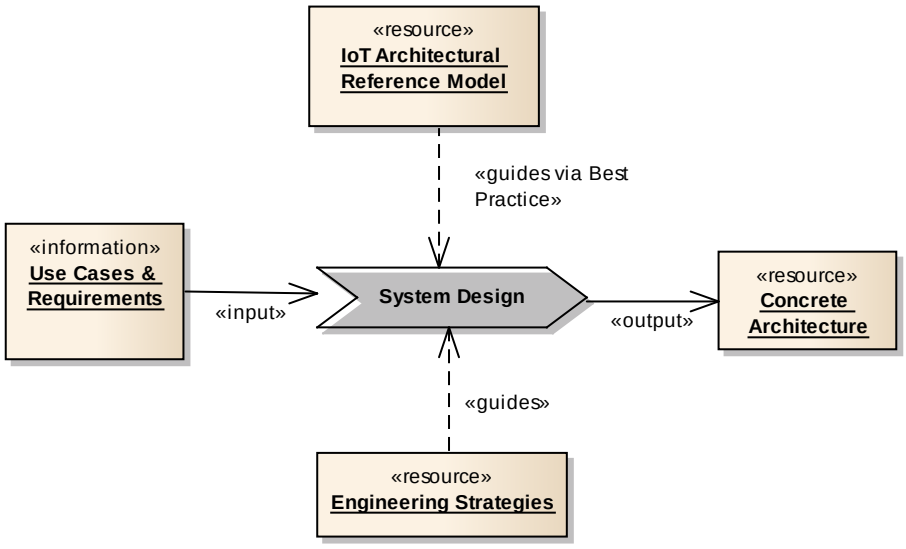

Figure 7: Process for the generation of concrete architectures.

In Section 0 we describe how both the IoT Reference Model as well as the IoT Reference Architecture can be used in this design process. Even though we describe it here in a linear fashion, one needs to keep in mind that in practice it will not always be the case. Depending on the engineering strategies used, some of the steps can be done in parallel or even have to be reiterated due to additional understanding gained during the process, or due to changes in the requirements.

2.2.4.3 Choice of design and development methodology

The choice of a design and development methodology can be understood in two ways: first, a methodology for the ARM development and second, a methodology for the generation of specific concrete architectures. We have so far only provided high-level views of either case. In reality, one needs more guidance, viz. a recipe on how to derive all aspect of the ARM model as well as how to derive the best practices. Simply dissecting them into design steps and processes, as has been done so far, is not enough; one needs to know how to achieve each step.

In the case of the ARM there are, to our knowledge, no standardised approaches for developing such a model. Furthermore, the IoT usage domain is, compared to typical reference-architecture domains, extremely wide and varied, and common denominators are thus rather few and abstract. For examples of reference architectures and models the reader is directed to the literature [18], [6], [1], [19], [14], [16], [15]. This high level of abstraction in terms of the domain to be modelled stands in contrast to input needed for established and standardised methodologies such as, for instance, Aspect-Oriented Programming, Model-Driven Engineering, Pattern-Based Design, and SysML. All these methodologies were designed for very concrete use cases and application scenarios. Unfortunately, this high degree of specificity is even defining their inner workings. In other words, if one applies them to generalised use cases one does often not get generalised models on the abstract level of an ARM, but one does actually not yield anything, since the processes of which said methodologies are constituted, do not work for generalised use cases.

We illustrate the above issue with two examples, Model-Driven Engineering and Pattern-Based Design. In the first case, the methodology is not directly applicable, while, in the second case, the methodology can potentially be generalised for deriving the best-practice transformation shown in Figure 9.

Model-Driven Engineering for the generation of Model-Driven Architectures is standardised by the *Object Management Group* (OMG) [2]. Its application area is the development of software systems. It provides an approach in four steps:

1. Specify a system independently from the platform;
2. Specify platforms;
3. Choose a particular platform for the system;
4. Transform the system specification into that of the particular platform.

The goals behind this approach are portability, interoperability, and reusability through the architectural separation of concerns [20]. So, on the face of it, all this sounds very similar to the goals of our ARM development process.

In Figure 8, the main idea of model-driven architecture is shown. A platform-independent model, viz. an architecture, is to be transformed into a platform-specific model, viz. an implementation. An example for the former is a GUI user interface described in UML, and the latter is an implementation of said interface in a cell-phone model featuring a particular operation system.

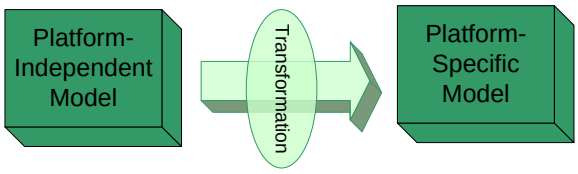

Figure 8: Generalised architecture approach according to the Model-Driven-Architecture methodology, a.k.a. Model-Driven Engineering [2].

While this sounds very much like the Best-Practice transformation depicted in Figure 4, it is not the same. This becomes clearer in Figure 9.

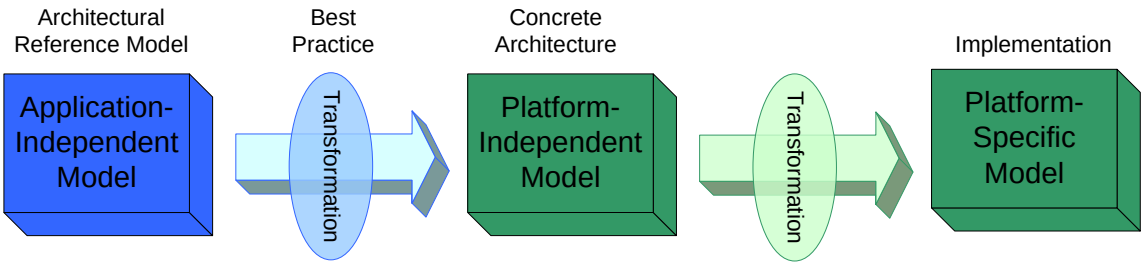

Figure 9: Relation of the Best-Practice-driven derivation of concrete architectures from an architectural reference model and the derivation of implementations from said concrete architecture. This Figure is a composite of Figure 4 and Figure 8.

Figure 9 is pieced together from Figure 4 (ARM) and Figure 8 (Model-Driven Engineering). As one can see, both the ARM and the Model-Driven-Engineering approach are linked to each other through platform-independent models, but they reside on different levels of abstraction. While the general idea of a model transformation, as promoted by MDE, resonates with our ARM approach (see Figure 4), the methodology developed for the derivation of transformations between platform-independent and platform-specific models can, upon a thorough analysis, not be transferred and adapted for the derivation of Best-Practice transformations.

Pattern-Based Design is a technique that reuses repeatable solutions to solve commonly occurring problems. This design was first introduced by Gamma et al. in the context of reusing elements in object-oriented software [21]. In this design method one records how object-oriented designers identify recurring design problems. The corresponding solutions are then documented, and a reuse of the solutions is strived for. Consequently, the design process becomes increasingly flexible, elegant, and, most important, reusable. The solutions are divided into solutions, where “A design pattern identifies the participating classes and instances, their roles and collaborations, and the distribution of responsibilities. Each design pattern focuses on a particular object-oriented design problem or issue” [21]. From this short discussion it becomes clear that (a) Pattern-Based Design was developed for implementation processes, viz. the transformation to the right in Figure 9, and that (b) the only way this method can be applied for the derivation of the best-practice transformation in the same Figure would be by trying to translate the ARM into a particular architecture and to see whether the “book-keeping” approach prescribed by Pattern-Based Design yields valuable insight. At the current stage we do not know whether this is possible and aim at finding out during out ongoing best-practice development, which, among others, encompasses the derivation of a concrete architecture.

Table 1 we summarise how we use ideas lent from standardised architecture methodologies for our work on the higher abstract level of an ARM.

Methodology	**Aspect adopted in our work**
Aspect-Oriented Programming	Delineation of functionalities by aspects. This is embodied in the concept of Functionality Groups (see Section 4.2.2).
Model-Driven Engineering	General concept of transformation from a generic to a more specific model. We use this concept for describing and developing our Best Practice.
Pattern-Based Design	As discussed earlier, pattern-based design is a method for documenting and classifying implementation solutions, and it can therefore not be readily applied to our ARM work. However, it might be valuable for the documentation of solutions devised during the translation of the ARM into a concrete architecture. The goal would be to feed back patterns discerned and lessons learned into the Best-Practice part of this document. We are planning such a translation of D1.5 and will revisit the applicability of Pattern-Based Design during this exercise.
Views and Perspectives	We adopt the concept of views and perspectives for the derivation of the IoT Reference Architecture, viz. we arrange all aspects of our reference architecture according to views and perspectives (see Section 0). The same is done for the unified requirements (see Appendix B).

Table 1: Usage of standardised architecture methodologies for the development of the IoT ARM.

2.2.4.4 Requirements process

The IoT Reference Model by itself does not specify the technical particularities of an IoT system. For example, how are things identified and addressed in an IoT context? Or: how are these things associated with services? Such particularities are addressed in the IoT Reference Architecture. In order to build such a reference architecture, we not only need the IoT Reference Model and the methodology to do so, but also technical requirements that can be used for inferring particularities of the architecture. This is reflected in Figure 5.

In this section we explain how the requirements for the IoT ARM have been inferred. The collection of requirements was done in a three-pronged process:

1. The rich experience and knowledge of the project partners guided the derivation of a minimum-requirement list, which also had a major influence in drafting the IoT Reference Model. The state of the art concerning thing-centric communication and Internet technologies was considered, and a list of internal requirements was inferred. The state of the art was collected in Deliverable D1.1 [10].

2. A group of external IoT stakeholders was established and queried for their use cases and their expectations toward IoT. They were also asked for their objectives, concerns, and business goals. As far as feasible, these overarching aspirations were broken down into requirements.

Usually, such stakeholder aspirations are not made as system requirements, rather as use-case specific goals. Therefore, each stakeholder aspiration was thoroughly analysed, and suitable translations into requirements were sought. Stakeholder aspirations can be rather general (strategic objectives, concerns, or business goals) or they can be very specific, i.e., a stakeholder spells out what kind of functionality or performance she/he needs. An example for the former is the functionality of the IoT systems. For instance, ETSI raised the following concern: *"Today, due to sub-optimal processes, a lot of time and money is wasted. This situation could be improved a lot by tracking all the items/things, providing context data on them at any time and location, allowing for automated evaluation of the collected data and reacting immediately on a dangerous situation to protect against the break down of items." [9]* This addresses the functional view, but it does not clearly address what functionalities are needed in order to meet this aspiration. In our requirement-engineering process (see [9]), we broke this concern down into two distinct functional requirements.

- "The system shall enable centralized or decentralized automated activities (control loops)." (UNI.31)

- "The system shall enable the planning of automated tasks." (UNI.32)

The above example was provided in order to briefly illustrate our requirement process. The unabridged list of requirements is provided in Annex B. The functional view is a recurring item in the list of unified requirements. This view is represented as a block-diagram of the reference architecture, which in itself constitutes a central result of the IoT-A project and an indispensable input for the development of a compliant IoT system. The IoT-A Functional View is addressed in detail in Section 4.2.2.

2.3 Business Scenarios

2.3.1 Rationale and Introduction

As discussed in the previous section about the IoT Reference Architecture dependencies and model influences, business scenarios play an important role in the external validation of the

architectural reference model (ARM). Business scenarios help defining application-specific requirements, i.e. they are one source of input regarding what potential systems and applications need to implement and deliver, if they are to realise certain business scenarios. At the same time, business scenarios help understanding the IoT Reference Model as such, as the domain components described in the reference model are reflected in the respective business scenarios, i.e. the reference model provides a formalised and abstracted model of the entities and their relationships that are brought to life within the different business scenarios.

The primary aim of business scenarios is to provide an external validation of the ARM in economic terms, i.e. business scenarios should demonstrate that concrete systems built utilising ARM-compliant concrete architectures are economically viable and beneficial, so that it makes sense for business stakeholders to develop business scenarios based on IoT-A models and best practices. Ideally, business scenarios should cover a diverse set of relevant application fields in order to demonstrate the broad applicability of IoT-A, especially since one of the primary goals of IoT-A is to develop an IoT Reference Architecture that transforms the isolated island solutions of the “intranets of things” as we know them today into a domain-spanning interoperable infrastructure of IoT platforms. This infrastructure should be viable from an economic point of view and should facilitate novel business opportunities. Within this section of the deliverable, we will only briefly discuss the application fields for which viable business scenarios compliant to IoT-A can be developed, but instead focus only on central application fields in the focus of the project. As IoT-A work package 7 explicitly focuses on health and retail as the primary application fields for which technical demonstrators are to be developed and validated both in terms of being relevant for the application fields as such (the stakeholder group was selected partly based on the industry or domain the stakeholders are experts in) and in terms of technical and economic viability, it makes sense to concentrate on these use cases for a more in-depth analysis of business cases. In that respect, we do provide a certain level of width by briefly outlining various application fields as well as a certain level of depth by exemplarily examining selected business cases.

The narrowed focus of the use cases comes from the fact the stakeholder group of the IoT-A project focuses mostly on selected application fields. Within these fields, stakeholder aspirations can of course be diverse, because of differences in their background and differences in their business views. Nevertheless, there are some common themes in stakeholder aspirations that make us confident that there is some potential for generalizing business scenarios.

- Many stakeholders see IoT as a means of improving their current business. IoT will thus serve various business goals and strategic objectives, such as future-proofness, lowered costs, etc.
- Other stakeholders see IoT as a disruptive technology, which will aid them in creating new applications and thus new business opportunities (selling access to sensor data, etc.).
- In order to achieve a maximum of flexibility of IoT technology and its use, short product-development cycles and a maximum leverage of existing and new solutions to common problems is needed. For that reason, many stakeholders advocate open IoT platforms and frameworks. The underlying business goal for this advocacy is to lower costs in product development. Strategic objectives are to enhance product interoperability and to shorten the development cycles. The latter is important for responding to customers’ emerging needs in an agile manner.
- Since active supervision of IoT interactions is even more elusive than monitoring today’s Internet traffic, security and privacy have, as expected, been identified as a core topic. Privacy is strongly related to the overall acceptance of IoT. If individuals and other users cannot experience a sufficient level of privacy when utilising IoT technology, this

will critically challenge the acceptance of this novel technology. Security equates of course not only to privacy, but also to the protection of the IoT against interferences, such as service attacks, Trojans, viruses, etc.

For our business case evaluation within the context of this deliverable, we will focus on a single use case from work package 7 based on certain common stakeholder aspirations that comes from the retail domain. Correspondingly, within the next deliverable D1.4 we will provide a similar examination for a central use case from the health domain, so that both of the central application fields of IoT-A are reflected appropriately.

Within the next two sections we will outline both the relevant application fields, building upon the work performed in our "sister project" IoT-i, as well as a business case evaluation of a retail use case that is based upon a business case methodology developed in a preceding IoT project called SemProM (Semantic Product Memories, http://www.semprom.de/semprom_engl/).

2.3.2 Fields of Application

In order to maximise the impact of our architectural reference model, we have to identify those scenarios where IoT technologies have a special relevance, taking into account that these scenarios frequently share the same applications, sensors, stakeholders and, of course, users. We will base this identification on scenarios that have been kindly provided by the IoT-I CSA [40].

Field of application	**Impacts**
Transportation/ Logistics	In transport logistics, IoT improves not only material flow systems, but also global positioning and auto identification of freights. Additionally, it increases energy efficiency and thus decreases energy consumption. In conclusion, IoT is expected to bring profound changes to the global supply chain via intelligent cargo movement. This will be achieved by means of continuous synchronisation of supply-chain information, and seamless real-time tracking and tracing of objects. It will provide the supply chain a transparent, visible and controllable nature, enabling intelligent communication between people and cargo.
Smart home	Future smart homes will be conscious about what happens inside a building, mainly impacting three aspects: resource usage (water conservation and energy consumption), security, and comfort. The goal with all this is to achieve better levels of comfort while cutting overall expenditure. Moreover, smart homes also address security issues by means of complex security systems to detect theft, fire or unauthorized entries. The stakeholders involved in this scenario constitute a very heterogeneous group. There are different actors that will cooperate in the user's home, such as Internet companies, device manufacturers, telecommunications operators, media-service providers, security companies, electric-utility companies, etc.

Field of application	Impacts
Smart city	While the term smart city is still a fuzzy concept, there is a general agreement that it is an urban area which creates sustainable development and high quality of life. Giffinger et al.'s model elucidates the characteristics of a smart city, encompassing economy, people, governance, mobility, environment and living [22]. Outperforming in these key areas can be done through strong human or social capital and/or ICT infrastructure. For the latter, a first business analysis concludes that several sectors/industries will benefit from more digitalised and intelligent cities (examples for a city of 1 million people [23]): • Smart metering, 600.000 meters, US $ 120 million opportunity • Infrastructure for charging electric vehicles, 45.000 electric vehicles, US $ 225 million opportunity • Remote patient monitoring (diabetes), 70.000 people, US $ 14 million opportunity • Smart retail, 4.000 stores, US $ 200 million opportunity • Smart-bank branches, 3.200 PTMs, US $ 160 million opportunity
Smart factory	Companies will be able to track all their products by means of RFID tags by means of a global supply chain; as a consequence, companies will reduce their OPEX and improve their productivity due to a tighter integration with ERP and other systems. Generally, IoT will provide automatic procedures that imply a drastic transformation of employees towards higher level activities, as workers will be replaced by bar-code scanners, readers, sensors and actuators, and in the end by complex robots, as efficient as a human. Without any doubt, these technologies will bring opportunities for white-collar workers and a big number of technicians will be necessary to program and repair these machines. This is synonymous to a transfer to maintenance jobs, but it also constitutes a new challenge for providing all blue-collar workers with an opportunity to move toward these types of jobs and to avoid unemployment.
Retail	IoT realises both customer needs and business needs. Price comparison of a product; or looking for other products of the same quality at lower prices, or with shop promotions gives not only information to customers but also to shops and business. Having this information in real time helps enterprises to improve their business and to satisfy customer needs. Obviously, big retail chains will take advantage of their dominant position in order to enforce the future IoT retail market, as it happened with RFID adoption, which was enforced by WalMart in 2004 [24]. Particularly, companies with controlling positions, such as WalMart, Carrefour, Metro AG, etc. are able to push the adoption of IoT technology due to their sizable market shares.

Field of application	Impacts
e-Health	Controlling and preventing is one of the main goals of future health care. Already today, people have the possibility of being remotely tracked and monitored by specialists. Tracing peoples' health history is another aspect that makes IoT-Assisted eHealth very versatile. Business applications could offer the possibility of medical service not only to patients but also to specialists, who need information to proceed in their medical evaluation. In this domain, IoT makes human interaction much more efficient because it not only permits localization, but also tracking and monitoring of patients. The most important stakeholders in this scenario will be public and private hospitals and institutes such as, e.g., the Institute of Applied eHealth at Edinburgh Napier University, which partook in the first stakeholder session of IoT-A. It is worth mentioning that telecommunications operators are quite active in e-health (for instance, O2 UK).
Energy	From the aforementioned application we infer that environment has many overlaps with other scenarios, such as smart home and smart city. One key issue in these scenarios is to detect means that help to save energy. We are basically referring to what is known as Smart Grid. Concerning this application area one needs to highlight initiatives that imply a more distributed energy production, since many houses have a solar panel today. As a vital part, smart metering is considered as a pre-condition for enabling intelligent monitoring, control, and communication in grid applications. The use of IoT platforms in Smart Metering will provide the following benefits: • An efficient network of smart meters allows for faster outage detection and restoration of service. Such capabilities redound to the benefit of customers • Provides customers with greater control over their energy or water consumption, providing them more choices for managing their bills. • IoT deployment of smart meters is expected to reduce the need to build power plants. Building power plants that are necessary only for occasional peak demand is very expensive. A more economical approach is to enable customers to reduce their demand through time-based rates or other incentive programs, or to use automatic recording of consumptions to temporarily turn off devices which are not in use. Finally, combining the analysis of supply and demand, energy enterprises will be able to realize a more efficient demand shaping. They will not just give incentives to consumers, but actually turn off devices that are not needed (like the freezer for 20 minutes). Also most of this needs to happen automatically. Here we again face a heterogeneous scenario, in which diverse stakeholders are involved. Main actors are of course energy utilities, but also public entities will be important players.

The application fields outlined above serve as a foundation for describing the business processes, applications, context in which the applications enable the technology, actors (e.g.,

people or physical entities) and computing components which are involved in the scenario and the desired outcome of proper execution.

In order to describe a well-defined business model, it is necessary to define what needs to be done in the business, which are the metrics for success, which are the problems that must be solved and the plans that solve these problems. Knowing which part of the problem is possible to solve, how much time is needed, and which part cannot be solved is an important step that we must take into account when we develop concrete business cases for some of the application fields discussed above.

As we can only go into the details of one business case in the context of this document, we will pick a use case from the application field of retail, as this is a central application field for the project, and apply an appropriate business case methodology to it. This methodology is outlined in the next section. The use of a methodology instead of merely calculating “some kind of business case” enables us to perform comparisons between different application fields, for instance when we consider e-Health under an economic perspective within the context of the forthcoming deliverable D1.4.

2.3.3 Business Case Methodology

As the scenarios and use cases developed in work package 7 demonstrate, there is a huge potential for realising IoT applications in different application fields that are based on architectural concepts of IoT-A and potentially bring novel business opportunities, for instance when sensor technology contributes to changing distribution models for perishable goods, so that e.g. fruits or vegetables can still be sold to the consumer, before their quality deteriorates and the goods are wasted. However, to make such scenarios possible, large investments are needed in e.g., hardware, software, installation, configuration, maintenance, business process reengineering and training of personnel. To justify such investments, a Business Case (BC) is usually developed, describing the benefits, costs and risks of each investment alternative.

BCs commonly appear as spread-sheets, often accompanied by presentations or explanatory documents. They may be presented by the project leader (BC ‘owner’ or ‘champion’) to senior management, which is responsible for prioritizing BCs and making investment decisions. This way, the BC can be used to decide about investment before project execution (‘ex-ante’), to evaluate progress during project execution, and to determine to what extent the proposed value of the investment has been realized after project execution (‘ex-post’). Naturally, the development of BCs is a complex task. First, collecting, transforming and aggregating the required information demands interdisciplinary teamwork and expertise in a wide range of fields such as business strategy, business operations (‘work practice’), information technology, accounting and project management. Second, BCs are based on assumptions concerning the future development of certain variables. Predicting those variables requires accurate data and reliable analysis methods. Third, BCs are subject to a constantly changing business environment, requiring an agile BC development process to adapt to these changes.

Within the context of IoT-A, BCs should be based on a generic BC process to allow for their development, use and improvement across different application fields. We therefore base the BC process on a framework being developed in the IoT project SemProM that proposes a BC framework which is based on a generic BC process, consisting of six steps: Scope, Processes, Criteria, Methods, Results, Conclusion. During this process, domain-specific components consisting of criteria and methods may be reused.

The BC framework provides a set of spread-sheets in Microsoft Excel that accompany the process proposed. In the following section, we apply this framework to two of the primary retail use cases developed in work package 7, namely the NFC Based Shopping Assistant (retail scene 5) in combination with the sensor-based quality control (retail scene 7).

2.3.4 Retail Business Case

In this section we exemplify a business case from the retail domain that shows the beneficial effects of the IoT-based use cases investigated in work package 7 of the IoT-A project. The complete details can be found in D7.1 and D7.2 and the prototypical implementation and evaluation in the project's retail living lab is expected for the end of 2012. The use case shows how IoT technologies like sensor technologies built into consumer electronic devices and NFC tags coupled with a concrete architecture derived from the Reference Architecture can provide useful meta-information to the customer to enhance the overall shopping experience and at the same time significantly reduce the costs for consulting that sales personnel in the retail stores need to conduct today, as there are no such systems in widespread use. The use case demonstrates what a direct human-to-machine interaction enabled by IoT-A would be like. As such, the overlap between the existing Internet and the future Internet of Things is shown. From a business perspective, the use case is primarily interesting, because the NFC-based product information has the potential to reduce the consultation time of the sales personnel in the store.

In order to make the business case somewhat more complex, we also integrate the core functionality of use case scene 7 (sensor based quality control) of D7.2 into the case and assume that both scenes are interconnected, because they are based on a common architecture, namely a concrete architecture based on the IoT-A Reference Architecture.

This sensor based quality control scene shows how sensors monitor perishable goods in a store. Depending on the luminance, humidity, and temperature of the environment, the estimated future quality of the perishable products is determined and prices are reduced, even before a perceivable degradation of quality occurs. By applying this sensor based quality control and combining it with dynamic pricing, it is ensured that the goods are sold before quality degradation is likely to occur. From a business and industry perspective, the scene demonstrates two important retail related concepts: dynamic pricing and quality control of perishable goods. Dynamic pricing as a real-time tool for price optimization strategies has always been crucial for profit maximization. In contrast to the state of the art, dynamic pricing in the featured use case is not performed based on static information such as best before end dates in the transaction data of the backend ERP system, but it is based on real time IoT data gathered from a sensor infrastructure. As about 20% of perishable goods never reach the consumer, but are disposed of before, either in the store or in the supply chain, the utilization of IoT sensors is also an interesting concept to implement quality control of perishables and thus reduce waste and increase profits at the same time.

As we have stated before, we assume that both the self-contained NFC-based product information and the sensor based quality control are based on the same technical system realised in accordance with the IoT-A ARM. Therefore, we calculate their anticipated effects in a combined business case. The actual Excel sheets are available on the IoT-A website at http://www.IoT-A.eu/public/public-documents accompanying the deliverable, but in the following tables we already provide the respective criteria, on which the calculations are based, as well as instantiations of these criteria calculated for cases, when the IoT-A -based use cases are realised and when they are not realised (= the baseline).

In our calculations we base our BC on an example case for German Retailers trading fast moving consumer goods in a higher market segment. The following two tables illustrate some of the respective parameters used.

BC_IoT-A.xlsx - Microsoft Excel

Criteria	Method	Unit	Goal	Impact	Business Process	Reference
1 2 3 4 5 6						
Benefits			max.			
Sales	MS = sum(AS) * reference time frame	€	max.		06. Sales	Lipschitz(2006): Retail Key Performance Indicators (KPI) , http:
Sales per hour	HS = sum(all items sold) / sum(hours within tfloat		max.		06.04 Sales Processing	Lipschitz(2006): Retail Key Performance Indicators (KPI) , http:
Consulting time	CT = customers * store visits * time per custor time		min.		06.04 Sales Processing	
Self-contained product information			max.		06.04 Sales Processing	
Average sale	AS = total price of items sold per sale / receip	€	max.		06.04 Sales Processing	Lipschitz(2006): Retail Key Performance Indicators (KPI) , http:
Price optimization	price = f(optimization)	€	max.		06.03 Offering Design	Dynamic Pricing in the Presence of Inventory Considerations:
Dynamic quality based price optimizatic	price = f(quality)	€	max.	Sensor-Based Quality Control a	06.03 Offering Design	Dynamic Pricing in the Presence of Inventory Considerations:
Transparency of origin	price = sum(prices of products with transpare	€	max.		06.03 Offering Design	
Items per sale	IS = sum(all items sold per reference time fra	int	max.		06.04 Sales Processing	Lipschitz(2006): Retail Key Performance Indicators (KPI) , http:
Conversion rate	CR = (Number of visitors / number of receipts)	%	max.		06.01 Customer Management	Lipschitz(2006): Retail Key Performance Indicators (KPI) , http:
Attraction through product information	AT = (Number of visitors accessing additional	%	max.	The additional information tha	12.03 Store Design	Senecal, S. & Nantel, J., The Influence of Online Product Reco
Supply chain optimization	SCO= CCI + sum(other wasted products) / sum	%	max.		05.04. Transport Planning	Dada, A. and Thiesse, F. 2008. Sensor applications in the sup
Cool chain integrity	CCI = QB / sum(all perishable products)	%	max.		05.04. Transport Planning	
Quality based issuing of perishables	QB = sum(wasted perishable products)	int	min.	The continuous monitoring of p	05.06 Warehouse Control	
Assortment selection	AS = sum(CCP) + sum(other products)	int	max.		06.02 Assortment Selection	Shelby H. McIntyre, Christopher M. Miller, The selection and
Conscious consumer products	CCP = TO + sum(other conscious consumer pro	int	max.		06.02 Assortment Selection	
Transparency of origin	TO = sum(products with transpareny of origin	int	max.		06.02 Assortment Selection	
Costs						
Store operation costs	SOC = IPM + EC					
Instore peripheral management	IPM = sum(software infrastructure costs in-st	€	min.	A Real World Integration Platfo	12.03 Store Design	
Employee costs	EC = sum(salary of each person)	€	min.	For example, the Shopping Assi	11.05 Organisational Managem	Resatsch, F.; Sandner, U.; Leimeister, J. M. & Krcmar, H. Do Poi
Acquisition of trading goods	ATG = sum(all below)				03. Acquisition of Trading Goods	
Production	PRO = production costs	€	max.		04. Production	
Raw materials and supplies	RMC = Costs of Material	€	max.		04. Production	
Procurement	PRC = Total costs for procurement	€	max.		03. Acquisition of Trading Goods	
Goods receipt	GR = Costs per Good received	€	min.		05.01 Goods Received	
Waste costs	WC = Costs of Waste	€	max.		05.02 Inventory Management	
Inventory costs	INV = Current Inventory * Item Costs	€	max.		05.02 Inventory Management	
Risks						
Dependability on technical infrastructure	DEP = dependability	%	min.	IoT infrastructures introduce a	significant amount of technical components, both on the products and on the backend. Core t	
Sales personnel competencies	SPC = necessity of being a domain expert	%	max.	As NFC based shopping assista	11.05 Organisational Management	
Digital divide causing loss of consumer base	DD = digital divide (DDH + DDL)	%	min.	With consumer-facing compone	06.02 Assortment Selection	Coroama et al (2003): Living in a smart environment. The soci
high education population	DDH = digital divide high education	%			06.02 Assortment Selection	Coroama et al (2003): Living in a smart environment. The soci
low education population	DDL = digital divide low education	%			06.02 Assortment Selection	Coroama et al (2003): Living in a smart environment. The soci
Technical faults and errors	FE = faults and errors	%	min.	Any technical infrastructure imp	12.03 Store Design	
Process re-engineering threats						
Handling and storage procedures	HS = handling and storage	%	min.	Any RFID based system is heav	05.04. Transport Planning	Finkenzeller (1999): The RFID handbook
Sales, return and recall related information governe	IG = information governance	%	min.	Legal restrictions impose signi	06.01 Customer Management	Coroama et al (2003): Living in a smart environment. The soci

Scope | Processes | Criteria | Methods & Values | Results

Table 2: Criteria for the Retail Business Case

BC_IoT-A.xlsx - Microsoft Excel

	A	B	C	D	E	F	G	H	I	J	K	L
1	**Benefits > Sales per hour > Consulting time > Self-contained product information via NFC based Shopping assistant**											
2	Description: Sales per Hour is a traditional KPI to measure salesperson or store performance											
3	Reference: Lipschitz(2006): Retail Key Performance Indicators (KPI) , http://www.hillsorient.com/articles/2006/07/121.html											
4												
5	Example case for German Retailers trading FMCGS: Consulting time for organic products purchases											
6												
7												
8												
9	**Instantiation with Traditional Consulting by Sales Personnel**											
10	Parameters	2010	2011	2012	2013	Metric	Reliability	Source				
11	RR: Relevant retailers	31174	30550	29939	29340	abs. number	fact	Statistisisches Bundesamt, consilidation is likely c				
12	HS: Households per Store	300	305	310	315	abs. number	accurate estimate	Estimated from http://salzburg.orf.at/stories/439				
13	HO: Households	40'200'000	40'119'000	40'039'000	39'959'000	abs. number	fact	Statistisisches Bundesamt, decrease in sync with s				
14	CO: Consumers	70'000'000	69'860'000	69'720'000	69'580'000	abs. number	rough estimate	Active population is decreasing 0.2% each year in				
15	IOC: 'Intensive' Organic Consumers	0.02	0.02	0.02	0.02	%	rough estimate	Bruhn, Maike (2002) Warum kaufen Verbraucher B				
16	CR: Consumers requiring Consulting	0.1	0.1	0.1	0.1	%	guess	Estimation				
17	VW: Retail Store Visits per week	1.5	1.5	1.5	1.5	abs. number	accurate estimate	HJ Becker (2005) Einkaufszentren in Berlin http://v				
18	MWT: Mean Waiting Time for Consulting	110	110	110	110	seconds	accurate estimate	Jungbauer et al (2005) Does Clothing Make the Ma				
19	MCT: Mean Consulting time per customer	240	240	240	240	seconds	accurate estimate	Estimate derived from pharmacy domain http://w				
20												
21	Derived Measures											
22	RC: Relevant Consumer store visits per week (CO*IOC*VW)	2'100'000	2'095'800	2'091'600	2'087'400	abs. number						
23	CW: Consulting time per week (RC * CR * (MWT + MCT))	20'417	20'376	20'335	20'294	hours						
24	RT: Weekly Consulting time per retailer (CW / HO / HS / RR)	**4'750**	**4'732**	**4'714**	**4'694**	hours						
25												
26												
27	**Instantiation with IoT-A assisted Self-Contained Product Information**											
28	Parameters	2010	2011	2012	2013	Metric	Reliability	Source				
29	RR: Relevant retailers	31174	30550	29939	29340	abs. number	fact	Statistisisches Bundesamt, consilidation is likely c				
30	HS: Households per Store	300	305	310	315	abs. number	accurate estimate	Estimated from http://salzburg.orf.at/stories/439				
31	HO: Households	40'200'000	40'119'000	40'039'000	39'959'000	abs. number	fact	Statistisisches Bundesamt, decrease in sync with s				
32	CO: Consumers	70'000'000	69'860'000	69'720'000	69'580'000	abs. number	rough estimate	Active population is decreasing 0.2% each year in				
33	IOC: 'Intensive' Organic Consumers	0.02	0.02	0.02	0.02	%	rough estimate	Bruhn, Maike (2002) Warum kaufen Verbraucher B				
34	CR: Consumers requiring Consulting	0.1	0.1	0.1	0.1	%	guess	Estimation				

Scope | Processes | Criteria | **Methods & Values** | Results

Table 3: Sample Instantiations for the Retail Business Case

The core result of the business case calculation for the retail domain is that, apart from the reduced waste of perishables due to the sensor based quality control, the consulting time of sales personnel being reduced significantly, in our case about 30%, so that IoT-based scenarios indeed appear to have a significant business impact. The business case as we have calculated it for a retailer from the *Fast Moving Consumer Goods* (FMCG) field does not yet take into account the savings and benefits of software systems that are compliant with the IoT Reference Architecture and that follow the best practices and design choices laid out by the IoT-A project. While we envision the best practices to be a strong and central contribution of the project, it is currently difficult to calculate its economic impact. By the next deliverable D1.4 we expect to have more in-depth implementation design choices and best practices at hand, so that these effects can be taken into account for the next business case. Apart from the best practices, we also believe that substantial economic benefits can emerge from the modular and component-based approach to building IoT systems that are compliant with the IoT-A Reference Architecture. These additional business effects will also be taken into account for future business cases.

While we can already state that from an economic perspective the IoT-A approach simply makes sense, it is also important to note that solid business scenarios are only a precondition to the application of the ARM. In order to implement Internet of Things use cases based on the IoT-A Reference Architecture, the project aims at providing much more than just the Reference Architecture itself and an economic validation: The business cases are just one building block towards a fully featured “Cook Book” with information on various aspects concerning the implementation of IoT systems. It will provide best practices for the various modules that the ARM comprises and will discuss design choices that academics and practitioners alike will be faced with when implementing concrete systems based on IoT-A.

3 Reference model

3.1 Interaction of all sub-models

The reference model aims at establishing a common grounding for IoT architectures and IoT systems. It consists of the sub-models shown Figure 10 that will be explained in the following.

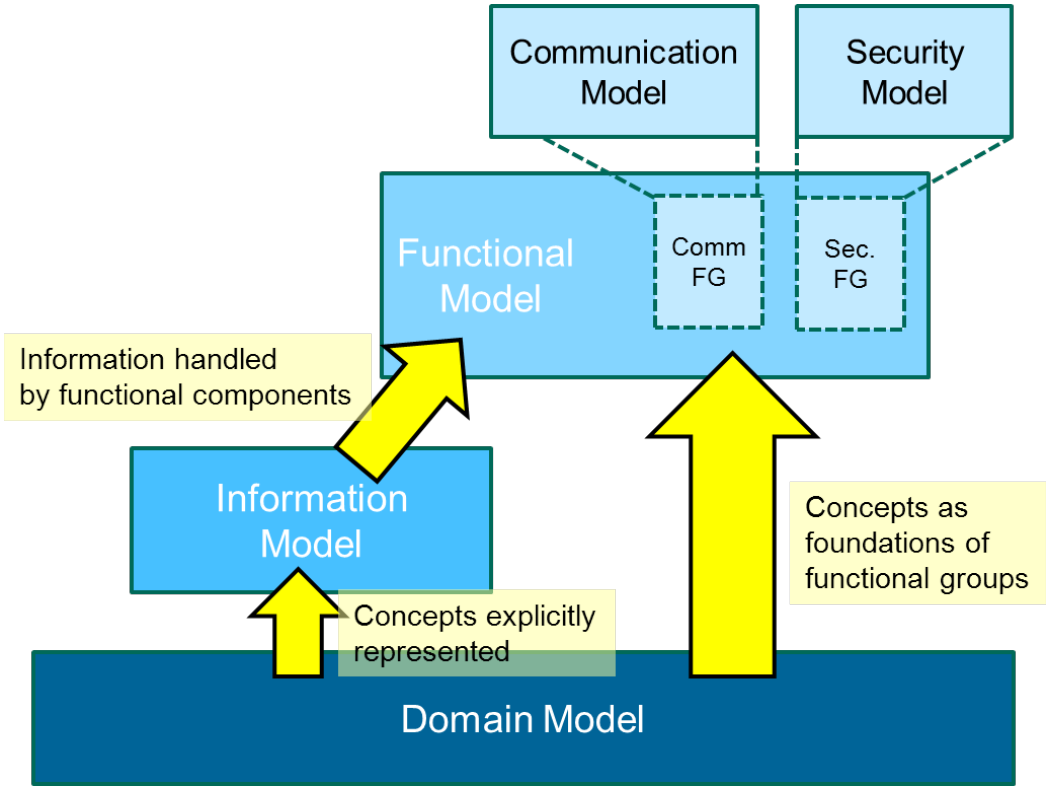

Figure 10: Interaction of all sub-models

The foundation of the Reference Model is the Domain Model that introduces the main concepts of the Internet of Things and the relations between these concepts. The abstraction level has been chosen in such a way that the concepts are independent of specific technologies and are invariant, i.e. are not expected to change over time.

Based on the Domain Model, the Information Model has been developed that defines the structure (e.g. relations, attributes) of all the information (data) that is handled in an IoT system on a conceptual level. So the information pertaining to those concepts of the Domain Model is modelled, which is explicitly gathered, stored and processed in an IoT system.

The Functional Model identifies groups of functionalities that are in most cases centred around key concepts of the Domain Model. A number of these *Functional Groups* (FG) build one on top of the other, following the relations identified in the Domain Model. The Functional Group then provides the functionalities for interacting with the instances of these concepts or managing the information related to the concepts. The functionalities managing information use the Information Model as the basis for structuring their information.

A key functionality in any distributed computer system is the communication between the different components. Specific to an IoT system is the heterogeneity of communication technologies. The Communication Model introduces concepts for handling the complexity of communication in heterogeneous IoT environments. Communication also constitutes one Functionality Group in the Functional Model.

Finally, security and privacy are of paramount importance in typical IoT environments. Therefore, the relevant functionalities and their interdependencies and interactions are introduced in the Security Model. As in the case of communication, security constitutes one Functionality Group in the Functional Model.

3.2 Domain Model

3.2.1 Definition and Purpose

The IoT-A project defines a domain model as a description of concepts belonging to a particular area of interest. The domain model also defines basic attributes of these objects, such as name and identifier. Furthermore, the domain model defines relationships between objects, for instance “instruments produce data sets”. Domain models also help to facilitate correlative use and exchange of data between domains [18]. Besides this official definition, and looking at our interpretation of it, our domain model also provides a common lexicon and taxonomy [1]. The terminology definitions of IoT-A are provided online as well as in Annex C.

The domain model is an important part of any reference model because it includes a definition of the main abstract concepts (abstractions), their responsibilities, and their relationships. Regarding the level of detail, the domain model should separate out what does not vary much from what does [25]. For example, in the IoT domain, the device concept will likely stay around, while the types of devices used will change over time or vary depending on the application context. For instance, there are many technologies to identify objects –RFID, bar codes, image recognition etc. But which of these will still be in use 20 years from now? And which is the best-suited technology for a particular application? For these and related reasons, the domain model does not include particular technologies, but rather abstractions thereof.

The main purpose of a domain model is to generate a common understanding of the target domain in question. In our case the question is what does the IoT define?

Such a common understanding is important, not just project-internally, but also for the scientific discourse. Only with a common understanding of the main concepts it becomes possible to argue about architectural solutions and to evaluate them. As has been pointed out in literature, the IoT domain suffers already from an inconsistent usage and understanding of the meaning of many central terms [26].

3.2.2 Main abstractions and relationships

3.2.2.1 Interpreting the model diagram

This section describes the IoT domain model used in the IoT-A project. It was developed refining and extending two models found in the literature [26], [27]. It is meant to capture the main concepts and the relationships that are relevant for stakeholders concerned with the IoT. For better understanding, this is followed in Section 3.2.3 with more detailed explanations. Guidelines and best practices on how to use the domain model will be given in Section 5.2.

UML is used to graphically illustrate the model. Generalization is used to depict an “is-a” relationship and should not be misinterpreted as sub-classing. Only the most important specialisations are shown, others are possible however. For example, not every *Device* can be

characterized as either a *Tag*, a *Sensor*, or an *Actuator*. The specialisations are however generally disjoint, if not noted otherwise[1].

Concepts depicting hardware are shown in blue, software in green, animate beings in yellow, and concepts that fit into either multiple or no categories in brown.

3.2.2.2 The core IoT Domain Model

The generic IoT scenario can be identified with that of a generic *User* that needs to interact with a (possibly remote) *Physical Entity* (PE) of the physical world (see Figure 11). In this short description we have already introduced the two key actors of the IoT. The *User* is a human person or some kind of a *Digital Artefact* (e.g., a *Service*, an application, or a software agent) that has an interest in interacting with a *Physical Entity*.

In the physical environment, interactions can happen directly (e.g., by moving a pallet from A to B manually). In the IoT though, we want to be able to interact indirectly or *mediated*, i.e., by calling a *service* that will either give information about the *Physical Entity* or actuate on it. When a *Human User* is accessing a service, he does so through a service client, i.e., some software with an accessible user interface. For simplicity reasons, the service client is not shown in Figure 12. For the scope of the domain model, the interaction is usually characterized by a goal of the user. The *Physical Entity* is a discrete, identifiable part of the physical environment which is of interest to the *user* for the completion of his goal. *Physical Entities* can be almost any object or environment; from humans or animals to cars; from store or logistic chain items to computers; from electronic appliances to closed or open environments.

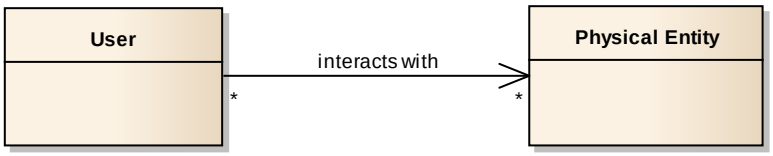

Figure 11: Basic abstraction of an IoT interaction.

Physical Entities are represented in the digital world via a *Virtual Entity*. This term is also referred to as „virtual counterpart“ in the literature [28], but using the same root term „entity“ in both concepts clearer shows the relationship of these concepts. There are many kinds of digital representations of *Physical Entities*: 3D models, avatars, data-base entries, objects (or instances of a class in an object-oriented programming language), and even a social-network account could be viewed as such a representation. However, in the IoT context, *Virtual Entities* have two fundamental properties:

- They are *Digital Artefacts*. Virtual Entities are associated to a single *Physical Entity* that they represent. While generally there is only one *Physical Entity* for each *Virtual Entity*, it is possible that the same *Physical Entity* can be associated to several *Virtual Entities*, e.g., a different representation per application domain or per IT system. Each *Virtual Entity* must have one and only one ID that identifies it univocally. *Virtual Entities* are

[1] The one exception is the specialisations of the *Digital Artefact*: All *Digital Artefacts* can be classified as either *Active* or *Passive Digital Artefacts*. *Virtual Entities* are also *Digital Artefacts* that can be either active or passive.

Digital Artefacts that can be classified as either active or passive. *Active Digital Artefacts* are running software applications, agents or *Services* that may access other *services* or *Resources*. *Passive Digital Artefacts* are passive software elements such as data-base entries or other digital representations of the *Physical Entity*.

- Ideally, *Virtual Entities* are synchronised representations of a given set of aspects (or properties) of the *Physical Entity*. This means that relevant digital parameters representing the characteristics of the *Physical Entity* can be updated upon any change of the former. In the same way, changes that affect the *Virtual Entity* could manifest themselves in the *Physical Entity.*

At this point it should be noted that while Figure 11 at first sight seems to suggest only a person interacting with some physical objects, it also covers interaction between two machines: in this case, the controlling software of the first machine is an *Active Digital Artefact* and thus a *User*, and the second machine – or a *Device* in the terms of the domain model – can be modelled as a *Physical Entity*. See Section 5.2.1.5 for more details on how to model machine-to-machine (M2M) interactions.

We introduce the concept of an *Augmented Entity* as 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, and also in order to have a name for the composition. 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.

The relationship between *Augmented*, *Physical* and *Virtual Entities* is shown in Figure 12, together with other terms and concepts that will be introduced in the remainder of this section.

The relation between *Virtual* and *Physical Entity* is usually achieved by embedding into, by attaching to, or by simply placing in close vicinity of the *Physical Entity* one or more ICT *Devices* that provide the technological interface for interacting with or gaining information about the *Physical Entity*. By so doing the *Device* actually enhances the *Physical Entity* and allows the latter to be part of the digital world. This can be achieved by using *Devices* of the same class, as in the case of body-area network nodes, or by using *Devices* of different classes, as in the case of an RFID tag and reader. A *Device* thus mediates the interactions between *Physical Entities* (that have no projections in the digital world) and *Virtual Entities* (which have no projections in the physical world), generating a paired couple that can be seen as an extension of either one. *Devices* are thus technical artefacts for bridging the real world of *Physical Entities* with the digital world of the Internet. This is done by providing monitoring, sensing, actuation, computation, storage and processing capabilities. It is noteworthy that a *Device* is also a *Physical Entity* and can be regarded as such, especially in the context of certain applications. An example for such an application is device management, whose main concern is the devices themselves and not the objects or environments that these devices monitor.

From an IoT point of view, the following three basic types of *Devices* are of interest:

- *Sensors* provide information about the *Physical Entity* they monitor. Information in this context ranges from the identity of the *Physical Entity* to measures of the physical state of the *Physical Entity*. Like other *Devices*, they can be attached or otherwise embedded in the physical structure of the *Physical Entity,* or be placed in the environment and indirectly monitor entities. An example for the latter is a face-recognition enabled camera. Information from *sensors* can be recorded for later retrieval (e.g., in a storage type of Resource, see 3.2.3.2).

- *Tags* are used to identify *Physical Entities* to which they are usually attached to. The identification process is called “reading” and it is carried out by specific *sensor* Devices, which are usually called readers. The sole purpose of *tags* is to facilitate and increase

the accuracy of the identification process. This process can be optical, as in the case of barcodes and QR code, or it can be RF-based, as in the case of microwave car-plate recognition systems and RFID. The actual physics of the process as well as the many types of tags are however irrelevant for the domain model as these technologies vary and change over time. These are important however when selecting the right technology when implementing a concrete system.

- *Actuators* can modify the physical state of a *Physical Entity*, like changing the state (translate, rotate, stir, inflate, switch on/off,...) of simple *Physical Entities* or activating/deactivating functionalities of more complex ones.

Notice though, that *Devices* can be an aggregation of several Devices of different types. For instance, what we call a sensor node often contains both *sensors* (e.g., movement sensing) as well as *actuators* (e.g., wheel engines). In some cases, *Virtual Entities* that are related to large *Physical Entities* might need to rely on several, possibly heterogeneous, *Resources* and *Devices* in order to provide a meaningful representation of the *Physical Entity*.

Resources are software components that provide information about or enable the actuation on *Physical Entities*. *Resources* typically have native interfaces. There is a distinction between *On-Device Resources* and *Network Resources*. *On-Device Resources* are hosted on *Devices,* viz. software that is deployed locally on the *Device* that is attached to the *Physical Entity*. They include executable code for accessing, processing, and storing sensor information, as well as code for controlling *actuators*. On the other hand, *Network Resources* are Resources available somewhere in the network, e.g., back-end or cloud-based data bases. A *Virtual Entity* can also be related to *Resources* that enable interaction with the *Physical Entity* that the *Virtual Entity* represents.

In contrast to heterogeneous *Resources* – implementations of which can be highly dependent on the underlying hardware of the *Device* –, a *Service* provides a well-defined and standardised interface, offering all necessary functionalities for interacting with *Physical Entities* and related processes. Interaction with the service is done via the network. On the lowest level – the one interfacing with the *Resource* and closer to the actual device hardware –, *services* expose the functionality of a *Device* through its hosted *Resources*. Other *services* may invoke such low-level services for providing higher-level functionalities, for instance executing an activity of a specified business process.

Since it is the service that makes a Resource accessible, the above-mentioned relations between Resources and Virtual Entities are modelled as associations between Virtual Entities and services. For each Virtual Entity there can be associations with different services that may provide different functionalities like retrieving information or enabling the execution of actuation tasks. Services can also be redundant, i.e., the same type of service may be provided by different instances. In this case, there could be multiple associations of the same kind for the same Virtual Entity. Associations are important in look-up and discovery processes.

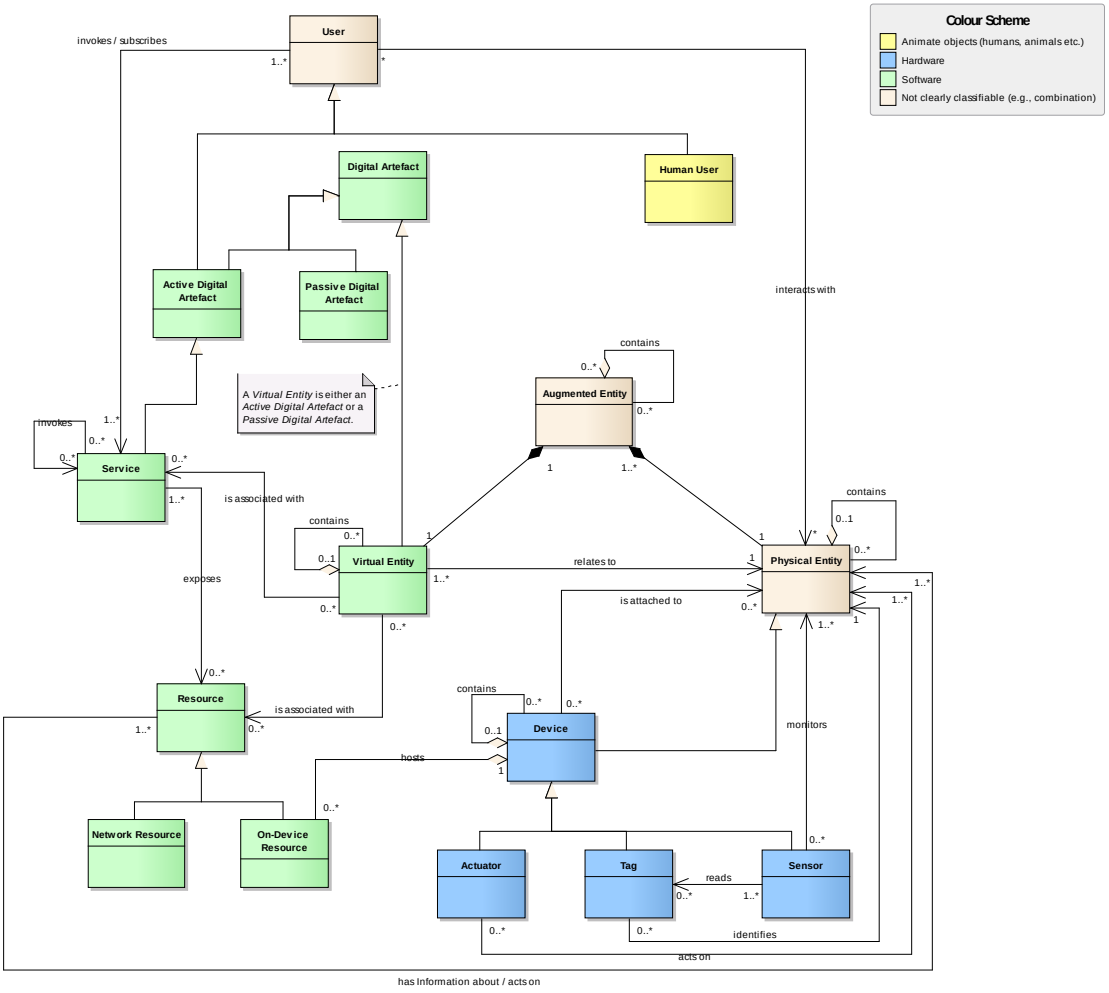

Figure 12: The IoT Domain Model

3.2.3 Detailed explanations and related concepts

The domain model as explained in the previous section is focusing on the main concepts at a high level of abstraction, capturing the essence. However, for easier understanding we give here further explanations for the following purposes:

- Explain some related aspects, e.g., the role of location;
- Show specific model instantiations, e.g., M2M interaction;
- Elaborate on certain concepts like Devices and Resources.

3.2.3.1 Devices and Device capabilities

From a Domain Model point of view, *Devices* are only technical artefacts meant to provide an interface between the digital and the physical worlds, i.e. a link between the *Virtual Entities* and the *Physical Entities*. For this reason, Devices must be able to operate both in the physical and

digital world and the domain model only focuses on their capability to provide observation and modification of the physical environment from the digital environment.

The hardware underlying the *Devices* is very important though and must have at least some degree of communication, computation and storage capabilities for the purposes of the IoT. Moreover, power resources are also very important as they can provide operational autonomy to the Devices. Many technologies and products are available and their capabilities vary quite a lot. While these capabilities might not impact directly the domain model, they are very important during the application design phase.

Communication capabilities are relative to the type of data exchanged with the *Device* (identifier, identifier + data, sensor data, or commands) and the communication topology (network, reader-tag or peer-to-peer). These aspects are very important in the IoT context and have a large impact on energy consumption, data collection frequency, and the amount of data transmitted. Security features also impact communication capabilities, because they usually introduce a consistent communication overhead. Communication capabilities indirectly impact the location of *Resources* (on-device or on the network).

Computation capabilities on the other hand have a huge impact on the chosen architecture, the implementable security features, and power resources of the *Devices*. They are also relevant for what concerns the availability of *On-Device Resources* and their complexity.

The term storage usually refers to the capability of supporting the firmware/software running on the *Device* by storing data provided by on-board sensor hardware or gathered from other *services* and needed for providing a given *Resource*. It can range from none as in the case of RFID technology to kilobytes in the case of typical embedded *Devices* or even more in case of unconstrained *Devices*.

3.2.3.2 Resources

Resources are software components that provide some functionality. They either provide some information or allow changing some aspects in the digital or physical world pertaining to one or more *Physical Entities*. The latter functionality is typically referred to as actuation. Resources can either run on a Device – hence called *On-Device Resources* – or they can run somewhere in the network (*Network Resources*). On-Device Resources are typically sensor Resources that provide sensing data or actuator Resources, e.g. a machine controller that effects some actuation in the physical world. They thus can be seen as a “bridge” between the digital and physical world. On-Device Resources may also be storage Resources, e.g., store a history of sensor measurements, but are limited by the storage capacity of the Device.

As Network Resources run on a dedicated server in the network or in the “cloud”, they do not rely on special hardware that allows a direct connection to the physical world. They rather provide enhanced services that require more system resources than Devices typical for the IoT can provide. Such resources can process data, for instance they can take sensor information as input and producing aggregated or more high-level information as output. Also, Network Resources can be storage Resources, which typically do not suffer from the limitations of their on-device counterparts. Storage Resources can store information coming from *Resources* and thus provide information about *Physical Entities*. This may include location and state-tracking information (history), static data, like product type information, and many other properties. An example of a storage Resource is an EPCIS repository (Electronic Product Code Information Services [29]) that aggregates information about a large number of *Physical Entities*. Note that also *Human Users* can update the information in a storage *Resource,* since not all known information about an entity is, or even can be, provided by Devices.

3.2.3.3 Services

Services are a widely used concept in today's IT systems. According to [6], "services are the mechanism by which needs and capabilities are brought together". This definition is very broad, and the service concept in the domain model is covering this broad definition – but restricted to technical services implemented in software. As such, services provide the link between the IoT aspects of a system and the general IT issues; IoT-related services and non-IoT services can be orchestrated together in order to form a complete system.

As it has been pointed out in [30], IoT-related services need to be explained in more detail: *IoT Services* provide well-defined and standardised interfaces, hiding the complexity of accessing a variety of heterogeneous Resources. The interaction with a *Physical Entity* can be accomplished via one or more *services* associated with the corresponding *Virtual Entity*. This association becomes important in the process of look-up and discovery. An IoT Service can thus be defined as a type of service enabling interactions with the real world.

According to [30], IoT Services can be classified according to the level of abstraction:

- ***Resource-level Services*** expose the functionality of a Device by accessing its hosted Resources. These kinds of services refer to a single Resource. In addition to exposing the *Resource's* functionality, they deal with non-functional aspects, such as dependability, security (e.g., access control), resilience (e.g., availability) and performance (e.g., scalability, timeliness).
- ***Virtual Entity-level Services*** provide access to information on a Virtual Entity level. They can be services associated to a single Virtual Entity that give access to attributes for reading attribute information or for updating attributes in order to trigger associations. An alternative is to provide a common Virtual Entity-level service with an interface for accessing attributes of different Virtual Entities, e.g. as the NGSI Context Interface [31] provides for getting attribute information.
- ***Integrated Services*** are the result of a service composition of Resource-level and Virtual Entity-level Services as well as any combinations thereof.

3.2.3.4 Identification of Physical Entities

In order to track and monitor physical identities, they have to be identified. There are basically two ways how this can be done, as is very well described in [32]: Using either natural feature identification (classified as "primary identification" in [32]) or using some type of tags or labels (classified as "secondary identification") that are attached to the *Physical Entity*.

Both means of identification are covered in the domain model. *Tags* are modelled as *Devices* that explicitly identify a *Physical Entity*. Natural feature identification can be modelled for example using a camera – a kind of *Sensor* – that monitors the *Physical Entity* and a specific *Resource* that does the natural feature extraction. The result of the natural feature extraction can then be used as a key to look up the corresponding *Virtual Entity*.

RFID tags are a prominent example often used in the context of IoT. As they come with their own electronic circuitry it seems quite natural to classify RFID tags as *Devices* in terms of the domain model. The case is less clear-cut regarding the classification of a barcode label however. As [26] points out, classifying it as a *Device* seems a little far-fetched; regarding it as a "natural feature" of the *Physical Entity* it is attached to seems more appropriate. However, as with many modelling questions, this is a matter of taste – the domain model is not prescribing which variant to use.

3.2.3.5 Context and location

As the Internet of Things pertains to the physical world, the characteristics of the physical world play an important role. All elements of the physical world are situated within a certain context and location is an essential aspect of it. All concepts in the domain model that refer to elements of the physical world, i.e., *Physical Entities*, Devices, and Human Users inherently have a location. This location may or may not be known within the IoT system.

The location of a *Physical Entity* can be modelled as an attribute of a Virtual Entity. This location could then be provided through Resources. In the case of a stationary Physical Entity, the Resource providing the location could be storage Resource, in the case of a mobile Physical Entity the Resource could be a positioning system like GPS or a tracking system like some existing indoor location systems.

3.3 Information model

The information model defines the structure (e.g. relations, attributes) of all the information (data) that is handled in a system on a conceptual level. This includes the modelling of the main concepts for information flow, storage and how they are related. The description of the representation of the information (e.g. binary, XML, RDF etc.) and concrete implementations are not part of the information model but can be found in the information view (see Section 3.3) and the related design choices (see Section 5.3.1.2).

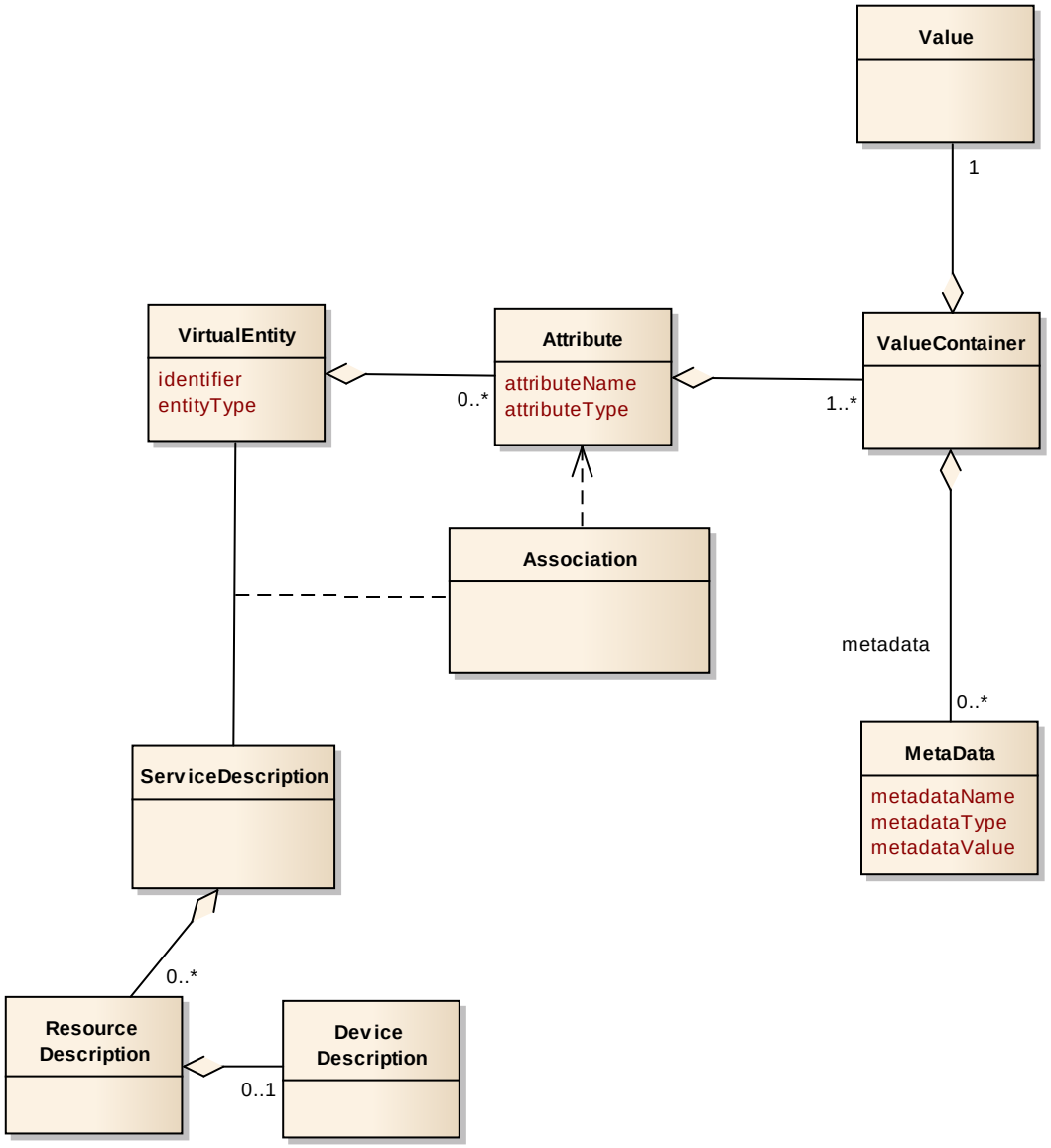

Figure 13: Information Model

The diagram in Figure 13 shows the structure of the information that is handled and processed in an IoT System. The main aspects are represented by the elements *VirtualEntity*, ServiceDescription and Association. A *VirtualEntity* models a *PhysicalEntity*, a ServiceDescription describes a service that acts as a bridge to the physical world, and finally an *Association* models the connection between the two.

Every *VirtualEntity* has a unique identifier or entityType, defining the type of the entity representation, e.g. a human, a car or even a temperature sensor. Furthermore, a *VirtualEntity* can have *zero* to *n* different attributes (Attribute). The entityType may refer to concepts in an ontology that may define what attributes a VirtualEntity of this type may have (see, for instance,

[33]). Each attribute has a name (attributeName), a type (attributeType), and one to n values (ValueContainer). This way, one can for instance, model an attribute nearbyDevices, which itself has several values. Each ValueContainer groups one Value and zero to n metadata information units (MetaData) 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. The *VirtualEntity* is also connected to the ServiceDescription via the ServiceEntityAssociation.

A ServiceDescription describes the relevant aspects of a *Service*, including its interface. In addition it may contain one (or more) ResourceDescription 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.

3.3.1 Relation of Information Model to Domain Model

The Information Model models all the concepts of the Domain Model that are to be explicitly represented and manipulated in the digital world. In addition the Information Model explicitly models relations between these concepts. The Information Model is a meta-model that provides a structure for the information. This structure provides the basis for all aspects of the system that deal with the representation, gathering, processing, storage and retrieval of information and as such is used as a basis for defining the functional interfaces of the IoT system.

Figure 14shows the relation between the Domain Model concepts and the Information Model elements. The main Domain Model concepts that are explicitly represented in an IoT system are the *VirtualEntity* and the *Service*. The latter also comprises aspects of the *Resource* and the *Device*. As the *VirtualEntity* is the model of the *PhysicalEntity* in the digital world, there is no other representation of the *PhysicalEntity* as part of the information model.

The Information Model especially details the modelling of the *VirtualEntity*. It has attributes with a name and a type and one or more values to which meta-information can be associated. Important meta-information are, for example, at what time a value was measured (i.e. timestamp), the location where a measurement took place and the quality of the measurement.

Finally, the *Association* between *VirtualEntity* and *Service* is detailed in the sense that is pertains to a certain *Attribute* of the *VirtualEntity*.

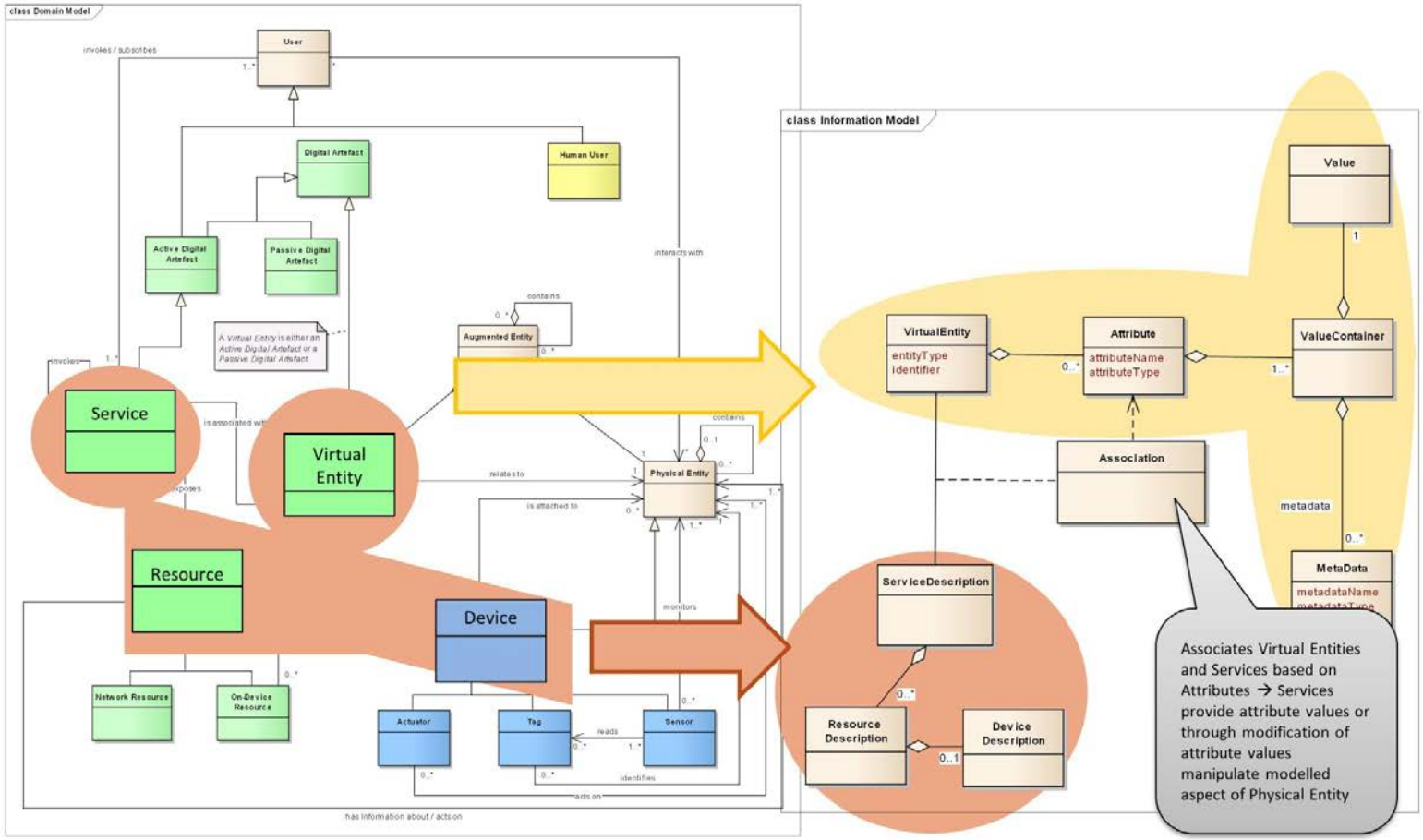

Figure 14: Relation between Domain Model and Information Model

3.3.2 Data in IoT systems

Since data is a very general term, the following section gives a distinction between different kinds of data which need to be dealt within IoT applications:

- **Real-time** data is data reflecting the current status of the system. In IoT, probably, only the data read directly from the sensor can be considered real-time data.
- **Derived data** is data that has been created perhaps by summarizing, averaging, or aggregating the real-time data through some process.
- **Inferred data** is knowledge that has been inferred by applying logic on facts provided as given data
- **Reconciled data** is real-time data that has been cleansed, adjusted, or enhanced to provide an integrated source of quality data that can be used by data analysts.

3.3.3 Other information-related models in IoT-A

Throughout IoT-A several other information related models exist. Most of them are defined in the technical work packages WP2 till WP5. More information can be found in the respective deliverables, references are given below. The next section gives a brief overview of the models and references to more detailed descriptions.

- **Entity model:** The Entity Model specifies which attributes and features of real word objects are represented by the virtual counterpart, i.e. the Virtual Entity of the respective Physical Entity. For every attribute specified in the entity model, services can be found that are able to either provide information about the attribute (sensing) or manipulate it, leading to an effect in the real world (actuating). More information about the entity model can be found in [34] Section 3.2.1.
- **Resource model:** The Resource Model contains the information that is essential to identify Resources by a unique identifier and to classify Resources by their type, like sensor, actuator, processor or tag. Furthermore the model specifies the geographic location of the Resource, the Device the Resource is hosted on (if so) as well as the IoT Services the Resource is exposed through. More information can be found in [30] Section 3.3.
- **Service description model:** Services provide access to Resources and are used to access information or to control Physical Entities. An IoT service accesses IoT Resources in order to provide information about attributes of entities or manipulates them leading to an effect in the real world. A service description describes a service, using for instance a service description language such as USDL [35]. For more information see [30] Section 4.6.3.
- **Event Model:** Event representation and processing is not yet specified. The respective documentation will be provided in the future deliverable D2.6.

3.4 Functional model

3.4.1 Functional decomposition

In the IoT-A project, functional decomposition refers to the process by which the different Functionality Groups (FG) that make up the IoT-A architectural reference model are identified and related one to another.

The main purpose of functional decomposition is to break up the complexity of a system compliant to the IoT-A ARM in smaller and more manageable parts on the one hand, and to understand and illustrate their relationship at the other hand.

Additionally, the output of functional decomposition produces a super set of functionalities that can be used to build any IoT system. The output of functional decomposition is described in this document at two levels of abstraction:

- The functional model (purpose of this section);
- The functional view (presented in Section 4.2.2).

The definition of the functional model is derived by applying the definition of a reference model to functional decomposition: “The functional model is an abstract framework for understanding the main functionality groups of the IoT-A environment and their relationships. This framework defines the common semantics and will be used for the development of IoT-A compliant functional views.”

The definition contains the following concepts that need more explanation:

- **Abstract**: The functional model is not directly tied to a certain technology, application domain or implementation. It does not explain what the different functional components are that make up a certain functionality group.
- **Functionality groups and their relationships**: The functional model contains both the functionality groups and the relationship between those parts. A list of the functionality groups alone would not be enough to make up the functional model. Both the functionality groups and their relationship are mandatory.
- **IoT-A environment**: The functional model is limited to the IoT environment described by the domain model of Section 3.2.
- **Functional view**: The functional view describes the system’s runtime functional components and their responsibilities, interfaces and primary interactions. Note that various functional views could be derived from the functional model.

As a side note, in deliverable D1.2, only the functional view was included as part of the Reference Architecture section.

3.4.2 Functional Model Diagram

The functional model diagram is depicted in Figure15: Functional Model and is derived as follows:

- From the main abstractions identified in the domain model (Virtual Entities, Devices, Resources and users) the “Application”, “Virtual Entity”, “IoT Service” and “Device” FGs are derived.
- With regards to the plethora of communication technologies that the IoT-A ARM needs to support, the need for a “Communication” FG is identified.
- Requirements expressed by stakeholders regarding the possibility to build services and applications on top of the IoT are covered by the “IoT Business Process Management” and “Service Organisation” FGs.
- To address consistently the concern expressed about IoT security and privacy, the need for a “Security” transversal FG is identified.

- Finally, the “Management” transversal FG is required for the management and/or interaction between the different functionality groups.

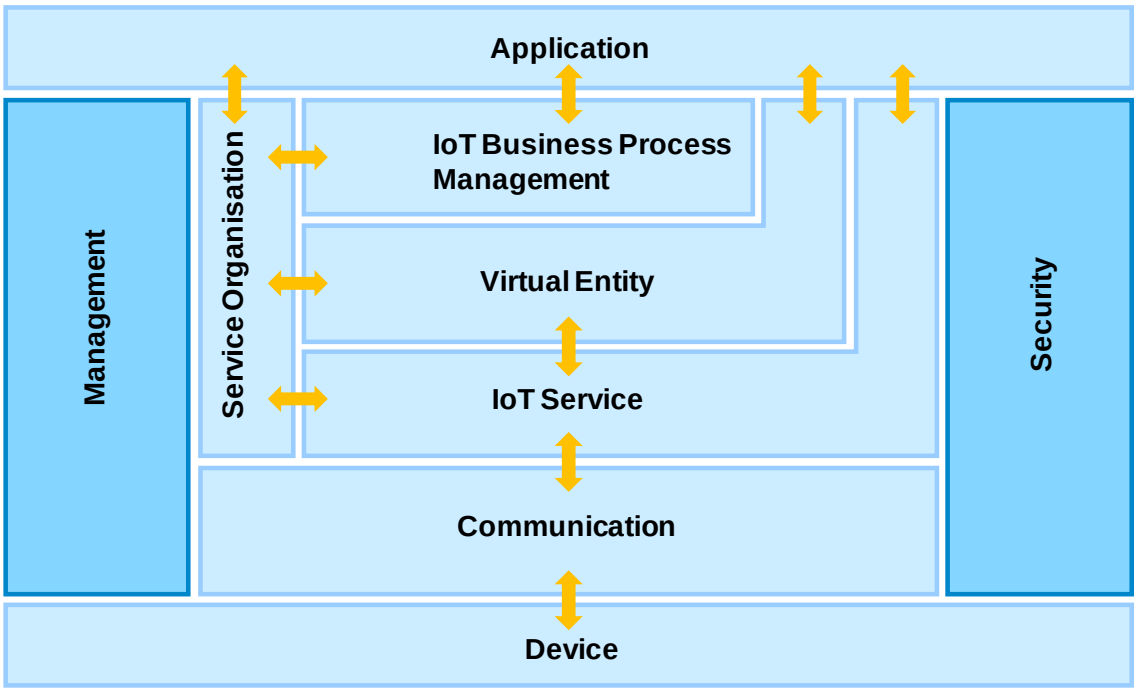

Figure15: Functional Model

The functional model contains seven longitudinal functionality groups complemented by two transversal functionality groups (Management and Security). These transversal groups provide functionalities that are required by each of the previously discussed longitudinal groups. The policies governing the transversal groups will not only be applied to the groups themselves, but do also pertain to the longitudinal groups.

As an example: for a security policy to be effective, it must ensure that there is no functionality provided by a component that would circumvent the policy and provide an unauthorised access.

Next, the relationship between the FGs is defined. As can be seen from Figure15, the functional model is a layered model and the main communication flows between the FGs are depicted with arrows. Since the transversal FGs (Management & Security) interface with most of the other FGs, their relationships are not explicitly depicted.

In the reminder of this section, each of the FGs will now be described in more detail (with exception of the Application and Device FGs since trying to capture their properties would be so generic that it does not add any value):

3.4.2.1 IoT Business Process Management

The IoT *Business Process Management Functionality Group* (BPM FG) relates to the integration of traditional business process management systems, as they are common in the enterprise world, with the IoT-A ARM. The overall aim of this FG is to provide the functional concepts and interfaces necessary to augment traditional business processes with the idiosyncrasies of the

IoT world, so that enterprises can effectively utilize IoT subsystems adhering to common standards and best practices, thus avoiding the overhead and costs of isolated and proprietary “intranet-of-things” island solutions.

In the IoT-A project, the IoT BPM FG is mapped to WP2 that deals with the integration of IoT and BPM towards a Future Internet. The IoT BPM FG provides additions and extensions to the industry standard BPMN 2.0 that include IoT-specific aspects of business processes, such as the reliability or accountability of sensor data providing information about Virtual Entities or the required processing capabilities of Devices hosting certain Resources relevant for the real world. Applications that interact with the IoT BPM FG via IoT-Augmented process models can effectively be shielded from IoT-specific details of lower layers of the functional model which greatly reduces integration costs and thus contributes to an increased adoption of IoT-A based IoT systems.

The IoT BPM FG is conceptually closely related to the *Service Organisation Functionality Group* (SO FG) and acts as a façade to applications that need to integrate an IoT-A compliant IoT system. Applications can utilize the tools and interfaces defined for the FG in order to stay on the (abstract) conceptual level of a business process while at the same time making use of IoT related functionality without the necessity of dealing with the complexities of concrete IoT service. In this respect, it provides interfaces to the IoT-A ARM that are alternatives to the more concrete VE FG and SO FG interfaces which are on a lower and more detailed level of abstraction. Naturally, the IoT BPM FG has a dependency on the SO FG, as a central concept in the execution of business processes is the finding, binding, and invoking of services that are used for each process step. The IoT BPM FG therefore relies on service organization to map the abstract process definitions to more concrete service invocations.

3.4.2.2 Service Organisation

The Service Organisation Functionality Group is the central functional group that acts as a communication hub between several other functional groups. As the primarily concept of communication within the IoT-A ARM is the notion of a “service”, the service organisation is used for composing and orchestrating services of different levels of abstraction. Within the reference architecture, it effectively links the service requests from high level FGs such as the IoT BPM FG or even external applications to basic services that IoT Resources provide (such as services hosted on a WSN gateway) and enables the association of entities with these services utilising the *Virtual Entity Functionality Group* (VE FG), so that a transformation of high level requests dealing with properties of entities (e.g. “give me please the temperature in the room 123”) down to the concrete IoT services that can be invoked to respond to these requests (e.g. “sensor service XYZ”) can be realised. In order to provide the necessary functionality to allow for querying Virtual Entities or IoT services that relate to these entities, the SO FG is comprised of service composition and service orchestration functional components that are used to resolve IoT services and also deal with the composition of services. Service composition is a central concept within the architecture, as 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 on a higher level of abstraction (e.g. the combination of a humidity sensing service and a temperature service could make up for a fire detection service).

As discussed in the previous section about the IoT BPM FG, the SO FG is closely tied to this FG, as it allows business processes or external applications to find and bind services that can be used to execute process steps or to be integrated in other ways with external applications. While the functional model as such does not have a layered structure in the strict sense of the concept, the relationships of the SO FG to the other FGs follows the layer structure in so far as the abstract service requirements from the IoT BPM FG are then processed in the VE FG in

order to manage the associations of IoT services to VEs. The requests coming from the IoT BPM FG can therefore deal with the abstract concept of entities and are only then translated to concrete IoT services that are associated with entities in the VE FG and are themselves located in the lowest FG that is relevant for the SO FG, namely the IoT Service FG. In this respect, the SO FG mitigates between the three layers of abstraction and serves as a central communication hub.

3.4.2.3 Virtual Entity & IoT Service

The Virtual Entity and IoT Service Functionality Groups include functions that relate to interactions on the Virtual Entity and IoT Service abstraction levels respectively. Figure 16 shows how the abstraction levels and how they are related. On the left side of Figure 16 the physical world is depicted. In the physical world there are a number of sensors and actuators that respectively capture and allow the change of certain aspects of the physical world. The Resources associated to the sensors and actuators are exposed as IoT Services on the IoT Service Level. Example interactions between applications and the IoT system on this abstraction level are “Give me the value of Sensor 456” or “Set Actuator 867 to On”. Applications can only interact with these services in a meaningful way, if they already know the semantics of the values, e.g. if Sensor 456 returns the value 20, the application has to be programmed or configured in such a way that it knows that this is the indoor temperature of the room of interest, e.g. Room 1.23. So 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.

Whereas interaction on the IoT Service level is useful for a certain set of applications that are programmed or configured for a specific environment, there is another set of applications that wants to opportunistically use suitable services in a possibly changing environment. For these types of applications and especially also the Human Users of such applications, the Virtual Entity level directly models higher-level aspects of the physical world that can also be used for discovering service. Examples for interactions between applications and the IoT system on this abstraction level are “Give me the indoor temperature in Room 1.23” or “Set light level in Room 2.57 to 15”. To support the interactions on the Virtual Entity level, the relation between IoT Services and Virtual Entities needs to be modelled, which is done in form of associations. For example, the association will contain the information that the indoor temperature of Room 1.23 is provided by Sensor 456.

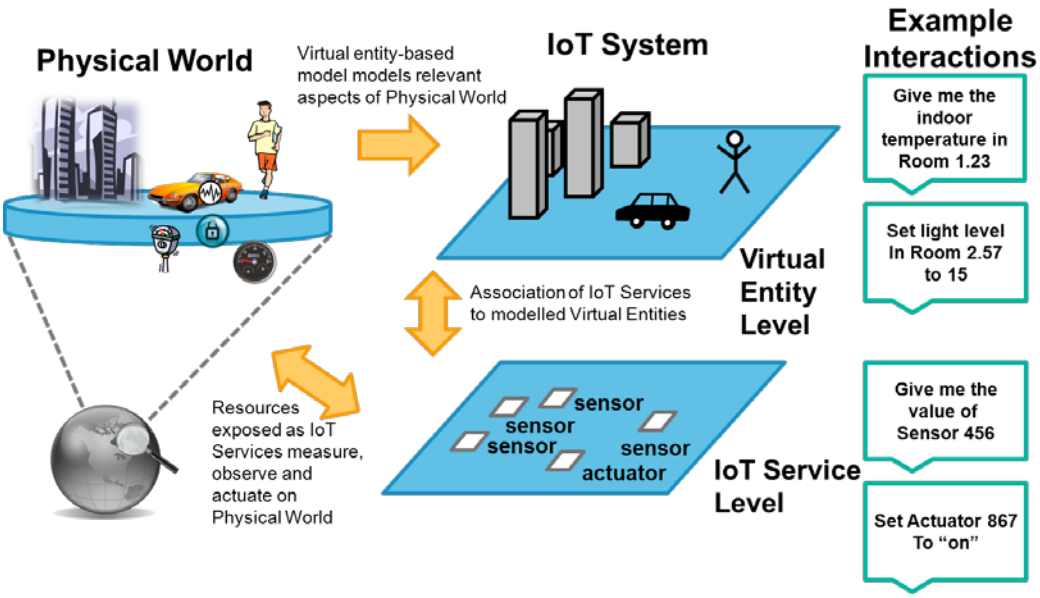

Figure 16: IoT Service and Virtual Entity abstraction levels

Virtual Entity

The VE FG contains functions for interacting with the IoT System on the basis of VEs, as well as functionalities for discovering and looking up services that can provide information about VEs or allow the interaction with VEs. Furthermore, it contains all the functionality needed for managing associations, as well as dynamically finding new associations and monitoring their validity, e.g. due to the mobility of Virtual Entities or Devices.

IoT Service

The IoT Service functional group contains the IoT Services as well as functionalities for discovery, look-up and name resolution of IoT services.

3.4.2.4 Communication

The *Communication Functionality Group* (CFG) aims to tackle all communication needs of IoT-A compliant systems. Both data plane and control plane are taken into account. The main idea is to have a slicing in functional components abstracting from the reference model layer itself being almost orthogonal, since a lot of functionalities can be achieved at different layers. The best way to understand this functional group is as the sum of his functional components. Hence, the CFG enables addressing and routes propagation in order to enable various communication modes and bypassing the limitation of hop-to-hop communication. The CFG ensures as well reliable communication and flow control, and even expands it to multiple flows, enabling in this way QoS enforcement. The CFG ensures also energy optimization exposing functions dealing directly with the radio control but also application level duty cycles. Finally, the CFG enables bridging among different networks, allowing Devices to perform as a network entry point implementing forwarding, filtering, connection tracking and packets aggregation functions. All those functionalities are as well supported by an error detection and correction infrastructure implemented by this FG.

3.4.2.5 Management

The *Management Functionality Group* (Management FG) is responsible for the composition and tracking of actions that involve one or more other FGs. One example for such an action is

turning the entire IoT system into a sleep mode during an energy-harvesting cycle. Furthermore, if the interaction of the Application and/or Device FG necessitates the composition and tracking of at least two FGs, such actions are also candidates for the sphere of responsibility of the Management FG.

By exclusion, the following management activities are thus out of the scope of the Management FG. First, activities that only pertain to a single functionality group. An example for this is the management of authorisations in the Security FG. Second, the management of interactions between functionality groups that do not require “external” intervention. An example for the latter are requests between two FGs that can be managed by the requesting functionality group only.

3.4.2.6 Security

The Security Functionality Group (Security FG) is responsible for ensuring the security and privacy of the IoT-A compliant system. It is in charge of handling the initial registration of a client to the network in a secure manner. This ensures that only legitimate clients may access services provided by the IoT infrastructure. The Security FG is also in charge of protecting the user's private parameters by featuring anonymity (ensuring that the user’s identity remain confidential when he accesses a Resource or a service) and unlink-ability (ensuring that the user may make multiple uses of Resources or services without an attacker being able to establish links between those uses). This privacy support relies on fine-tuned identity management, able to assign various pseudo-random identifiers to a single user.

The Security FG also ensures that legitimate interaction occurs between peers that are statically authorized to interact with each other, or that are trusted by each other. This happens through the use of dedicated authorization functions or through the reliance of a trust and reputation model, able to identify trustworthy peers in a privacy-capable and highly mutable architecture.

Finally, the Security FG enables secure communications between peers by managing the establishment of integrity and confidentiality features between two entities lacking initial knowledge of each other.

3.5 Communication model

The communication model aims at defining the main communication paradigms for connecting entities, as defined in the domain model. We provide a reference communication stack, together with insights about the main interactions among the actors in the domain model. We developed propose a communication stack similar to the ISO OSI 7-layer model for networks, mapping the needed features of the domain model onto communication paradigms. We also describe how communication schemes can be applied to different types of networks in IoT.

3.5.1 Communication stack

This model aims at mimicking the ISO/OSI stack, but it puts the focus on IoT systems requirements and characteristics.

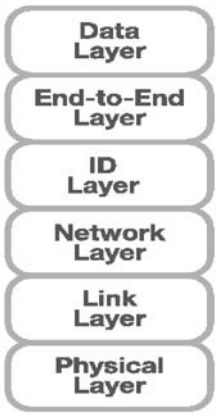

Figure 17: IoT communication stack.

The model, as depicted in Figure 17stresses the relevance of the layers above the link layer. In fact, the main strength of this communication model is the interoperability between heterogeneous networks.

In the following, details of the different layers are provided; viz. how each of them is designed to satisfy one or more particular requirements of the reference model.

Physical layer: The physical layer remains unchanged from the OSI definition. This is necessary in order to neither exclude any available technology, nor to prevent emerging solutions from being integrated into the reference model. The convergence of the different solutions taking part in the communication stack will be managed in the upper layer.

Link layer: In order to address the heterogeneousness of networking technologies represented in the IoT field, the link layer requires special attention. In fact, most networks implement similar, but customised communication schemes and security solutions. In order for IoT systems to achieve full interoperability, as well as the support of heterogeneous technologies and a comprehensive security framework, this layer must allow for diversity. But, at the same time, it needs to provide upper layers with uniform capabilities and interfaces (init, send packet, input packet, on, off, check interval?).

Network layer: Here, again, the layer provides the same functionalities as the correspondent OSI stack. However, in order to support global manageability, interoperability, and scalability, this layer needs to provide a common communication paradigm for every possible networking solution.

ID layer: The *Virtual-Entity IDentifier* (VE-ID), split from the locator, is the centre of the first convergence point in the communication stack, i.e. the ID layer. Leveraging on uniform interfaces provided by the link layers, the ID Layer allows for a common resolution framework for the IoT. Also, security, authentication, and high-end services will exploit this layer for providing uniform addressing to the many different devices and technologies in IoT networks.

End-to-end layer: This layer takes care of translation functionalities, proxies/gateways support and of tuning configuration parameters when the communication crosses different networking environments. By building on top of the ID and the network layers, the end-to-end layer provides the final building block for achieving a global M2M communication model.

Data layer: at the top of the communication stack, the entry point is the data layer. A high-level description of the data pertinent to IoT is provided by the information model (see Section 3.3).

3.5.2 Actors in IoT communication

For the communication model of IoT systems, it is important to identify the communicating system elements and/or the communicating users. One, if not the main peculiarity of the IoT is that users can belong to many disjoint categories: human or services; virtual, digital or Physical Entities. While the same picture is emerging in today's Internet use, the percentage of human-invoked communication will be even lower in the IoT. Moreover, entities can be physical, digital, or virtual. While a Physical Entity cannot directly take part to communication, it can towards its virtual counterpart.

The communication between these users needs to support different paradigms: unicast is the mandatory solution for one-to-one connectivity. However, multicast and anycast are needed for fulfilling many other IoT-Application requirements, such as data collection and information dissemination, etc.

Although the actual communication interaction is performed between two or more Devices, it is important for the communication model to track the differences between communication pertaining to human interaction, and those that only happen between services and other non-human entities. In the former case, viz. human interaction, it is important to address the quality of the communication, both in terms of quality of service and quality of data. Hereby, the degree of quality is judged by humans (human-centred QoS and quality of experience). In the latter case, M2M communication requirements do not involve quality-of-experience but QoS requirements.

3.5.3 Channel model for IoT communication

This model aims to detail and model the content of the "channel box" in the Shannon-Weaver model in the context of the IoT domain.

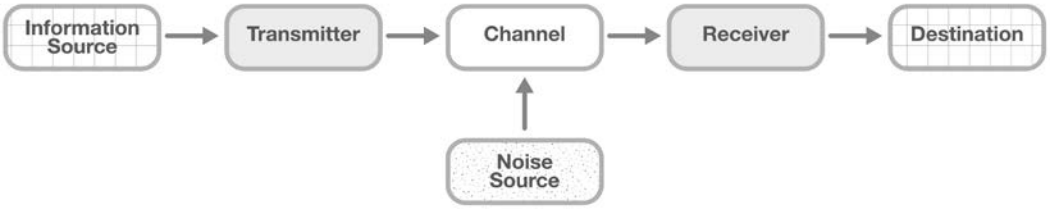

Figure 18: Schematic diagram of a general communication system.

Figure 18 depicts end-to-end abstraction of a packet delivery between distant Devices. The pair "information source" and "transmitter" is embodied by the digital entity, and the pair "receiver" and "destination" is embodied by a user, which could be a service, a human or, a distinct digital entity, or vice-versa.

Following this abstraction, and pushing it forward, here we will focus on the channel modelling. In the IoT context the channel can assume a multiplicity of forms. The channel is generally formed by a series of network Devices coupled with software.

It is important to point out that there is a distinction between the channel model in the current Internet and that of the IoT. The former is depicted in Figure 19, where the Internet provides an almost transparent “glue” between two gateways.

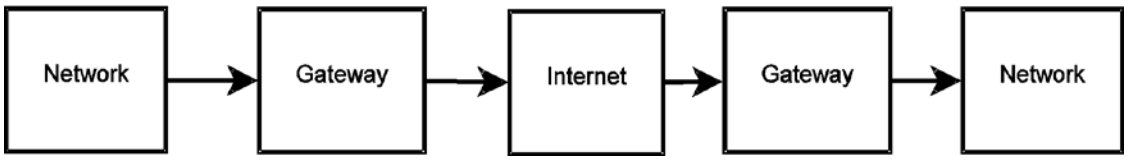

Figure 19: Channel model for the current Internet.

To proceed in modelling the channel in IoT it is important to give a definition of what we call constrained and unconstrained networks.

Unconstrained networks are characterized by high speed communication links (e.g., offering transfer rates in the Mbit/s range or higher) as the wired Internet of today. Link level transfer latencies are also short and mainly impacted by possible congestion events in the network rather than by the physical transmission technology.

Constrained networks are characterized by relatively low transfer rates, typically smaller than 1 Mbit/s, as offered by, e.g., IEEE 802.15.4. These networks are also characterized by long latencies and this is due to several factors including: 1) the involved low rate physical layer technology and 2) the power saving policy of the terminals populating these networks, which may imply the periodic power off of their radios for energy efficiency purposes.

The picture is much different in the IoT. In the simplest IoT case, namely a WSN island, the channel consists of a single constrained network, as depicted in Figure 20.

Figure 20: IoT channel for a single constrained network.

n a slightly more complicated case, the IoT channel can consist of several constrained networks, which can rely on different network technologies (see Figure 21).

Figure 21: IoT channel for communication over two constrained networks.

A different case consists of a channel embodied by a constrained network and an unconstrained one (see Figure 22).

Figure 22: IoT channel for communication constrained to unconstrained networks.

An additional case consists of a channel formed by two constrained networks intermediated by an unconstrained one, of which, one common implementation is the case we consider the most important in the IoT: the one involving two constrained networks linked by the Internet (see Figure 23).

Figure 23: IoT channel for communication over two constrained networks intermediated by the Internet.

What makes IoT very peculiar is the nature of the constrained networks it relies on. Such networks are formed by constrained Devices, and the communication between the Devices can:

1. Be based on different protocols;
2. Require additional processing in the gateways.

It is important to point out that the characteristics of each network can have a noticeable impact on the overall end-to-end communication.

3.5.4 IoT Communication model as seen from the application level

Complex IoT applications will typically encompass the orchestration of a number of digital entities. Due to the highly distributed nature of the IoT, we can assume that the orchestration will too happen in a distributed way. An application-centred diagram of IoT communication can is provided in Figure 24, where we outline which components can initiate communication with other components. A digital entity itself can, without introducing any lack of generality, be seen as a group of conceptual distributed components.

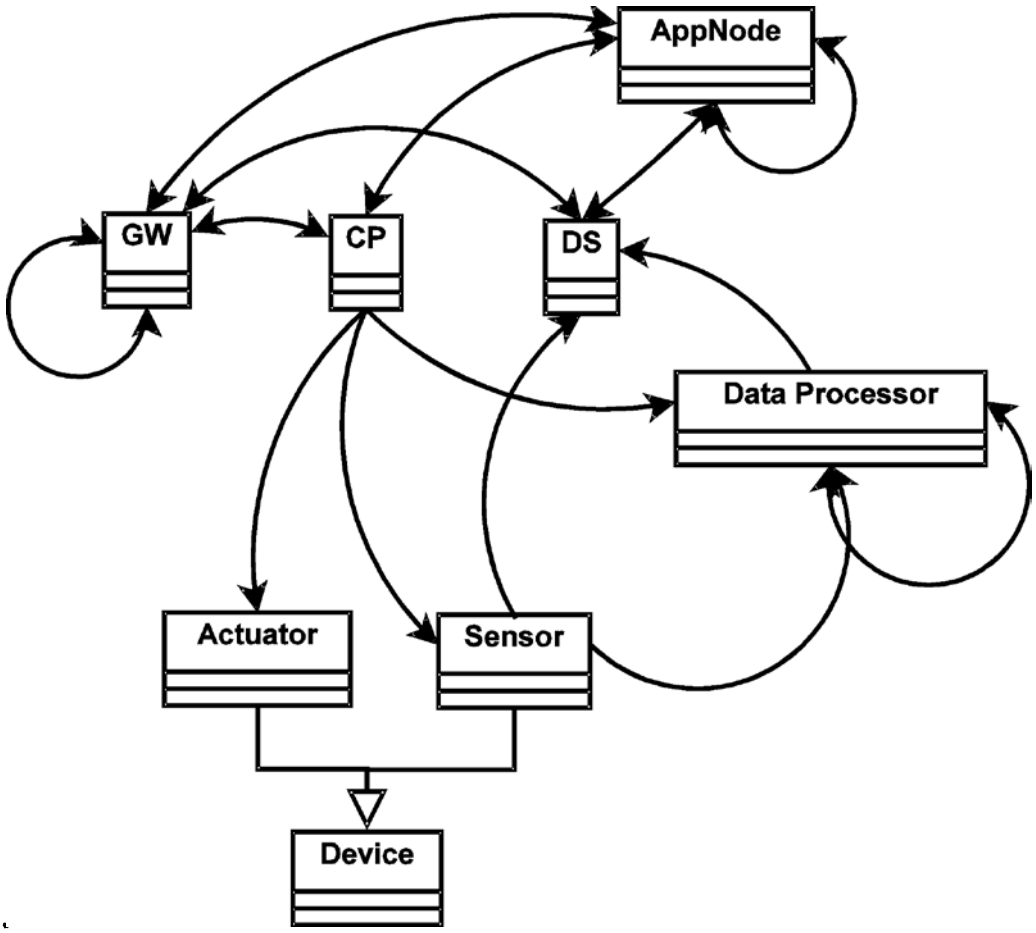

Figure 24: Communications in the IoT domain model from an application point of view. AppNode: application node; GW: gateway; CP: control point; DS: data sink.

In this section we attempt to outline the interactions between atomic “conceptual components” of the IoT applications. We can imagine a digital entity to be formed by a group of sensors and actuators. Furthermore, we can imagine a digital entity to consist of a group of data processors, data sinks, and control points, with at least an AppNode implementing the behaviour of the digital entity.

Application node (AppNode): An application node is a software agent implementing an application or part of it. AppNodes orchestrate different digital entities. The application doesn’t deal directly with sensors and actuators but it requires communication with control points and data sinks. AppNodes can obviously communicate among themselves, and in this way create a distributed application.

Control point (CP): A control point is a software agent that controls actuators and sensors, and sends related messages to sensors and actuators. A CP will communicate with sensors, actuators, and data processors, sending them configuration and control messages. A CP can handle bidirectional communication with an AppNode. The CP is usually called by AppNodes, but it is also enabled to call AppNodes after certain events, for instance an error. Control points don't process, store or forward data themselves, but orchestrate other software agents doing it.

Data end point (DS): A data end point is a software agent that receives data -which it will consume or store- directly from a sensor or a data processor. This communication is event-driven and initiated by either the sensor or the data processor. Data end points are controlled by AppNodes, but they also can initiate communications to the AppNodes on given events, like crossing a threshold.

Data processor (DP): A data processor is a software agent receiving data directly from sensors or from other data processors, performing operations like filtering or aggregation, before sending data to a data sink.

Gateway (GW): While not belonging directly to the data processing architecture it is important to have this element depicted here because of his, possible, influence in communication. A Gateway is a forwarding element, enabling various local networks to be connected. In this model, sensors and actuators cannot communicate directly with a gateway. Therefore, a control point, a data processor, or a data end point need to be hosted in the same network. A gateway can obviously communicate with other gateways and forward traffic from control points, data end points, data processors, and AppNodes.

3.6 Trust, Security and Privacy

This section will describe the high level, abstract concepts related to trust, security and privacy in the frame of IoT. These qualities of an IoT system are tightly related among themselves and impact all views. In this section we will only provide abstract introduction to these topics, which are not specific to any (reference) architecture. A description of the security functional components, the methodology used to identify them as well as definitions of terms we recall in this section can be found in [36].

Trust, Security and Privacy are horizontal qualities of an IoT system detailing the interaction between the two subjects of the Domain Model, the Service and the User, as well as the relationship with the infrastructural Security and Resolution components. Thus they impact all views:

- Information view:
  - Service descriptions should be extended in order to contain access policies to that service (and its description itself). Access policies are generally stored on the Authorization component which acts as a Decision Point. In some cases, depending on the architecture pattern adopted, it can also act as Enforcement Point. In order to do so,
    - when using Certificate- or Role-Based Access Control, this part of the Information Model shall also contain the certificates used for validation
    - when using Authentication-Based Access Control, the Authorization component shall contain a set of (subject or group) identities that are allowed to access the Service after authentication is performed with the homonymous component
  - Information about Resources and services should be hidden or made anonymous in order to protect the service provider's privacy. This means that the information returned to Users of the Resolution Services must be carefully controlled, in order not to allow the inference of private information by mining the publicly available Service descriptions.
- Functional view:
  - Security related functionalities have been derived from requirements and risk analysis performed on a series of key use cases
  - Functional components (either centralized, federated or decentralized) are needed to implement and manage the abovementioned functionalities
- Operational and deployment view

- Specific best practices needed to securely deploy and operate an IoT system should be followed. Some of them could be found in the Section 0.

3.6.1 Trust

Trust is an essential quality in IoT systems. Our definition, based on [37], is the subjectively evaluated level of probability with which an IoT system will perform a particular action or exhibit a given behaviour, both before he can monitor such action (or independently even of his capacity to be able to monitor it) and in a context in which it is relevant to him. Trust is thus a complex quality related to the extent to which a user expects an IoT system to be dependable and includes compliance to the expected functional behaviour and several security aspects.

In the frame of IoT, trust can be addressed at least at two levels: networking and application levels. *Networking trust* is related to the integrity of the routing processes, i.e. to the fact that nodes will route packets in proper way and with an acceptable timing. Generally, this is a feature that regards peripheral networks and thus it is very specific to the communication technology adopted. In the last years, with the development of WSNs, a large number of scientific works have addressed this topic, but generally the proposed solutions lack the scalability necessary to be used in the IoT context.

This section will focus on the *Application-level trust* instead. In particular, we base the trust on the following qualities of an IoT system: information quality, data-source authentication and non repudiation, confidentiality, privacy policy, information access policy and ability to access information. These qualities can be evaluated only when the same trust model applies to a specific couple of subjects (User/Service) and depend on the characteristics of the interaction and the specific context of interaction.

3.6.1.1 Trust models

IoT trust models shall be designed after the initial requirement and context analysis. They shall detail how trust is defined in a system (i.e. what is the point of view from which a subject should be evaluated), how should it be measured and how relationships with other subjects should be managed based on their trust evaluation. Generally, the system trust models apply to only a specific set of entities, pertaining to one organization and this set should be well defined during the later phases of architecture design as well as the software and infrastructure tools to evaluate the trustworthiness of other subjects.

While describing all the trust models archetypes that could be used in the frame of IoT is out of the scope of this document, a list of mandatory aspects that need to be taken into account is provided:

- The trust model **domain** defines the specific set of subjects to which a trust model applies. In the frame of IoT, this definition can be based on subscription or by the physical or network context/domain.
- Trust **evaluation mechanisms** must be defined in order to define a coherent and safe method for calculating the degree of trustworthiness of a subject. Evaluation mechanisms should also define the point of view from which trust should be evaluated and which aspects should be deemed relevant.
- The **behaviour policies** must define how subjects that use the model may interact with other subjects that can be evaluated using the same trust model. Different behaviours could be defined according to the trustworthiness of the latter. Though it is not recommended, a trust model could define specific behaviours for interacting with subjects that cannot be evaluated with that model.
- The **trust anchor** is a subject trusted by default (possibly after authentication) and used in the evaluation of third parties' trustworthiness by all the subjects that will use the same trust model. In the IoT environment this can be a service running on a node in the

same peripheral network (e.g. the gateway), a centralized service deployed in the Internet or the node itself.

- **Federation of trust** is essential in order to provide interoperability between subjects which use different trust models. In the IoT scenario where many trust models will coexist, federation will probably be a relevant phenomenon due to the very large amount of subjects of the IoT and, thus, trust models should also specify if and how trust relationships can be established among different systems.
- **M2M support** will be essential in the frame of IoT where interaction between autonomous machines, needing to dynamically identify and access Resources will be common place. Specific steps are thus needed so that machines can autonomously evaluate the trustworthiness of other machines.

3.6.1.2 Interoperability

These aspects need to be addressed during system design in order to guarantee to the IoT user as well as the system manager that the behaviour of the system is not altered by interacting with other systems. It is also worth noting that more than one trust model can apply to a given subject. For example, a user device must comply with the trust model of the IoT connectivity provider, with the restrictions set by the owner due to his privacy concerns and with the trust model that governs the service it wants to access.

Privacy policy strictness: in order to maintain coherency with the internal behaviour, a system shall not interact with other systems which have laxer privacy policies (e.g. for what concerns user profiling, data dissemination and data usage purposes).

Security settings: an IoT system cannot interact with other systems with a lower degree of security and yet provide the same degree of trust. Security should be evaluated against all its aspects. The following aspects should be taken into account individually: communication security (most notably confidentiality), user/service authentication, service availability, service access policy and system integrity and reliability. Note that, in the frame of IoT, the interoperability between two systems is also a key factor in order to evaluate the availability aspect.

Reputation: while trust is evaluated before actual interaction with a subject, reputation can be used in evaluating the trustworthiness of subjects which have already interacted with trusted referrals. Referrals can either be subjects that have evaluated the trust-related qualities in an interaction with the subject that needs to be evaluated or reference registries which monitor subjects (or gather data about their behaviour) specifically for evaluating their trustworthiness. Moreover, as proposed in [36], reputation metering could also be provided by an infrastructure component. The advantage of this solution is that

- Each node would have a reference, trusted subject for uploading and retrieving reputation information in a secure way. There would be no need to discover such a service and security material for authenticating the Service would already be in place.
- As such, an infrastructure component would be trusted by default
- Evaluation of trust policy integrity in a federated environment is easier and more reliable.

M2M compatibility: it is essential that in a M2M environment the agents in control of the machines are able to autonomously

- model the trustworthiness of the other agents they need to interact with;
- authenticate other subjects;

- determine/retrieve and enforce the access policies which apply to the users requesting their services (only applies to service subjects).

3.6.2 Security

Security is an essential quality of an IoT system and it is tightly related to specific security features which are often a basic prerequisite for enabling Trust and Privacy qualities in a system. This section is an architecture primer for detailing the security features in IoT solutions.

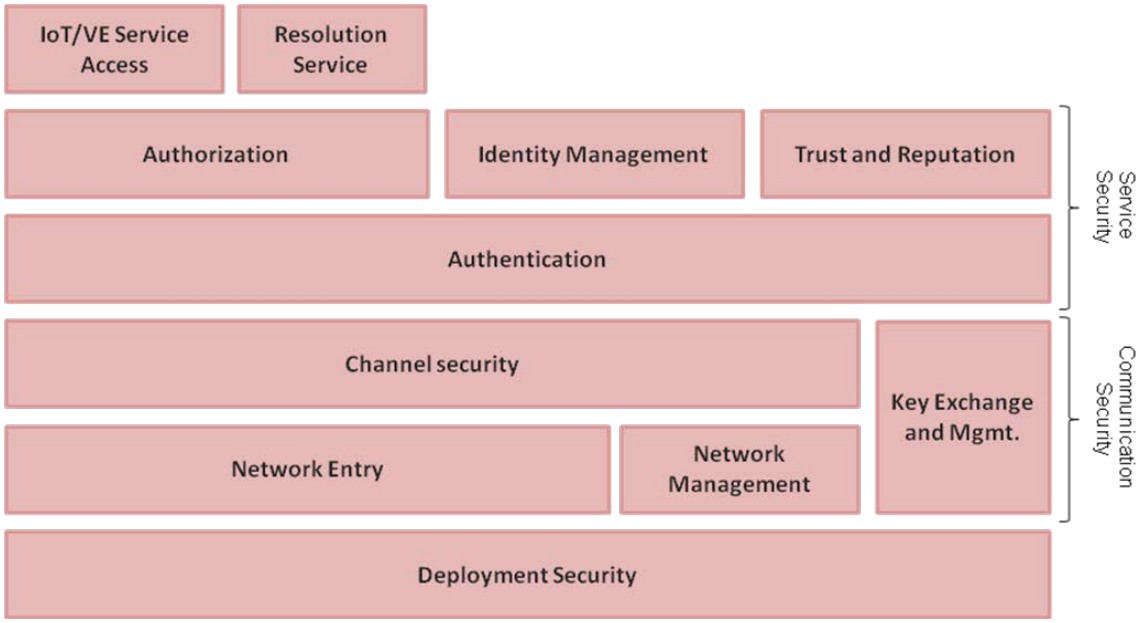

Figure 25: Security features and general layering. Some architectures can exhibit a slightly different approach, depending on the actual implementation. For example, some optional components might not have been implemented while some features could have been implemented in a cross-layered approach.

Figure 25 presents a generic overview[2] of the approach to security features and components. There can be different implementations of each of these layers providing different levels of security. All of them are optional though recommended. Some of them though come with requirements on the lower layers (e.g. in some architectures, in order to implement authorization, authentication is needed first).

Service Security is well described in [36] and thus the topic will not be investigated in this context. For what concerns Communication Security, the following section will provide an overview at Reference Model level.

[2] As this approach does not cover all possible implementation solutions we avoid using the term “model” here.

3.6.2.1 Communication Security

As stated in [38], securing the communication at protocol level is very difficult in the case of IoT, since device communication and processing capabilities resources are constrained. This typically entails that bandwidth, power supply, processing capabilities, and security features have to be balanced.

The model proposed hereafter has been designed under the assumption that the IoT device space can be divided into two main categories: constrained networks (NTU) and unconstrained networks (NTC) (See Networks and communication entities, Chapter 2 in [38]). The domain of constrained devices contains a great heterogeneity of communication technologies (and related security solutions) and this poses a great problem in designing a model encompassing all of them. Examples for such communication technologies can be found in the literature [10].

Moreover, there is also the problem of different functional and communication patterns between connected devices and auto-ID devices, which adds to the complexity of the situation.

One solution can be to provide a security model with a very high degree of abstraction, so that the above heterogeneities can be mitigated. A very high degree of abstraction is not useful though, as it doesn't provide enough constraints for defining a RA. The same issue may arise again when implementing a concrete architecture. As in the Communication Model (see Section 3.5), we will address the problem by introducing profiles which will group the highly heterogeneous devices into groups characterized by given specifications. Standard interfaces will also be provided in the future for making security features interoperable.

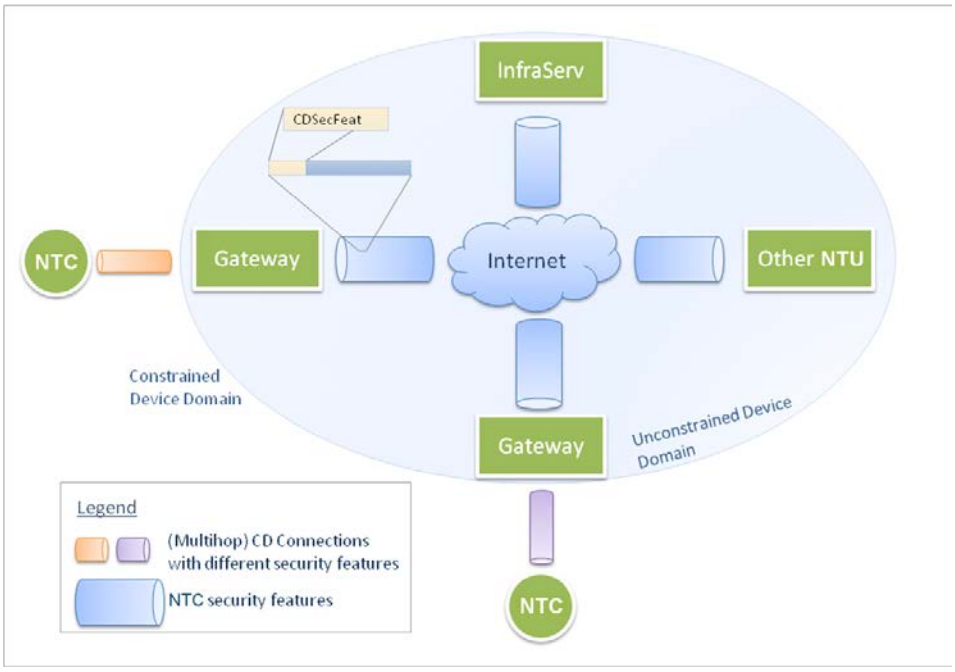

Figure 26: Providing the best security features for the lower layers in each IoT domain by introducing Gateways with adaptive functions aimed to provide scalability functions (including security scalability). NTC: Constrained Device Network; NTU: Unconstrained Device Network. CDSecFeat: implementation of security feature for the constrained device leverages the extension of the functionalities of gateway devices.

On the edge between the domains of unconstrained and constrained devices, gateways have the role of adapting communication between the two domains (see Figure 26). This usually involves the adaptation between different protocol-layer implementations up to the network or ID layer (see Section 3.5). The fact that gateways are generally unconstrained devices means that they can also be used for scaling down functionalities (such as security) from the NTC domain to the NTU domain. They can also be used for managing security settings in peripheral (constrained-device) networks. Gateways have to provide the following functionalities in order to hide underlying heterogeneity:

- Protocol adaptation between different networks (by definition).
- Tunnelling between themselves and other nodes of the NTU domain. (Optional; impacts on trust assessment.)
- Management of security features belonging to the peripheral network. (Optional)
- Description of security options related to traffic originated by a node attached to the gateway. (Authentication of source node, cryptographic strength, ...)
- Filtering of incoming traffic (i.e. traffic sent to one of the nodes attached to the gateway) according to network policies, user-defined policies and destination-node preferences. (Optional)

Gateways are not relevant and thus invisible at the end-to-end layer level. Despite the availability of end-to-end security features available at ID-layer level, lower layers might need security features for securing network entry and point-to-point communication which are specific to the single network sub-domains. The security settings provided by these layers should be available to the applications that need and manage the communication.

While gateways are the most suited element that could provide information about the security settings of underlying networks, this solution poses some issues. Thus, other solutions will also be taken into account and analysed, especially in the way they will interact with existing standards and protocols. This activity will be carried out during the next phase of the IoT-A project.

4 Reference architecture

4.1 Short definition of views and perspectives

A system architecture, and thus by default, a reference architecture, needs to answer a wide range of questions. Such questions can, for instance, address:

- Functional elements
- Interactions of said elements
- Information management
- Operational features
- Deployment of the system

What the user of an architecture expects is an architectural description, viz. “a set of artifacts that documents an architecture in way its stakeholders can understand and demonstrates that the architecture has met their concerns.” [13]. Instead of providing these artifacts as monolithic description one often chooses to delineate them by so-called views. The idea behind doing so is to focus on system aspects that can be isolated. Views make both the derivation of the architecture and its validation easier. The above bullet-point list provides examples of such views. A more detailed discussion of views and how we adapted them to the reference-architecture realm is provided in the next section.

In the past it has been found that views are unfortunately not enough for describing system architectures rather that many stakeholder aspirations are of a qualitative nature [12]. Such qualitative aspirations cut across more than one view. Such cross-cutting qualitative aspects are referred to perspectives, of which privacy is one example. A more detailed introduction to perspectives is provided in Section 4.3.

The joint use of views and perspectives in architecture descriptions is described in more detail in the pertinent literature [12].

4.2 Views

Views are used during the design and implementation phase of a concrete system architecture and defined by Rozanski and Woods [12] in the following way:

“A view is a representation of one or more structural aspects of an architecture that illustrates how the architecture addresses one or more concerns held by one or more of its stakeholders.”

Viewpoints aggregate several concepts to make the work with views more easy. The IEEE Standard 1471 defines viewpoints as follows:

“A viewpoint is a collection of patterns, templates, and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint and the guidelines, principles, and template models for constructing its views.“

Some typical examples for viewpoints are Functional, Information, Concurrency, Development, Deployment and Operational viewpoints.

4.2.1 Usage for the IoT-A Reference Architecture

The IoT-A Reference Architecture is domain- and application- independent and is therefore not compatible to the concept of views and viewpoints one-by-one. But the idea behind the concept is nevertheless helpful and will be adopted for the use within the IoT-A Reference Architecture:

“A view is a representation of one or more structural aspects of an reference architecture that illustrates how this reference architecture can be adopted to address one or more concerns held by its stakeholders.”

“A viewpoint is a collection of patterns, templates, and conventions for constructing one type of view. It defines the stakeholders whose concerns are reflected in the viewpoint and the guidelines, principles, and template models for constructing its views.“

The views and viewpoints will be complemented with Design choices which will be used to illustrate one or more different implementation aspects with their advantages, disadvantages and relations to the perspectives, see Section 5.3.1.

The following sections will therefore address the functional view, information view and the deployment and operation view.

4.2.2 Functional

4.2.2.1 Functional View Process

The functional view is defined by applying the methodology defined in Section 2.2 to functional decomposition as can be seen in Figure 27:

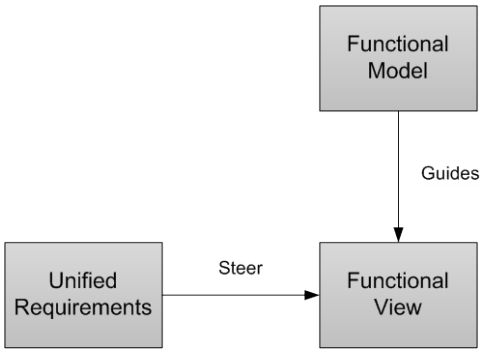

Figure 27: Functional view process

In a first step, the unified requirements are mapped to the different functionality groups of the functional model.

Next, clusters of requirements of similar functionality are formed and a functional component for these requirements defined.

Finally, the functional components are refined by cross-checking against the Description of Work and by discussing with the technical work packages.

The viewpoints used for constructing the functional view are hence:

1) The unified requirements;

2) The Functional Model;

3) The Description of Work.

Once all functional components are defined, system use cases, sequence charts and interface definitions are made, which can be found back in Annex C.

The functional view diagram is depicted in Figure 28 and shows the 9 functionality groups (FGs) of the functional model:

- The Application FG and Device FG are out-of-scope of the IoT-A Reference Architecture and are coloured in yellow.
- Management and Security FG are transversal functionality groups and are coloured dark blue.

For each of the Functionality Groups, the *Functional Components* (FC) are depicted.

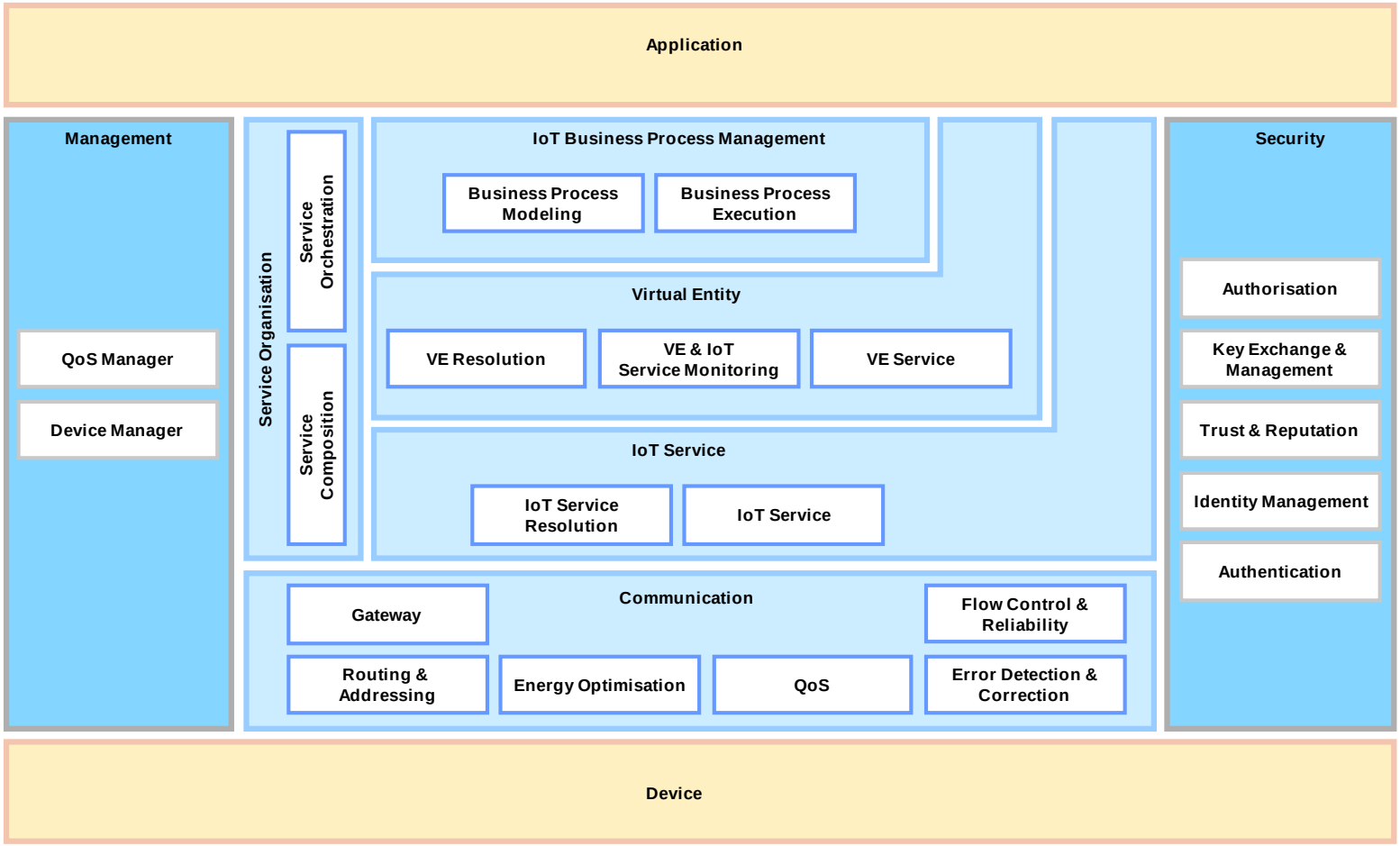

Figure 28: Functional View

In the following sub-sections, the FCs will be described in more detail.

4.2.2.2 IoT Business Process Management

Business Process Modelling

Description	Provides an environment for the modelling of IoT-Aware business processes that will be serialised and executed in the process-execution functional component. The business-process-modelling component is located within the IoT Business Process Management layer.
Additional description	The component is described in detail in deliverables D2.2 and in the upcoming D2.4.
Pertaining requirements	UNI.031, UNI.032, UNI.211, UNI.212, UNI.213, UNI.214, UNI.215
Technical use case	C.1.1.1

Default function set

Function name	*Function description*	*Usage example*
IoT business-processes modeler	Provides the tools necessary for modelling business processes using the standardised notation,[3] i.e. using novel modelling concepts specifically addressing the idiosyncrasies of the IoT ecosystem.	C.1.1.1

Business Process Execution

Description	Executes IoT-Aware business processes that will be modelled in the Business Process Modelling FC. This execution is achieved by utilising IoT services that are orchestrated in the Service Organisation layer. The Business Process Execution component is located within the IoT Business Process Management layer.
Additional description	The component is described in detail in deliverables D2.3 and D2.5.
Pertaining	UNI.008, UNI.031, UNI.032, UNI.229, UNI.230, UNI.232

[3] A such notation is currently been developed as part of the IoT-A project.

requirements	
Technical use case	C.1.1.1 & C.1.2.1

Default function set

Function name	*Function description*	*Usage example*
Deploy process models to execution environments	Activities of IoT-Aware process models are applied to appropriate execution environments, which perform the actual process execution by finding and invoking appropriate IoT services.	C.1.1.1
Align application requirements with service capabilities	For the execution of applications, IoT service requirements must be resolved before specific services can be invoked. For this step, the Business Process Execution component utilises the service organization functionalities.	C.1.1.1
Run application	After resolving IoT services, the respective services are invoked. The invocation of a service leads to a progressive step forward in the process execution. Thus, the next adequate process based on the outcome of a service invocation will be executed.	C.1.1.1

4.2.2.3 Service Organisation

Service Orchestration

Description	The Service Orchestration component resolves the IoT Services that are suitable to fulfil service requests coming from Business Process Execution component or from IoT-A users. The Service Orchestration component resides in the Service Organisation layer.
Additional description	The component is described in detail in deliverables D2.3 and the upcoming D2.5.
Pertaining requirements	UNI.008, UNI.043, UNI.096, UNI.230, UNI.232, UNI.234, UNI.235 UNI.244, UNI.245, UNI.247, UNI.251, UNI.252, UNI.253
Technical use case	C.1.1.1 & C.1.2.1

Default function set

Function name	*Function description*	*Usage example*
Orchestrate IoT services	This function resolves the appropriate services that are capable of handling the IoT-user's request. If needed, temporary resources will be set up to store intermediate results that feed into service	C.1.2.2

	composition or complex event processing	

Service Composition

Description	The Service Composition FC resolves services that are composed of IoT Services and other services in order to create services with extended functionality. The Service Orchestration component is located within the Service Organisation layer.
Additional description	The component is described in detail in deliverables D2.3 and D2.5.
Pertaining requirements	UNI.043, UNI.096, UNI.234, UNI.235 UNI.244, UNI.245, UNI.247, UNI.251, UNI.252, UNI.253
Technical use case	C.1.1.1 & C.1.2.1

Default function set

Function name	*Function description*	*Usage example*
Support flexible service compositions	Provides dynamic resolution of complex services, composed of other services. These composable services are chosen based on their availability and the access rights of the requesting user.	C.1.2.2
Increase quality of information	This function can be used for increasing quality of information by combining information from several sources. For example, an average value –with an intrinsically lower uncertainty- can be calculated based on the information accessed through several resources.	C.1.2.2

4.2.2.4 Virtual Entity

Virtual-Entity (VE) Resolution

Description	The VE Resolution is the FC which provides the functionalities to the IoT User to retrieve associations between VEs and IoT Services. The functionalities needed by the Service Client in brief are: • Discovery functionality discovers the associations without any prior knowledge about the VE. The VE specification and the VEServiceSpecification, which describes the relation between the VE and the IoT Service, are used as parameters of the query. • Lookup is a functionality which enables the User to access Associations between the particular VE and IoT Services fitting the VEServiceSpecification based on a known VE-ID uniquely identifying a VE.

Additional description	The component is described in detail in deliverable D4.3, Section 2.2.2.
Pertaining requirements	UNI.016, UNI.030, UNI.036, UNI.095, UNI.098, UNI.099, UNI.401, UNI.402, UNI.403, UNI.404, UNI.406, UNI.408, UNI.414, UNI.415, UNI.416, UNI.422, UNI.423, UNI.428, UNI.432, UNI.623
Interface description	D4.3 2.2.2.1
Technical use case	C.3.1

Default function set

Function name	*Function description*	*Usage example*
Discover VE-related services	Discovers new (mostly dynamic) associations between VE and associated services. For the discovery qualifiers such as location, proximity, and other context information can be considered. If no association exists, it is created.	C3.1.2
(Un)Subscribe to association discovery	(Un)Subscribes the User for continuous notifications about Associations that fit provided VESpecification and the VEServiceSpecification, to be sent to a provided notificationCallback function A unique SubscriptionID is returned to the subscribing User that can be used to match notifications to the subscription and to unsubscribe.	C3.1.2
Lookup VE-related services	Searches for services exposing resources related to a VE.	C3.1.2
(Un)Subscribe to association look-up	(Un)Subscribes the User for notifications about Associations based on the VE-ID and the VEServiceSpecification, to be sent to the provided notificationCallback function. A unique SubscriptionID is returned to the subscribing User that can be used to match notifications to the subscription and to unsubscribe.	C3.1.2
Insert association	Inserts a new association between a VE and the IoT services that are associated to this entity.	C3.1.2

Delete association	Deletes an association between a VE and the IoT services that are associated to this entity.	C3.1.2
Update association	Updates associations between a VE and the IoT services that are associated to this entity.	C3.1.2

Virtual-Entity & IoT Service Monitoring

Description	The VE & IoT Service Monitoring functional component is responsible for automatically finding new associations, which are then inserted into the VE resolution functional component. New associations can be derived based on existing Associations, service descriptions and information about VEs.
Pertaining requirements	UNI.016, UNI.418, UNI.419, UNI.420, UNI.421
Interface description	The component is described in detail in deliverable D4.3, Section 2.2.3.1.
Technical use case	C.3.2.1

Default function set

Function name	*Function description*	*Usage example*
Assert static Association	Creates a new static (i.e. un-monitored) association between VEs and services described by the provided Association.	C3.2.2
Discovered dynamic Association	Creates a new dynamic (i.e. monitored) association between VEs and services described by the Association	C3.2.2
Association No Longer Valid	Deletes the Association from the VE Resolution.	C3.2.2
Update Association	Updates the Association upon changes.	C3.2.2

Virtual-Entity Service

Description	An Entity service represents an overall access point to a particular entity, offering means to learn and manipulate the status of the entity. Entity services provide access to an entity via operations that enable reading and/or updating the value(s) of the entities' attributes. The type of access to a particular attribute depends on the specifics of that attribute (read only / write only or

	both). A specific VE service can provide VE History storage functionality, to publish integrated context information (VE context information - dynamic and static), VE state information, VE capabilities.
Pertaining requirements	UNI.016, UNI.240, UNI.409, UNI.410

Default function set

Function name	*Function description*
Read Attribute Value	Returns the value of attribute parameter for the entity
Set Attribute Value	Sets the value of attribute parameter for the entity

4.2.2.5 IoT Service

IoT Service Resolution

Description	The IoT Service Resolution provides all the functionalities needed by the User in order to find and be able to contact IoT Services. The IoT Service Resolution also gives Services the capability to manage their service descriptions, so they can be looked up and discovered by the User. The User can be either a Human User or a software component. The functionalities needed in brief are: • Discovery functionality finds the IoT Service without any prior knowledge about the ServiceID. The functionality is used by providing a service specification as part of a query. • Lookup is a functionality which enables the User to access the service description having prior knowledge regarding the ServiceID. • Resolution function resolves the ServiceIDs to locators through which the User can contact the service. Other functionalities provided by the IoT Service Resolution are the management of the service descriptions. IoT Services can update, insert or simply delete the service descriptions from the IoT Service Resolution component. It is also possible that these functions are called by management components and not the IoT Services themselves.
Additional description	The component is described in detail in deliverable D4.3, Section 2.2.1.
Pertaining requirements	UNI.030, UNI.095, UNI.098, UNI.099, UNI.417, UNI.423, UNI.425, UNI.426, UNI.427, UNI.429, UNI.601, UNI.614, UNI.623
Interface description	D4.3 2.2.1.1
Technical use case	C2.1.1

Default function set

Function name	*Function description*	*Usage example*
Resolve Service with ID	Resolves the address of an IoT service given its ID.	C2.1.2
(Un)Subscribe to Service Resolution for service with given ID	(Un)Subscribes to the resolution (based on the ServiceID) to receive notifications whenever the ServiceURL changes on the provided callback. The IoT Service Resolution returns a SubscriptionID to the User that can be used to match notifications to the subscription and to unsubscribe.	C2.1.2
Lookup Service given ID	Retrieves the description of an IoT service given its ServiceID.	C2.1.2
(Un) Subscribe to Service Lookup for service with given ID	(Un)Subscribes to the resolution (based on the ServiceID) to receive notifications whenever service description changes over or service becomes unavailable. A unique SubscriptionID is returned to the subscribing User that can be used to match notifications to the subscription and to unsubscribe.	C2.1.2
Discover Service matching specification	Retrieves a list of services descriptions matching a given specification.	C2.1.2
(Un)Subscribe to Service Discovery for services matching given description	(Un)Subscribes for continuous notifications about services that fit the provided Service Specification, to be sent to the provided callback function. A unique SubscriptionID is returned to the subscribing User that can be used to match notifications to the subscription and to unsubscribe.	C2.1.2
Update Service with description	Updates service entry with new service description	C2.1.2
Insert Service with description	Adds new service entry with given service description	C2.1.2
Delete Service with ID	Removes service given a service ID.	C2.1.2

IoT Service

Description	Software component exposing a *resource* through a well-defined interface to make it accessible to other parts of the IoT system, often via the Internet. Typically, resource services expose the functionality of a device by accessing

	its hosted resources. These kinds of services refer to a single resource. In addition to exposing the resource' functionality, they deal with non-functional aspects, such as dependability security (e.g. access control), resilience (e.g. availability) and performance (e.g. scalability, timeliness). A particular type of IoT Service can be the Resource History Storage that provides storage capabilities for the measurements generated by resources (resource history).
Pertaining requirements	UNI.005, UNI.018, UNI.022, UNI.041, UNI.062, UNI.236, UNI.239, UNI.240, UNI.241, UNI.429, UNI.607, UNI.610, UNI.613, UNI.614 , UNI.623

4.2.2.6 Communication

Gateway

Description	This function component aims to enable bridging among different networks. It can tackle different network layers. It enables the device implementing it to act as an entry point to another network. The main duties of this Functional Component are to keep track and enforce protocol translations and address translations needed to cross network borders. Such tracking can be stateless (in case packets contain all needed information to be translated) or stateful. Additional Gateway functionalities are filtering, buffering and aggregation.
Pertaining requirements	UNI.048, UNI.095, UNI.096, UNI.506

Default function set

Function name	*Function description*
Forward	This function deals only with packet forwarding. That's the basic function of the gateway.
Connection Tracking/Aggregation	This function deals with several packets/messages at once, keeping a state between receptions. Packets are correlated between them and/or aggregated.
Filter	This function filter packets/messages analysing their headers or contents.

Flow Control & Reliability

Description	This Functional Component tackles all the needs for reliability and flow control. It can be deployed at MAC/point-to-point layer (e.g. a reliable MAC), at transport protocol level (e.g. TCP), at the application protocol layer (CON messages in COAP) or even in the application itself. It is important to note that communication modes different from unicast may need distributed strategies for both reliability and flow control. In order to implement such strategies offline messaging or gossip protocols may be required.

Pertaining requirements	UNI.508, UNI.610, UNI.615, UNI.618

Default function set

Function name	*Function description*
Connect	This function couples a destination to a source
I/O Control	This function enables exposing options of the channel/socket/connection
Send message	Confirmable or non-confirmable message, still needs to conform to the flow control.
TX	This function blindly emits the packet/message. Other functions are built on top of it.
RX	This function blindly receives the packet/message. Other functions are built on top of it.

Routing & Addressing

Description	This Functional Component aims to enable new devices to enter a network, get an address and be reachable. Coming to functionalities the crucial ones are: assigning addresses, maintaining routing tables or routing policies, and forwarding data packets. Resource Directory somewhat transcends from this functionality, but it probably leverages on it. A peculiarity of IoT is that in order to optimize traffic or algorithm simplicity some assumptions on the traffic patterns are useful. A case could be the "collection pattern", in which all the traffic will be multipoint to point.
Pertaining requirements	UNI.048, UNI.509, UNI.617

Default function set

Function name	*Function description*
Address Control	This function enables Discovery, Offer, Request, Acknowledge of addresses, according to the selected method.
Routes Control	This function enables to Purge, Remove and Add routes.
Neighbor Information Subscribe	This function enables subscribe or observe of the neighbour routing information.
Get Graph	Get the routing graph, which could be also quite simple, being

	represented only by the given parent.
Routing Information Control	This function enables the Information Solicitation or a Destination Advertisement, according the selected methods.
Calculate Rank	Calculate Rank of peers. This function enable as well to elect/select the best parent if the routing algorithm requires that.

Energy Optimization

Description	This functional component aims to manage energy consumption while communicating. It is generally implemented shutting off the radio for a given time frame. It could be implemented at low layers (e.g. Radio Duty Cycle at MAC) or even in higher layers, like application protocol (e.g. COAP sleep option) or even at the application level.
Pertaining requirements	UNI.100, UNI.101, UNI.505, UNI.508, UNI.512

Default function set

Function name	*Function description*
Sleep	Inform communication peer that the node is going to sleep in a given time, optionally informing when he will wake up.
Wake up	Inform communication peer that the node wake up, optionally informing when he will sleep again.
Radio Control	This function enables to turn the radio ON or OFF, at a given granularity.
Check Interval	This function enables to check time intervals occurred among communication related events.

QoS

Description	This Functional Component refers only to quality of communication services, namely fast paths, latency, packets priority, and so on. e.g. RED, SFB. Real time systems need to interact with QoS, so metrics could be exposed. QoS manager (in Management FG) doesn't actually enforce QoS, and it seems to have a broader scope. This FC is crucial to enforce QoS "wishes" from the QoS manager.
Pertaining requirements	UNI.026, UNI.028, UNI.060, UNI.614

Default function set

Function name	*Function description*
Traffic Class Attach	Attach a communication channel to a given traffic class. E.g. ioctl, qattach, diffserv.
Reserve	This is an alternative (higher level) model for QoS enforcement, in which instead of attaching the communication channel to an existing traffic class an actual reservation is made. This function leverage on the Traffic Class Attach function at local level and on the QoS manager on the remote.

Error Detection & Correction

Description	Error detection is deeply present at different layers, e.g. UDP checksum or ICMP. Error detection can be distributed as well, if taken at higher layers. Probably, on the other hand, Error correction is not a core functionality for IoT, but that could be deemed important, to keep coherence of communication and network topology during transient troubles (attacks or more generally incidents).
Pertaining requirements	UNI.012, UNI.020, UNI.021, UNI.066, UNI.089, UNI.608

Default function set

Function name	*Function description*
Compute Signature	Compute the signature for a given buffer according a given algorithm. Generally is used to sign a packet, but according the algorithm could be used to both sign and verify. E.g. UDP checksum.
Verify Signature	Verify that the signature present in the packet match with data part. This is optional in case that signature and algorithm are not symmetrical.
Report Error	Report an Error. E.g. ICMP host unreachable, MAC collisions.
Time Synchronization	This function provides Time Synchronization. The detail of the message exchange is not detailed here because several protocols exist. E.g. NTP.

4.2.2.7 Security

Authorization (AuthZ)

Description	The authorization component is a front end for performing access control decisions based on access control policies. This access control decision can be called whenever access to a restricted resource is requested. For example, this function is called inside the IoT service resolution component, to check if a user is allowed to perform a lookup on the requested resource. This is an

	important part of the privacy protection mechanisms.
Additional description	The component is described in detail in deliverable D4.2
Pertaining requirements	UNI.002, UNI.067, UNI.502, UNI.503, UNI.606, UNI.610, UNI.611, UNI.619, UNI.623, UNI.626
Technical use case	C4.1

Default function set

Function name	*Function description*	*Usage example*
Authorize	From assertion, service description and action type, determine whether the action is authorized or not.	C4.1.2

Authentication (AuthN)

Description	The Authentication component is involved in user and device authentication. It checks the credentials provided by a user, and, if valid, it returns an assertion as result, which is required to use the IoT Service Client. Upon checking the correctness of the credentials supplied by a newly joining node, it establishes secured contexts between this node and various entities in its local environment.
Additional description	The component is described in detail in deliverable D4.2
Pertaining requirements	UNI.501, UNI.503, UNI.610, UNI.612, UNI.619, UNI.626
Technical use case	D1.3 Annex C4.1

Default function set

Function name	*Function description*	*Usage example*
Authenticate	Authenticate a user based on provided and credentials, and return an assertion upon successful authentication	C4.1.2
Verify	Verify whether an assertion provided by a user is valid or invalid.	C4.1.2

Identity Management (IM)

Description	The Identity Management component addresses privacy questions by

	issuing pseudonyms and accessory information to trusted subjects so that they can operate (use or provide services) anonymously.
Additional description	The component is described in detail in deliverable D4.2
Pertaining requirements	UNI.001, UNI.423, UNI.424, UNI.605, UNI.606, UNI.611, UNI.612, UNI.624
Technical use case	C4.1

Default function set

Function name	*Function description*	*Usage example*
Create Pseudonym	Optional feature by which the discovered identifier will be replaced by a pseudonym and provided to the user	C4.1.2

Key Exchange and Management (KEM)

Description	The Key Exchange and Management component is involved to enable secure communications between two or more IoT-A peers that do not have initial knowledge of each other or whose interoperability is not guaranteed, ensuring integrity and confidentiality.
Additional description	The component is described in detail in deliverable D4.2
Pertaining requirements	UNI.022, UNI.047, UNI.062, UNI.501, UNI.503, UNI.607, UNI.608, UNI.609

Default function set

Function name	*Function description*
Establish Secure Connection	Requests the establishment of a given security context between the issuing node and a remote target. Security parameters, including the type of secure communications enablement, are provided.

Trust and Reputation Architecture (TRA)

Description	The Trust and Reputation Architecture component collects user reputation scores and calculates service trust levels.
Additional description	The component is described in detail in deliverable D4.2
Pertaining	UNI.062, UNI.610, UNI.613, UNI.619, UNI.622

requirements	

Default function set

Function name	*Function description*
Request Reputation Information	This function is invocated at a given remote entity to request reputation information about another entity. As input parameters, a unique identifier for the remote entity (subject), as well as the concrete context (what kind of service) is given. As a result a reputation bundle is provided.
Provide Reputation Information	This function is invocated at a given remote entity to provide reputation information (recommendations or feedback) about another entity. As input parameters, a unique identifier for the entity to be assessed (subject), as well as the concrete context, the given score and a timestamp are given. As a result, the corresponding reputation element is provided.

4.2.2.8 Management

QoS Manager

Description	Manages the QoS when using functionalities provided by several Functionality Groups of the architecture. Information about QoS capabilities and usage is provided to services and applications.
Pertaining requirements	UNI.614

Default function set

Function name	*Function description*
Assess policy	Manages consistency of the QoS requirements expressed and supported by the different functionality components
Get QoS policy	Informs about the QoS supported by the system's Functionality Groups.

Device Manager

Description	Manages the composition of non-device Functionality Groups with the Device Functionality Group.
Pertaining requirements	UNI.014, UNI.066, UNI.505

Default function set

Function name	*Function description*
Set device default configuration	Provides device with a default configuration that can be used when the device is initialising.
Update device firmware	Updates the firmware of the device.

4.2.3 Information

One of the main purposes of Connected and Smart Objects in the IoT is the exchange of information between each other and also external systems. Therefore the way how to define, structure, store, manipulate, manage and exchange information is very important. The information view helps to generate an overview about static information structure and dynamic information flow.

Based on the IoT Information Model, the information view gives more detailed information about how the relevant information is to be represented in an IoT system. As we are describing a reference architecture as opposed to a specific system architecture, various representation alternatives will then be discussed as part of the design choices in Section 5.3.1.2.

Going beyond the IoT Information Model, 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.

The current version of the Information View focuses on the information description, the information handling and the information life cycle. In a future version we will provide more details on the flow of information through the system and the components involved. Given the current level of detail, we will provide a viewpoint only for modelling the type system of Virtual Entities.

4.2.3.1 Information Description

Description of Virtual Entities

The Virtual Entity is the key concept of any IoT system as it models the Physical Entity or the Thing that is the real element of interest. As specified in the information model, Virtual Entities have an identifier, 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. Of special importance is the modelling of the entity type. The entity type can be used to determine what attributes a Virtual Entity instance can have, defining their semantics. The entity type can be modelled based on a flat type system or as a type hierarchy, enabling sub-type matching. For modelling entity type hierarchies, ontologies or UML class diagrams can be used. Of course, this choice is related to the design choice on how the overall Virtual Entity information is represented.

Figure 29: Example for flat entity type model

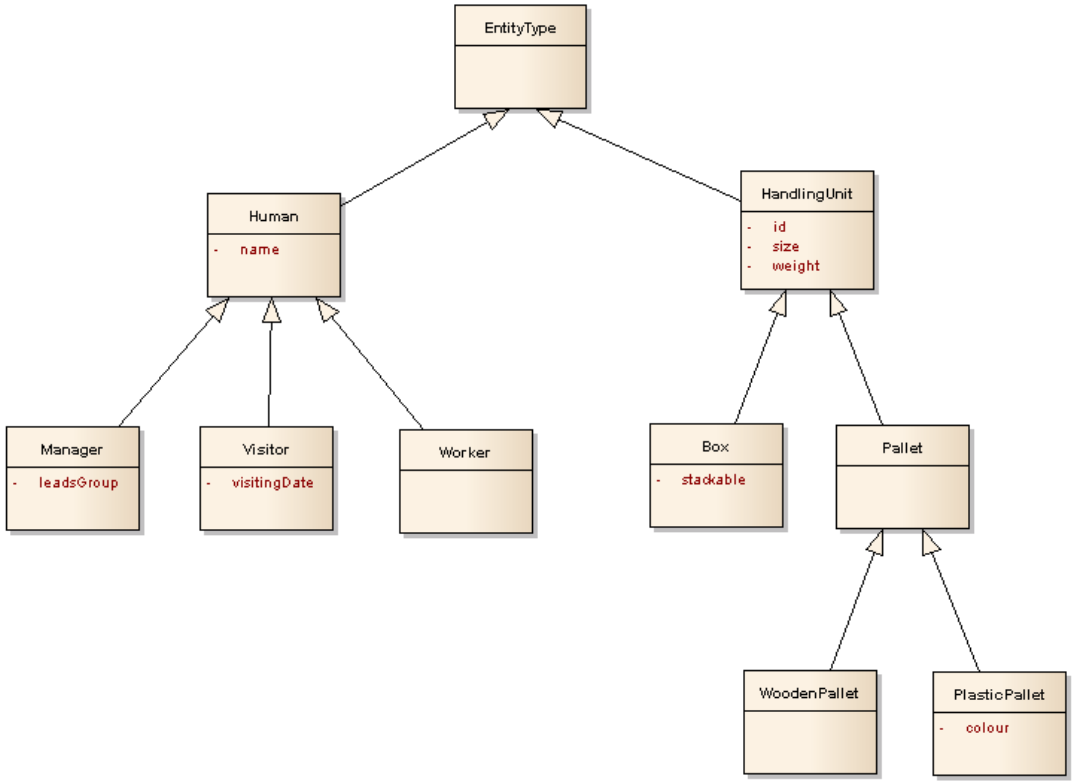

Figure 30: Example for hierarchical entity type model

Viewpoint for modelling entity type hierarchies

Entity types are similar to classes in object-oriented programming, so UML class diagrams as shown above are suitable for modelling entity types. As shown in Figure 30: Example for hierarchical entity type model the generalization relation can be used for modelling sub-classes, creating an entity type hierarchy. Alternatively, ontology languages like OWL also provide the means for modelling classes and sub-classes, so they can also be used for modelling type hierarchies. This is especially useful, if information in the IoT system is to be modelled using ontologies.

Service descriptions

Services provide access to the functionality with which information provided by resources, which may run on IoT devices, can be retrieved or actuation tasks can be executed. As a basis for finding and interacting with services, services need to be appropriately described, which is done in the form of service descriptions. Service descriptions contain information about the interface of the service, both on a syntactic as well as a semantic level, e.g. the required inputs, the provided outputs or the necessary pre-conditions as well as post-conditions. Furthermore, the service description may include information regarding the functionality of the resources, e.g. the type of resource, the processing method or algorithm etc., or information regarding the device on which the resource is running, e.g. it's hardware or its geographical location. Different specification languages for describing services are available, so again, there are different design choices.

Associations between Virtual Entities and services

Services can provide information or enable actuation, but the services themselves may not be aware for what Virtual Entity / Virtual Entities they can provide what kind of information or enable what kind of actuation. This information is captured by associations that relate to the Virtual Entity and the service. 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.

4.2.3.2 Information Handling

Information in the system is handled by IoT services. IoT services may provide access to On-Device Resources, e.g. sensor resources, which make real-time information about the physical world accessible to the system. Other IoT service 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.

IoT services are registered to the IoT system using service descriptions. Service descriptions can be provided by the services themselves, by users or by special management components that want to make the service visible and discoverable within the IoT system. The IoT Service Resolution is responsible for managing service descriptions and providing access to service descriptions. In detail the IoT Service Resolution provides an interface for discovering service descriptions based on service specifications given by the requestor, for looking up a service description based on the identifier of a service and for resolving a service identifier to a service locator. The latter can also be seen as a convenience function as the service description also contains the currently valid service locator.

Associations can be registered with the VE Resolution by services that know for what Virtual Entities they can provide information, by users, by special management components, or by the VE & IoT Service Management component that automatically derives them based on information existing in the system, including service descriptions and other associations.

4.2.3.3 Information Life Cycle

Information provided by sensor resources is transient in nature and may not even be measured or observed without a specific request. Information stored by a storage resource may be permanently stored there or have an expiry data after which the information is to be removed. For this purpose a storage resource may have to implement mechanisms that remove such information on a regular basis. It is also possible to adapt the granularity of information that is stored over time, i.e., for a certain time interval all the information is stored, for a further time interval only a fraction of the information is kept whereas the rest is discarded. Such a scheme may allow the definition of multiple such time intervals and also requires specific underlying mechanisms that can implement the scheme.

To avoid keeping service descriptions of services that no longer exist, a time-out mechanism needs to be implemented by the IoT Service Resolution. After the time-out has been reached without a renewal of the service description, the service description should be automatically be removed. This in turn requires that the components originally providing the service description renew the registration of the service description before the time-out is reached. The same applies for associations stored by the VE Resolution.

4.2.4 Deployment & Operation

The deployment and operation view aims at providing users of the IoT-A Reference Model with a set of guidelines to drive them through the different design choices that they have to 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 networking diagram for the deployment.

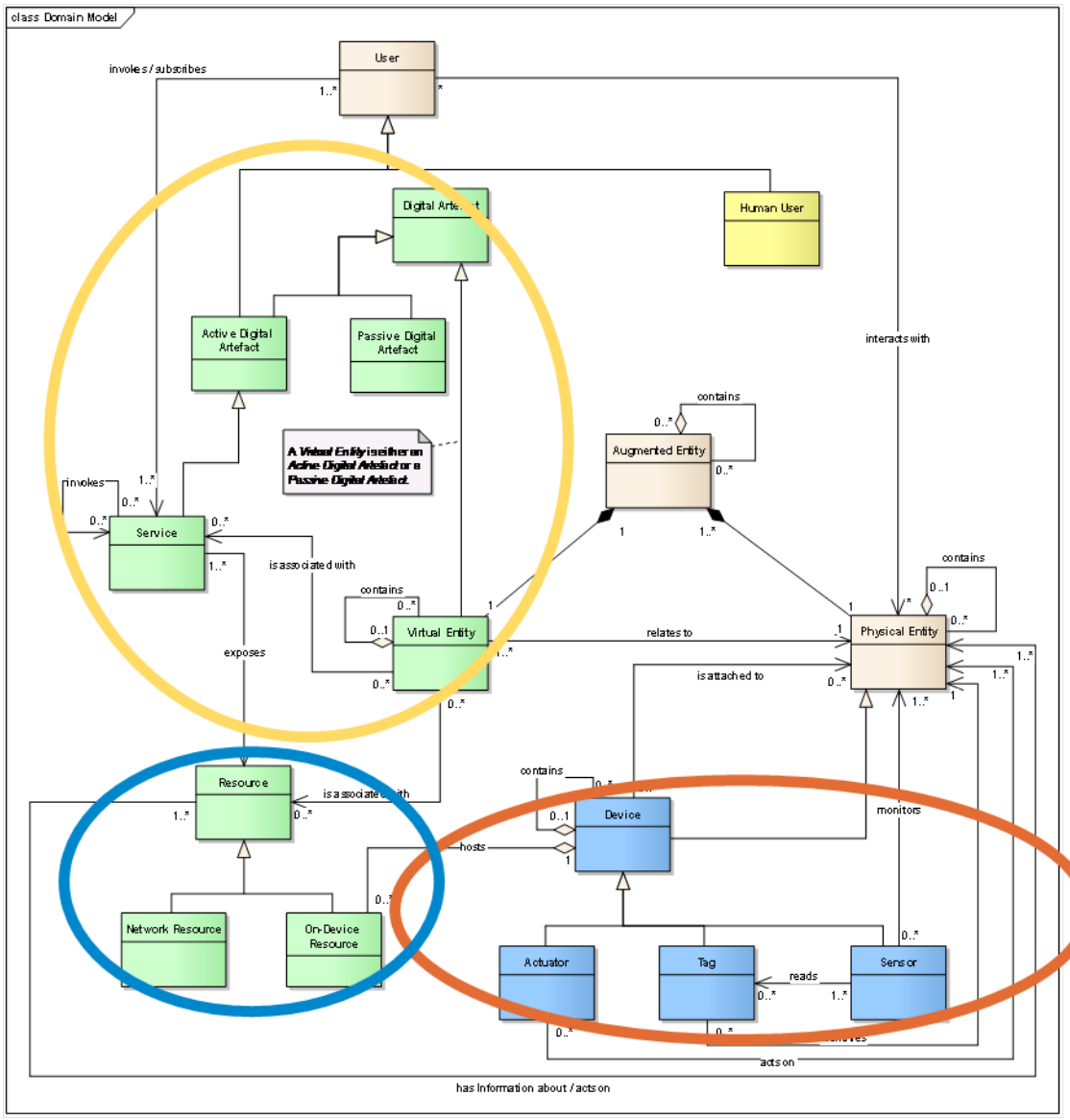

Figure 31 Domain model groups

Since a complete analysis of all the technological possibilities and their combination would make this document overlong, this section will identify those categories that have the strongest impact on IoT system realization: In particular, starting from the domain model, we found three main element groups (see Figure 31): devices, resources, and services in red, blue and yellow,

respectively, in the figure. 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:

1) the domain model diagram is used as a guideline to describe the specific application domain; to this extent UML diagram can be used to further detail the interaction among the many elements composing the target application;

2) Network connectivity diagrams have to 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;

3) Device description (such as datasheets and information manuals) have to be used to map actual hardware on the service and resource requirements of the target system.

First of all, devices in IoT systems include the whole spectrum of technologies ranging from the simplest of the radiofrequency tags to the most complex of the servers. The unifying characteristics are mainly two: On the hand, every device is connected with one another forming a part of the IoT; and, on the other hand, every device is “smart”, even though with different degree of complexity, in that it provides computational capabilities. 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 somewhat intrinsic to the target application. However, choosing among the different connectivity types is not as straightforward as different choices may provide comparable advantages, but in different areas. In this section, we will simply detail the main options for device connectivity, leaving their impact on the different perspective for Section 5.3.1 in which the design choices for the deployment view are discussed.

1. Sensor & Actuator Networks
2. RFIDs and smart tags
3. WiFi or other unconstrained technologies
4. Cellular networks

As a consequence 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. Although, IoT-A and WP3 in particular suggest a communication protocol suite aimed at the interoperability among different technologies with IP as the common denominator, the system designer, may be forced to make suboptimal choices [please refer to deliverable D3.3 for the specificities]. In particular, we identified the following possibilities:

1. IoT protocol suite: This is the main direction supported by this project and providing 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 patterns, the system designer has to account for services and resources. 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 byte.
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 has to 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 is capable of hosting the temperature resource with a simple service for providing it, but, if a more complex service is needed, the software has to deployed on a more powerful device as per option 2 or 3.

At 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 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. 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.

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 service core engine, which is in charge of semantically retrieving resources and services, discovering new elements and binding users with data, resources, and services. 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.

4.3 Perspectives

Architectural decisions often address concerns that are common to more than one view, or even all of them. These concerns are often related to non-functional or quality properties. We are following the approach of Rozanski/Woods [11] which suggest special perspectives to address these aspects of an concrete architecture. They emphasize the importance of stakeholders requirements just like we do within our project. Therefore we are adopting their definition of perspective for usage within IoT-A:

An architectural perspective is a collection of activities, tactics, and guidelines that are used to ensure that a system exhibits a particular set of related quality properties that require consideration across a number of the system's architectural views.[11]

where a quality property is defined as

A quality property is an externally visible, non-functional property of a system such as performance, security, or scalability. [11]

The stakeholder requirements clearly show a need of addressing non-functional requirements. Based on them we identified the perspectives which are most important for IoT-systems: Evolution and Interoperability, Availability and Resilience, and Performance and Scalability. As these requirements are requiring some kind of quality for a real system, the perspectives aim more on the actual architecture of a system than to an reference architecture.

We got more than ten requirements concerning the Evolution and Interoperability perspective, around six concerning Availability and Resilience, and ten more related to performance and scalability. As can be seen from the definition above there is a close relationship between Perspectives, Views and Best Practices.

4.3.1 Evolution and interoperability

The Evolution perspective addresses the fact that requirements change and software evolves sometimes rapidly. We identified a second, closely related, perspective namely interoperability which plays especially in IoT a crucial role. The vision of the Internet of Things is still evolving itself. There are, for example, not yet all used technologies mature enough, and there are for sure many more technologies to come in the future.

Desired Quality	The ability of the system to be flexible in the face of the inevitable change that all systems experience after deployment, balanced against the costs of providing such flexibility
IoT-A Requirements	UNI.003, UNI.010, UNI.012, UNI.047, UNI.048, UNI.049, UNI.071, UNI.093, UNI.094, UNI.096
Applicability	Important for all systems to some extent; more important for longer- lived and more widely used systems. IoT systems are expected, as an emerging technology, to be highly affected by evolution and interoperability issues.
Activities	characterize the evolution needs assess the current ease of evolution consider the evolution tradeoffs rework the architecture
Tactics	contain change create extensible interfaces apply design techniques that facilitate change apply metamodel-based architectural styles build variation points into the software use standard extension points achieve reliable change preserve development environments

Table 4: Evolution and interoperability (adopted from [11], extended with IoT specific aspects)

4.3.2 Performance and scalability

This perspective addresses two quality properties which are closely related: Performance and Scalability. Both are, compared to traditional information systems, even harder to cope with in a highly distributed scenario as we have it in IoT.

Desired Quality	The ability of the system to predictably execute within its mandated performance profile and to handle increased processing volumes in the future if required
IoT-A Requirements	UNI.008, UNI.926, UNI.027, UNI.028, UNI.066, UNI.089,UNI.101,UNI.102
Applicability	Any system with complex, unclear, or ambitious performance requirements; systems whose architecture includes elements whose performance is unknown; and systems where future expansion is likely to be significant. IoT systems are very likely to have unclear performance characteristics, due to its heterogeneity and high connectivity of devices.

Activities	capture the performance requirements create the performance models analyze the performance model conduct practical testing assess against the requirements rework the architecture
Tactics	optimize repeated processing reduce contention via replication prioritize processing consolidate related workload distribute processing over time minimize the use of shared resources reuse resources and results partition and parallelize scale up or scale out degrade gracefully use asynchronous processing relax transactional consistency make design compromises

Table 5: Performance and scalability (adopted from [11], extended with IoT specific aspects)

4.3.3 Trust, Security and privacy

Due to the importance of these free topics, while elsewhere in the document we use a integrated trust/security/privacy perspective, in this context, we provide a vertical approach and detail each of them.

4.3.3.1 Trust

This perspective addresses application level trust and specifically aims at supporting the architecture design process so that the resulting system behaves as expected and is dependable

Desired Quality	A complex quality related to the extent to which a subject expects (subjectively) an IoT system to be dependable regarding in all the aspects of its functional behaviour.
IoT-A Requirements	UNI.062, UNI.065, UNI.099, UNI.407, UNI.408, UNI.602, UNI.604, UNI.605, UNI.620, UNI.622
Applicability	Relevant to the systems that share the use of resources with subjects that are not a priori trusted.
Activities	capture trust requirements perform risk analysis check interoperability requirements define trust model
Tactics	harden root of trust ensure physical security and implement tampering detection

	ensure and check data freshness consider the impact of security/performance tradeoffs on trust avoid “leap of faith” use (trusted) infrastructural Trust and Reputation Agents for scalability use security imprinting check system integrity often balance privacy vs. non-repudiation (accountability)

Table 4: Trust perspective (adopted from [11], extended with IoT specific aspects)

4.3.3.2 Security

This perspective addresses two qualities which are closely related: Communication Security and Operational Security. Communication Security in most contexts is a prerequisite for Operation Security. The Security perspective is tightly related to the Trust, privacy and Performance perspectives.

Desired Quality	Ability of the system to enforce the intended confidentiality, integrity and service access policies and to detect and recover from failure in these security mechanisms.
IoT-A Requirements	UNI.062, UNI.407, UNI.408, UNI.410, UNI.412, UNI.413, UNI.424, UNI.502, UNI.507, UNI.604, UNI.609, UNI.611, UNI.612, UNI.617, UNI.618, UNI.624
Applicability	Relevant to all IoT systems.
Activities	capture the security requirements check interoperability requirements conduct risk analysis use infrastructural Authentication components that support more Identity Frameworks for scalability and interoperability Use infrastructural or federated KEM to secure communication initiation and tunnelling between gateways for interoperability Use an Authorization component to enable interoperability with other systems define security impact on interaction model address all aspects of Service and Communication Security integrate the trust model and support privacy features identify security hardware requirements consider performance/security tradeoffs

	validate against requirements
Tactics	harden infrastructural functional components authenticate subjects define and enforce access policies secure communication infrastructure (gateways, infrastructure services) secure communication between subjects secure peripheral networks (data link layer security, network entry, secure routing, mobility and handover) avoid wherever possible wireless communication physically protect peripheral devices avoid OTA device management if not properly secured

Table 5: Security perspective (adopted from [11], extended with IoT specific aspects)

4.3.3.3 Privacy

Desired Quality	
IoT-A Requirements	UNI.001, UNI.002, UNI.410, UNI.412, UNI.413, UNI.424, UNI.501, UNI.606, UNI.611, UNI.623, UNI.624
Applicability	Relevant to the systems to all IoT systems.
Activities	capture the security requirements conduct risk analysis evaluate compliancy with existing privacy frameworks.
Tactics	use an Identity Management component that supports Pseudonymization avoid transmitting identifiers in clear especially over wireless connections minimize unauthorized access to implicit information (e.g. deriving location information from service access requests) validate against requirements consider the impact of security/performance tradeoffs on privacy enable the user to control the privacy (and thus security and trust) settings

	balance privacy vs. non-repudiation (accountability)

Table 5: Privacy perspective (adopted from [11], extended with IoT specific aspects)

4.3.4 Availability and resilience

As we are dealing with distributed IoT systems, where a lot of things can go wrong the ability of the system to stay operational and to effectively handle failures that could affect a systems availability is crucial. The Activities and Tactics shown in Table 6 can be used for dealing with that kind of problems.

Desired Quality	The ability of the system to be fully or partly operational as and when required and to effectively handle failures that could affect system availability
IoT-A Requirements	Uni.040, UNI.050, UNI.058, UNI.065, UNI.092, UNI.610
Applicability	Any system that has complex or extended availability requirements, complex recovery processes, or a high profile (e.g., is visible to the public)
Activities	capture the availability requirements produce the availability schedule estimate platform availability estimate functional availability assess against the requirements rework the architecture
Tactics	select fault-tolerant hardware use high-availability clustering and load balancing log transactions apply software availability solutions select or create fault-tolerant software design for failure allow for component replication relax transactional consistency identify backup and disaster recovery solution

Table 6: Availability and resilience (adopted from [11], extended with IoT specific aspects)

5 Best practices

5.1 Overview

A major goal of best practices in IoT-A is to provide guidance for system architects. The guidance best practices provides is manifold: giving examples; highlighting design choices and lessons learnt from developers; recommending ways to document an implementation; and providing a detailed analysis of an IoT use case highlighting the technical and business benefits that IoT-A provides.

In this section we focus on the IoT Reference Model, in particular the IoT Domain Model as it currently is most mature. In particular we provide guidelines and examples on how to use the IoT Domain Model.

We will also provide a first overview of design choices that one typically faces during the derivation of specific architectures.

In the forthcoming version of this Deliverable, viz. D1.4 and D1.5, we will provide a more thorough discussion of the design choices and also guidelines and examples for the other aspects of the IoT Reference Model. The design-choice discourse will include a discussion of techniques, and solutions that are adapted for addressing specific issues while implementing the IoT Reference Architecture part of the IoT ARM.

Last but not least, we will put the advice contained in best practices in action. For one, we will analyse the requirements and constraints of an existing domain-specific application. We will highlight the steps taken to model this application using the IoT ARM. We will also provide an example derivation of a concrete architecture.

5.2 Usage of the IoT Reference Model

Similar to the identification of stakeholders and actors in standard software engineering practices, the IoT Domain Model is used in a first step of the architectural design process in order to:

1. Identify Physical Entities and related Virtual Entities
2. Identify resources (at least from a functionality perspective)
3. Identify devices (or device options)

The identification of resources and devices is used together with the IoT Communication Model to define the communication paradigms and how these devices and resources interact. This is comparable to interaction models in standard software engineering practices. And finally, the services to be used and where they should be deployed are analysed.

5.2.1 Guidelines for using the IoT Domain Model

This section is intended for architects who want to apply the IoT Domain Model on a specific use case. We discuss typical instantiations of the IoT Domain Model. These model cases can be used as basic patterns when doing concrete modelling.

5.2.1.1 Deployment configurations

Figure 32 shows a range of deployment configurations for resources, services, and users. In the first case from the left, resource, service, and the service are running on the same device. This is a configuration in which we have a powerful device, and the interaction with the user is local. In the second case from the left, the service of the user is running somewhere else, e.g., in the cloud, and the interaction is thus not local. The API used between the service client and the service, however, is the same. In the third configuration the service is not running on the device, but in the cloud. This is a typical configuration for a constrained device that may not be able to expose a user interface across the network. For example, due to energy constraints or other limiting factors, such a device may sleep most of the time and is therefore not be able to always handle user requests. The interface between the service and the resource may be very specific and proprietary.

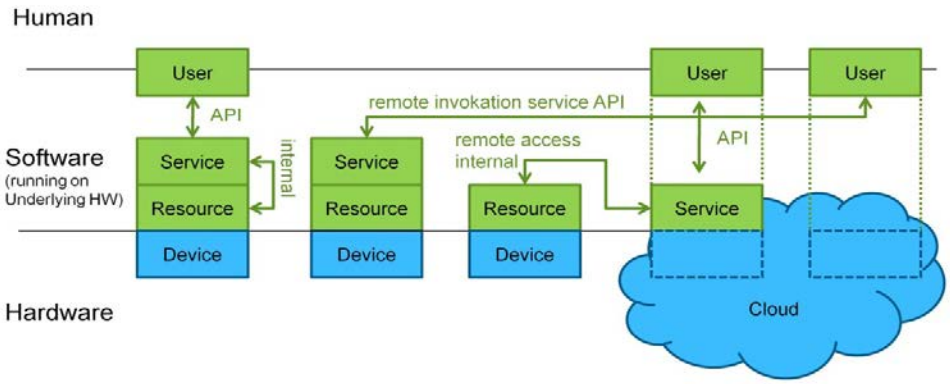

Figure 32: Various deployment configurations of devices, resources, and services.

Network-based resources are not shown in this Figure, as they can be regarded as being hidden behind cloud-based services.

5.2.1.2 Modelling of non-IoT-specific aspects

It is important to understand that the IoT Domain Model is not attempting to be a domain model for all types of ICT systems. Rather, it focuses on the IoT-specific parts. When modelling a complete system, many of the aspects to be covered that are not IoT-specific. For these aspects, the IoT Domain Model will provide only little help.

There are, however, two main concepts in the IoT Domain Model that provide a link to general I Services nowadays form the basis for many ICT systems and applications. When modelling a system, it is thus natural to include both IoT-related services as well as other services. The IoT Domain Model doesn't make a distinction between the different types of services as such a distinction is not a core IoT issue.

5.2.1.3 Identifiers and addresses

Identifiers and addresses are logically two different concepts, which unfortunately however are often confused in practice, in particular in the discussions about IoT [26]. While in some cases

the address might be used in the role of an identifier, it is important to distinguish between these terms.

Identifiers are used to identify something, for example a Physical Entity. In this case, the identifier is an attribute of the related Virtual Entity. Examples include EPCs and uIDs.

Addresses, on the other hand, are means for locating, accessing or communication with something, e.g., a service or a device. Addresses manifest themselves as attributes of the corresponding concepts, i.e., attributes of a service or a device. Examples include IPv6 or MAC addresses.

5.2.1.4 Granularity of concepts

In the IoT Domain Model, concepts like device, resource, and user have specialisations. Pertinent examples for devices are sensors and actuators. When modelling a concrete scenario, one can use either the general concepts or their specialisations, the IoT Domain Model doesn't prescribe anything. For example, instead of using a concrete concept like sensor it is also possible to use a more general concept like device. However, the specialisations are more precise and are therefore preferable where they apply. In other words, if at the time of modelling it is not (yet) clear what type of device is used, then just use device.

Modelling Rule 1	Model as precisely as possible at the time of modelling. Use more concrete, more fine-granular concepts and entities whenever possible.

5.2.1.5 Common patterns

Augmented Entities

As described in Section 3.2.2.2, Augmented Entities are the composition of a Physical Entity with its related Virtual Entity. In many cases though, the Augmented Entity is of little practical relevance and will have no concrete instantiation, as the example in Figure 33 shows. In this figure, a typical pattern is shown for how Physical Entities are mapped to data base records: In a data base of assets (a Network Resource in terms of the IoT Domain Model), a data base record (Virtual Entity, and also a Passive Digital Artefact) is stored for every building (Physical Entity).

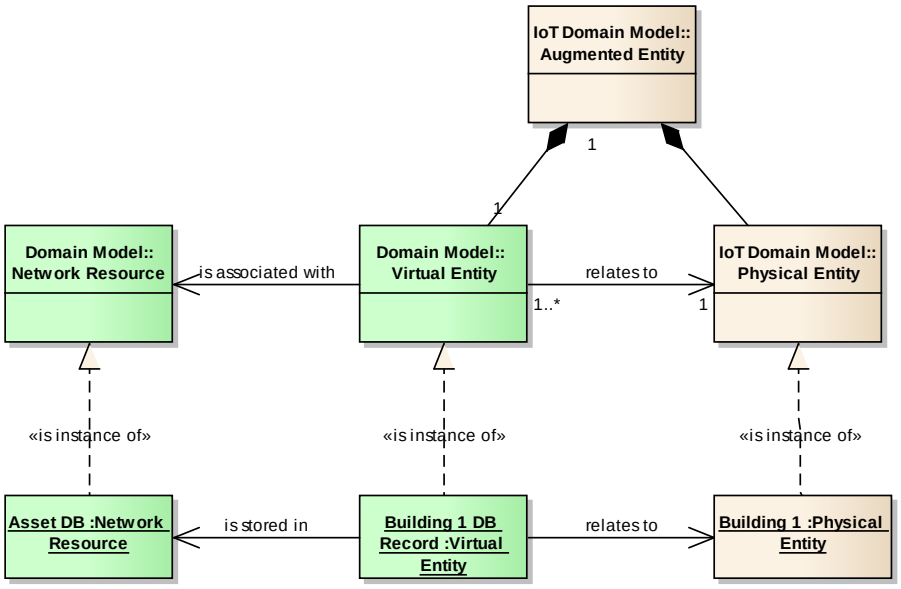

Figure 33: Data-base pattern as an example for an Augmented Entity.

Modelling Rule 2	The Virtual Entity for a given Physical Entity can be a data base record stored in a Network Resource.

A different case is truly smart objects, i.e., intelligent devices that have embedded logic seemingly able to act autonomously. In this case, the Augmented Entity is the smart object itself, and the associated Virtual Entity is an Active Digital Artefact, namely, the embedded logic (e.g., the software agent).

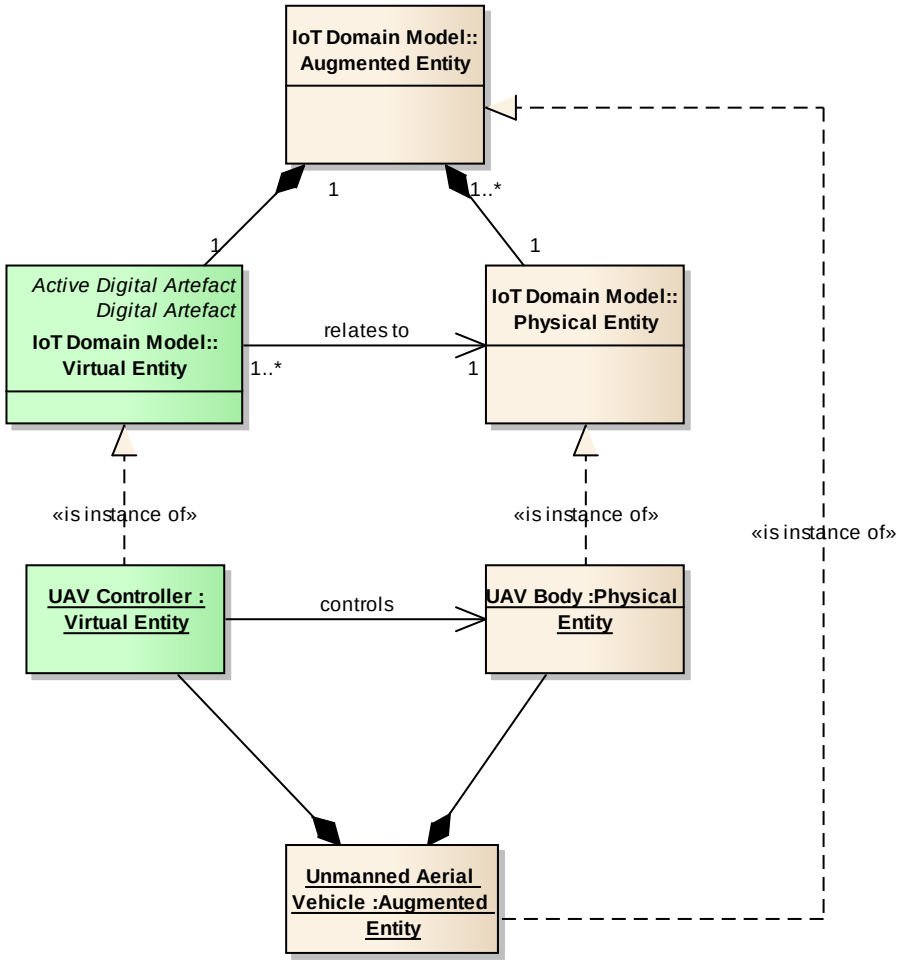

Figure 34: Smart-object pattern. UAV: unmanned aerial vehicle.

Figure 34 shows an example of a smart object: an unmanned aerial vehicle (UAV). The body of the UAV can be considered the Physical Entity, while the UAV controller is the related Virtual Entity. Together they form the Augmented Entity, the smart object.

Modelling Rule 3	When modelling an autonomous object, an Augmented Entity is used, consisting of a physical embodiment (the device) and its software controller (Virtual Entity).

Finally, the question often arises if something should be modelled as a Physical Entity or not. While possibly every real-world object could be modelled as a Physical Entity, this doesn't make sense. Not every sand corn needs to be represented in an IoT system. Hence we can deduce:

Modelling Rule 4	Only model something as a Physical Entity if it is clear what the associated Virtual Entity is and in what form it will manifest itself in the pertinent system.

Multiple Virtual Entities

In order to understand the case of multiple Virtual Entities, we take the example of a customer buying a new car. The customer visits the exhibition of an automobile manufacturing company and buys a new car. He then registers it under her name at the department of motor vehicles. In order to protect himself from unexpected financial expenses resulting from traffic collisions, he decides to buy car insurance. In this small scenario we notice that the same car, which is the Physical Entity, is registered at three stakeholders: The manufacturer, the vehicle-registration department, and the insurance company. As depicted in Figure 35 each of the three stakeholders maintains a unique entry in his data base identifying the car. These entries are multiple Virtual Entities for the same car.

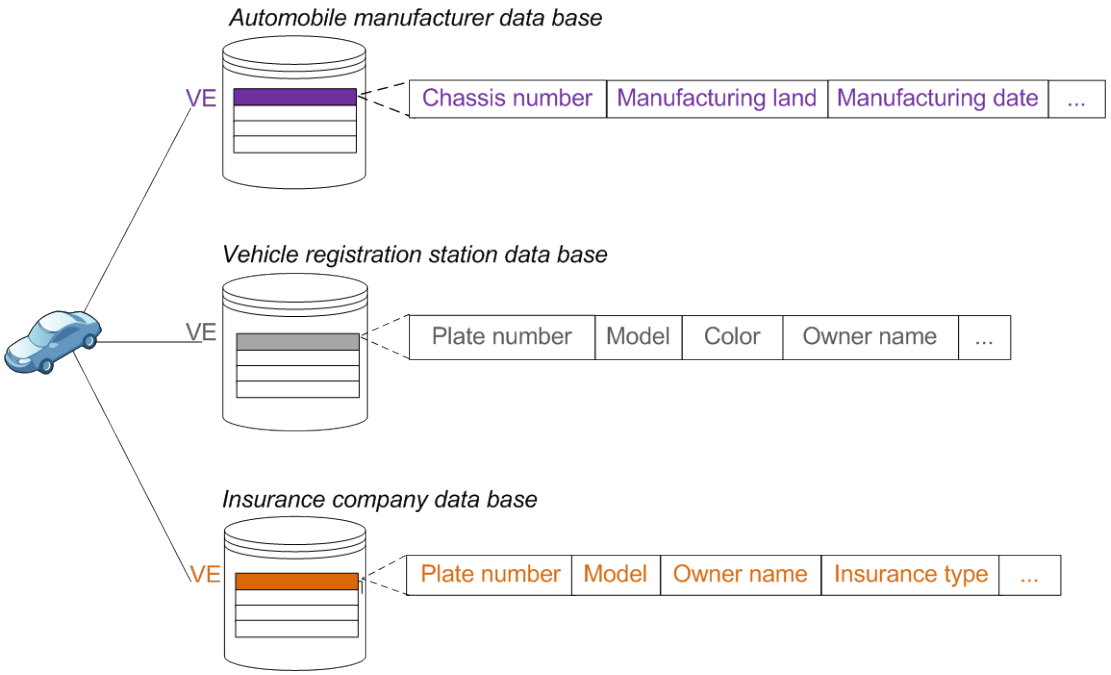

Figure 35: Multiple Virtual Entities (data-base entries) for a single car.

In practice, the number of Virtual Entities depends on the systems and domains, where the Physical Entity is represented and of course also what stakeholders are involved. We note that the characteristics of the Physical Entity change and, therefore, many of the Virtual Entities need to be maintained and kept up-to-date. Notice that the IoT Domain Model does not explicitly spell out any requirements on the maintenance of single and multiple Virtual Entities.

Smart phones

Smart phones are a very common element in many IoT-related scenarios. They are on the one hand devices containing a multitude of sensors, but they also host apps (Active Digital Artefacts), services, and resources. Figure 36 shows this in exemplary fashion: John's smart phone is used as a device to track the location of John, its owner. The GPS sensor is embedded in the phone itself. It is thus embedded sensor hardware. Its data is made accessible through the related On-Device Resource and the location service that exposes it. An app can be used to display the location information.

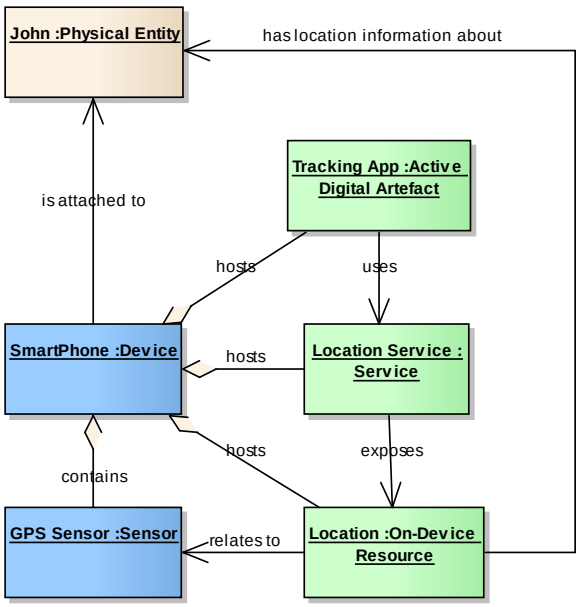

Figure 36: Exemplary modelling of a smart phone that is used as tracking device.

Note that in this example both the service as well as the application is shown to be hosted on the phone itself. While this depicts the most common case, other instantiations are possible.

Simple interactions

The IoT paradigm can be seen as a digital way to carry out interactions between the digital and the physical world. These interactions can be initiated either in the digital world or in the physical world.

In the first case, it is usually a *user* that needs to access a resource exposed through a service in order to attain a given goal. Such goals may range from observing a Physical Entity by using a sensor to modifying its state by leveraging an actuator device.

Notice that a user can invoke a service for getting some data or initiating some actuation, as well as for subscribing to certain events. After subscription, the resource (e.g., on a device) will detect the events of interest according to the specification provided by the user. The service providing access to the resource will then forward the event to the interested user. In an

alternative implementation, the service implementation is performing the event detection by processing all the raw data from the resource.

M2M interaction

Machine-to-Machine (M2M) communication is a technological approach for enabling meaningful information exchange between networked machines which show a certain degree of smartness. The term machine is generally related to an autonomous application while the smartness is related to the capability of controlling its own behaviour and communicating. This reflects the capability of making decisions on the basis of information retrieved from outside the system and being able to receive and execute commands. This approach is very relevant to the IoT and a specific definition of IoT Machine can be provided. In the terms of the IoT domain model, we define an IoT Machine as the union of:

- an Augmented Entity whose Virtual Entity component is an Active Digital Artefact. In this way, it can start interactions (being a User, it can invoke Services) and can control the behaviour of the machine
- one or more Resources and the underlying Devices which are used by the Active Digital Artefact to monitor/control the Physical Entity. Note that, because Resources are internal functionalities and the Active Digital Artefact is generally co-located on the same hardware, the interaction can happen even without the use of Services.
- the Services that are used for exposing Resources

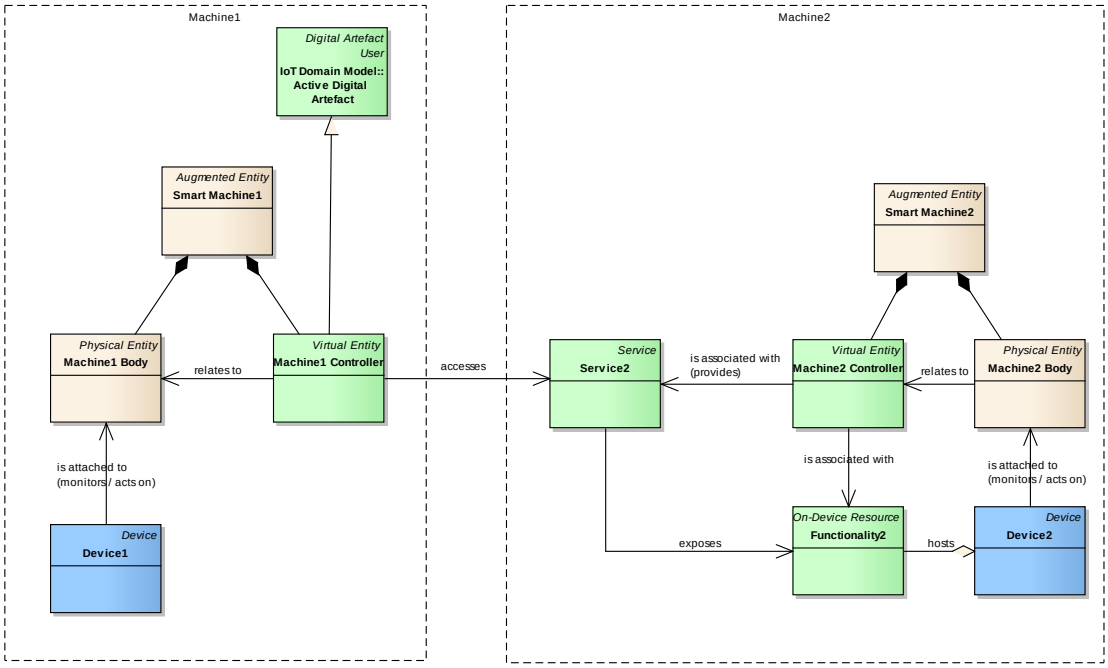

Figure 37: Domain model instantiation for a M2M communication scenario

An example is given Figure 37, where IoT Machine1 (for example an incoming car) needs information from IoT Machine2 (an automated barrier operator) in order to speed the passage through the barrier. The Machine1 Controller, an instantiation of an Active-Digital-Artefact

Virtual-Entity, will access as a User (Active-Digital-Artefact can be Users) Service2 and will require the activation of the barrier. Service2 provides access to Functionality2 (Resource) related to Machine2 and thus, by accessing Service2, the car can retrieve the information about the barrier status which is needed in turn to decide whether it needs to slow down or can pass through without danger. Another example of a smart machine is given in Section 5.2.1.5 on Augmented Entities.

As M2M is about the communication-based interaction between machines, it is important to clarify that IoT Machines can also interact with non-IoT Machines. For example, an IoT-Machine could need certain information provided by an autonomous web application, a non-IoT Machine, in order to make decisions.

However, as the controlling program of Machine 1 is a user, it can also communicate with other Machines by calling appropriate embedded services on another Machine, as shown in a simplified way in Figure 38.

Figure 38: M2M communication.

RFID gate in logistics

The term “Internet of Things” was originally coined by the MIT Auto-ID Centre around 1999 [39], the precursor to what is now known as EPCglobal. It is therefore worthwhile to map one of the most common scenarios of EPCglobal to the IoT Domain Model: The tracking of goods – pallets, cases, etc. – throughout the supply chain, from the manufacturer via distribution centres to the retail stores.

A first thing to note is that we often have a hierarchy of Physical Entities and the related Virtual Entities. A large boxed pallet is identified by a shipping company as PE5 with its corresponding Virtual Entity VE5. As depicted in Figure 39, the large boxed pallet contains multiple other cases that are identified as (PE1, VE1), (PE2, VE2), (PE3, VE3), and (PE4, VE4).

We note that the granularity of identifying PEs contained in other PEs is not defined by the IoT Domain Model, since it intimately depends on the application. In this example, if the large box contains four boxes of similar goods, e.g., shoes, the interest of the shipping company usually stops at identifying PE5 and thus tracking it by using VE5. Now if each of the four boxes contains different goods, e.g., shoes, hats, earrings, and bags, it might be of interest for the shipping company to additionally identify the four boxes as PE1, PE2, PE3, and PE4. The aim behind this higher granularity is to facilitate the process of sorting out the goods after delivery by checking VE1, VE2, VE3, and VE4.

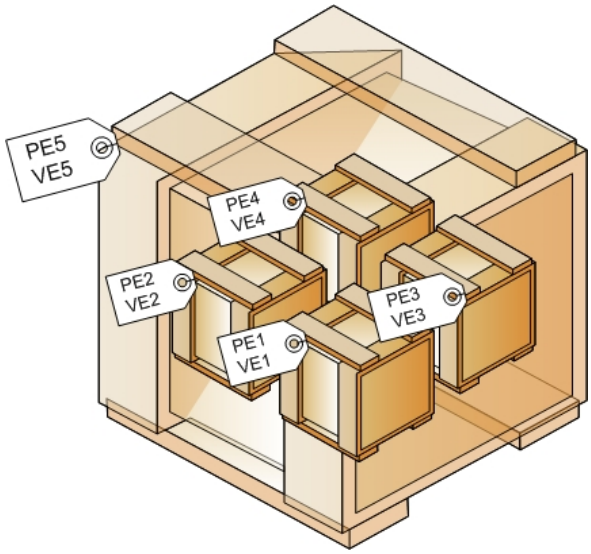

Figure 39: Shipping box containing multiple packets. The VE-to-PE mapping is exemplified by paper tags.

The result of the whole mapping of the RFID logistics scenario for only the pallet plus everything it contains, is depicted in Figure 39.

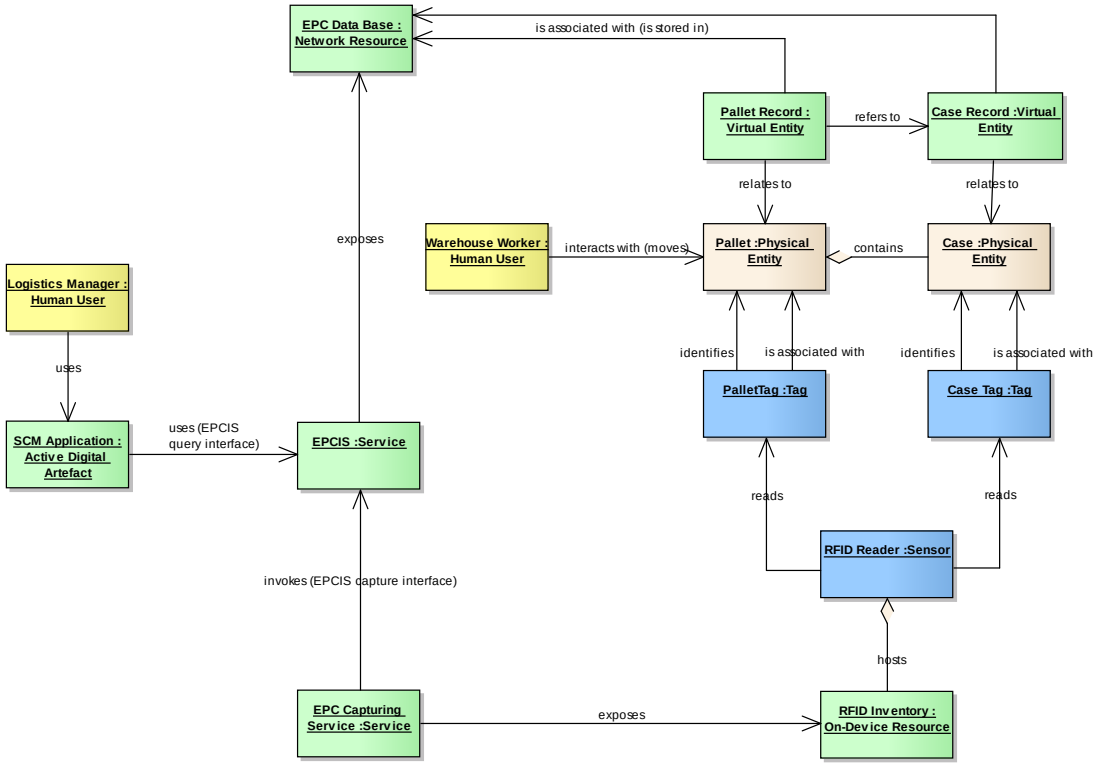

Figure 40: Domain modelling of a typical EPC-based RFID scenario (pallet containing cases).

In this example, the Virtual Entities take the form of data base records Figure 40) stored in a Network Resource, the EPC Data Base. This data base is exposed for querying and updating through the EPCIS service (EPC Information Service).

The logistics manager, a Human User, can use the SCM application in order to view the status of the tracked items (pallets and cases). The SCM application is invoking the EPCIS query interface in order to get the necessary data.

Both pallet and cases have RFID tags attached that identify them. A RFID reader – a type of sensor – reads the EPCs on the tags and hosts a resource that makes the RFID inventory data accessible. A special service, the EPC Capturing Service, is exposing this resource and is updating the EPC Data Base by invoking the EPCIS capture interface of the EPCIS service. The EPCIS capture interface and the EPCIS query interface are standardized and defined by EPCglobal [29].

Finally note that also physical interactions with the pallet can take place: a warehouse worker – a Human User – moves around the pallet.

5.2.2 Examples for IoT Domain Model objects

In this section we give examples on different objects in the IoT Domain Model. For each object we discuss a practical example and, where applicable, we highlight the dependency of the object on other objects and also provide some general information.

5.2.2.1 User

Application

- **Example:** A WSN installed in a wine cave monitors environmental factors such as temperature, humidity, and light intensity. These factors play an essential role in defining the quality of the final wine product. Therefore, the winegrower has an intelligent application running on his smart phone. The application allows him to periodically visualize the status of the cave. In this example, the application is a user and the cave is a Physical Entity.

- **Note:** An application is one kind of Active Digital Artefact.

Human User

- **Example:** The employee in a supermarket loads the fridge with meat instead of cheese. Therefore, he regulates the temperature of the fridge accordingly. In this example, the employee is a Human User and the fridge is a Physical Entity.

- **Note:** The case of multiple Human Users for one Physical Entity is possible as well. We take the example of the safe in a bank. For security reasons, more than one high-ranked employee is required to identify themselves simultaneously at the safe in order to be able to open it. In this example the eligible employees are Human Users and the safe is the Physical Entity.

5.2.2.2 Physical Entity

Environment

- **Example:** An optical fog sensor measures the density of water particles in the air that limit visibility. This sensor is used for traffic-control purposes, where it is often installed on the side of roads for monitoring visibility impairment through fog. The information

about the fog is sent to a traffic management system where it is analyzed. In this example the near surrounding above the road is the Physical Entity.

Living being

- **Example:** A WSN for agricultural monitoring. The network targets to report on the growth of fruits. To this end growth monitors are deployed. They are equipped with fruit-growth sensors as depicted in Figure 41. In this example, the fruits are Physical Entities that are living beings.

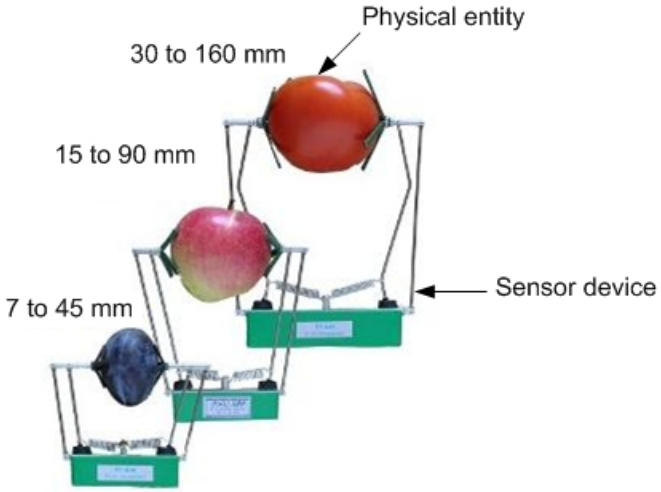

Figure 41: Growth fruit sensor [3].

Structural Asset

- **Example**: Equipping bridges with electrochemical fatigue sensors that reveal flaws in metal [41]. This works much the same way as an electrocardiogram tests the human heart. First, bridge inspectors identify parts of the bridge that are more susceptible to cracks. Second, they equip these areas with electrochemical fatigue sensors. Third, they apply a constant electrical current that runs between the sensors and the bridge. By monitoring the amplitude of the current passing through the metal, sensors can detect cracks. In this example, a susceptible area of the bridge is a structural-asset Physical Entity.

5.2.2.3 Resource

We explain the two examples for resources, one an On-Device Resource and the other a Network Resource.

On-Device Resource

- **Example:** TinyOS is an event-based OS for embedded networked sensors [42]. TinyOS provides predefined software components that manage the access and control of i.e., local LEDs, radio, or sensors. In this example, the software components are On-Device Resources.

Network Resource

- **Example:** HBase is an open-source, distributed, column-oriented database [43]. HBase offers a set of functionalities that allow the management of distributed information. In this example the HBase software libraries and components are -Network Resources.

5.2.2.4 Service

Interacting services

- **Example:** A system for home-patient monitoring. The system is composed of a body sensor network (BSN) attached to the body of the patient. Bioelectric chips monitor the status of the patient and require no direct involvement from a human being. As depicted in Figure 42, the intelligence of the system resides not only in the hardware but also in three main services. First, the BSN monitoring service that evaluates the readings of the bioelectric chips i.e., a blood pressure. Second, the automatic service call, which alerts the relatives of the patient whenever his situation deteriorates. Third, another automatic service call that alerts the ambulance. The diagram in Figure 42 shows the conditions to be fulfilled for one service to invoke another service.

- **Note:** A service demanding high processing and storage capabilities can be divided into multiple subservices running on different machines that invoke each other. In comparison to the original service, each of these subservices requires less storage and processing capabilities. Therefore, a trade-off exists between the number of subservices and the power consumption of the hosting machines. Distributed subservices induce an inter-communication overhead that increases the power-consumption of the hosting machines. This trade-off should be taken into consideration when dealing with low-power communicating devices [44].

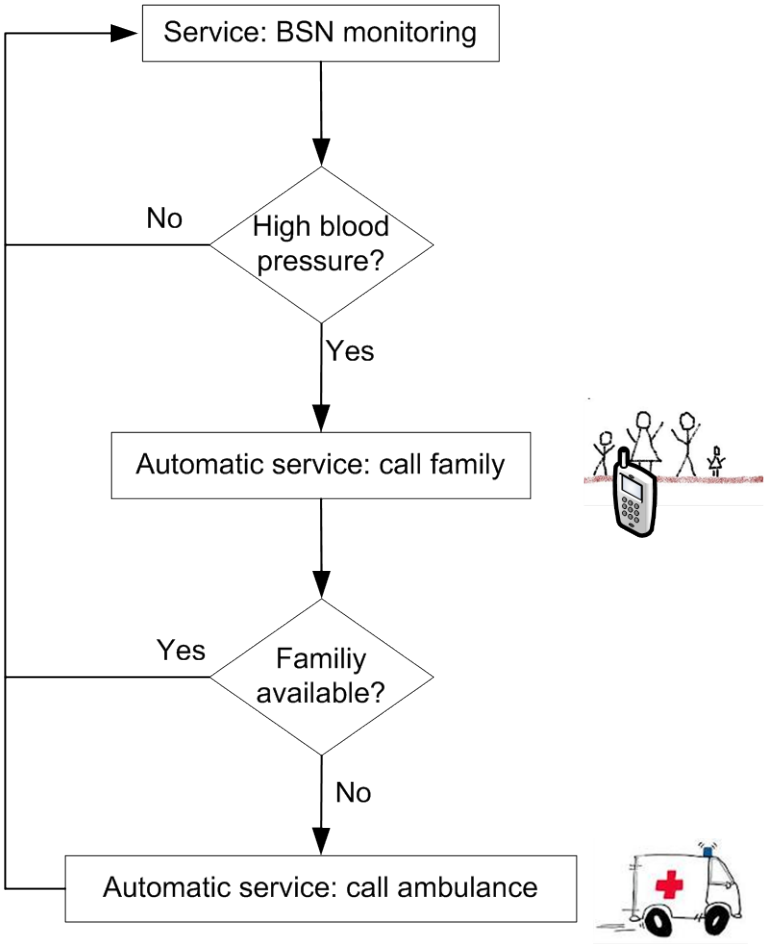

Figure 42: Interacting services for a home-patient monitoring scenario.

Service associated with a Virtual Entity

- **Example:** Services can be associated with Virtual Entities and these associations are stored and can be discovered in the IoT system. The management of these associations can be handled in a centralized database or in a highly distributed fashion as in a peer-to-peer system, depending on the characteristics of the underlying system.

Service accessing a resource

- **Example:** A service for monitoring air pollution. Sensor nodes are semi-randomly distributed in a city and measure the percentage of CO in the air. A remote server runs software that periodically queries the readings from the sensor nodes, analyses the readings, and monitors the evolution of the air pollution. In this example, the monitoring software is a service that accesses multiple resources. The latter are the components and functions running on sensor nodes, and these components allow operations such as reading from the sensors.

5.2.2.5 Device

Hierarchical devices

- **Example:** As depicted in Figure 43, a Telos node contains three types of integrated sensors (photodiode, humidity and temperature), several expansion pins to mount external sensors, and three integrated LEDs [44]. Two views of the node exist: The node as a whole may be seen as a single device or it can be seen as a composition of multiple sensors and actuators acting as individual devices.

- **Note:** A device can be seen as a single unit as well as a composition of multiple devices. This granularity of modulating a device is not specified in the IoT Domain Model due to the fact that it is application dependent.

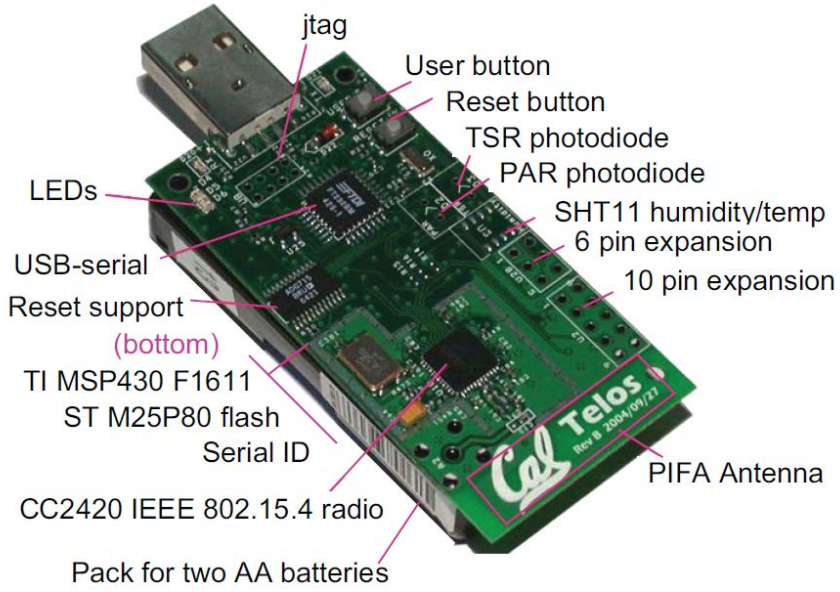

Figure 43: Telos ultra-low power wireless module.

5.3 Usage of the IoT Reference Architecture

The IoT Reference Architecture consists of views, which define core architectural aspects that need to be taken into account, and perspectives, which basically are quality aspects spanning across the views.

5.3.1 Design choices

For each of the four defined views (functional, information, deployment and operational view), a table as shown below will give the designer a number of design choices, and will list what the impact of the particular choice on the aspects of the different perspectives is.

The table contains a topic that addresses an architectural problem to which the respective design choice provides a suggested solution for. To assist architects with their selection of the most suitable design choice the impact on four perspectives is assessed in the table. The symbols in the 'Impact on'-column indicate the impact seen from Security & Privacy (S&P)-, Performance & Scalability-, Availability & Resilience-, as well as Evolution & Interoperability-perspectives when the respective design choice is applied:

+: has a positive impact onto the perspective, e.g. increases security or contributes to scalability

-: has a negative impact onto the perspective, e.g. increases the risk for security leaks

+/-: has positive and negative impact onto the perspective

0: has no impact onto the overall architecture

5.3.1.1 Functional view

Table 7: Design choices functional view

Topic	Design Choice	Impact on			
		Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
IoT Business Process Management / Application support	DC1.1 Business Process Modelling according to BPMN 2.0	+/-	+	+	+
	DC1.2 Business Process Execution by BPMN 2.0 execution engine	+/-	+	+	+
Service Organisation	DC2.1 Service Orchestration with mandatory security	+/-	0	+	0
	DC2.2 Service Orchestration with optional security	-	0	-	0
VE Resolution	DC3.1 VE Resolution with mandatory security	+/-	0	+	0
	DC3.2 VE Resolution with optional security	-	0	-	0
	DC3.3 VE Resolution with QoS	0	0	+	0
	DC3.4 VE Resolution domain-oriented	+	+	+	+
	DC3.5 VE Resolution location-oriented	-	+	+/-	+/-
	DC3.6 Resolution Semantic Web-oriented	0	0	+	+/-
	DC3.7 Resolution Peer-to-Peer-oriented	0	+	+	0
	DC3.8 Resolution Federation-based	0	+	+	0
VE & IoT Service Monitoring	DC3.9 VE & IoT Service Monitoring with mandatory security	+/-	0	+	0

	DC3.10 VE & IoT Service Monitoring with optional security	-	0	-	0
IoT Service Resolution	DC4.1 IoT Service Resolution with mandatory security	+/-	0	+	0
	DC4.2 IoT Service Resolution with optional security	-	0	-	0
IoT Service	DC4.3 IoT Service with mandatory security	+/-	0	+	0
	DC4.4 IoT Service with optional security	-	0	-	0
Identification and Authenticatio n	DC.51 Identifier based identification	-	+	+/-	0
	DC5.2 Crypto -based authentication over open channel	+/-	-	+/-	-
	DC5.3 Authentication over encrypted channel	+	-	+	-
Service Access Control	DC6.2 Unrestricted access to service	-	+	-	+
	DC6.2 Authentication based service access	+/-	+/-	+/-	+/-
	DC6.3 Policy-based service access	+	+/-	+/-	-
Sharing Public Keys/Certific ation	DC7.1 Manually/Out of Band shared Public Keys	0	+/-	0	-
	DC7.2 Leap-of-faith shared Public keys	-	0	0	-
	DC7.3 Certificate Authority	+	-	-	+
	DC7.4 Web of Trust	+	+/-	0	+/-
Public Keys exchange format	DC8.2 Raw Public Keys	0	+	0	-
	DC8.2 Explicit Certificate	0	-	-	+
	DC8.3 Implicit Certificate	0	+	0	+/-
Communicati on Confidentialit y	DC9.1 No encryption	-	+	0	+
	DC9.2 End-to-end Encryption	+	-	0	-
	DC9.3 Hop-to-hop Encryption	-	-	0	0
	DC9.4 Onion routing-like encryption	+	-	0	-
	DC9.5 Tunnelling	+	-	+/-	+/-
Identity	DC10.1 Local Identity	0	+	0	-

Scope	DC10.2 Global Identity	-	0	0	+/-

IoT business process management

Business process modelling according to BPMN 2.0 standard (DC1.1)

By using BPMN 2.0 conformant process modelling tools a smooth integration of IoT Services with enterprise level applications is supported. The expression of Security & Privacy (S&P) aspects is limited in standard BPMN 2.0; respective extensions have been proposed by [45]. BPMN 2.0 allows modelling applications without assigning particular Virtual Entities and IoT Services at design time of the application which contributes to scalability. That means that applications can be designed without knowing the IoT Services available at runtime. Using BPMN 2.0 contributes to interoperability since it is a widely used standard for modelling enterprise-level applications.

Business process execution by BPMN 2.0 execution engine (DC2.2)

BPMN 2.0 execution engines are able to execute the enterprise level applications modelled according to the widely accepted standard BPMN 2.0 that ensures interoperability with existing business processes. S&P policies modelled in BPMN 2.0 must be also supported by the underlying Functional Groups; otherwise the policies cannot be guaranteed. BPMN 2.0 execution engines are able to process applications modelled with Virtual Entities and IoT Services that are available at execution time.

Service organisation

Service management means resolution of suitable services and allocating them to the user's service request and the resolution of services that are needed to compose higher-level services.

Service Orchestration with mandatory security (DC2.1)

An orchestration engine enforces S&P policies during allocation of IoT Services and setting up service compositions. This requires that Virtual Entity Resolution and IoT Service Resolution must support S&P policies as well. Only those services are orchestrated that are aligned with the S&P policies.

Service Orchestration with optional security (DC2.2)

An orchestration engine can be configured to enforce security and privacy policies during allocation of IoT Services and setting up service compositions. It is left to the application if S&P policies have to be applied. S&P policies can only be enforced though if they are supported by the VE Resolution and IoT Service Resolution used during orchestration. S&P policies on Virtual Enities and IoT Services can be bypassed if security is not enforced.

Virtual Entity

The Virtual Entity topic covers all functions that concern the Virtual Entity handling within IoT-A such as resolution, monitoring, and storage of history. The so called Entity Services are also handled here. Due to the large amount of data stored in the VE Resolution framework distributed architectures are recommended here in order to provide scalability, availability and reliability. [34] suggests five approaches to structure the distributed search space in the VE Resolution framework: Domain-oriented and Location-oriented.

Virtual Entity Resolution with mandatory security (DC3.1)

During the resolution security and privacy policies must be applied. This requires having defined the policies on the Virtual Entity-associations for each association in the resolution framework. Furthermore it is required that the IoT Service Resolution used by the Virtual Entity Resolution must support S&P policies as well. If so only those services are provided that are aligned with the S&P policies.

Virtual Entity Resolution with optional security (DC3.2)

During the resolution security and privacy policies can be applied. This allows resolution of Virtual Entity-associations with no particular policies defined on them in the resolution framework. S&P policies can only be enforced if they are supported by IoT Service Resolution. S&P policies on Virtual Entities can be bypassed if security is not enforced.

Virtual Entity Resolution with QoS (DC3.3)

This allows filtering of Virtual Entity-associations returned upon discovery request that match given QoS criteria defined. This gives users more options to specify suitable Virtual Entities since only those services are provided that meet the QoS requirements.

Virtual Entity Resolution domain-oriented (DC3.4)

Domain-oriented VE Resolution approach organises the resolution framework in hierarchically organised domains similar to Domain Name System (DNS). The hierarchy is build according to the hierarchy of things captured by Virtual Entities from higher granularity to lower granularity, e.g. country → city → district → building → room. The resolution framework performs faster through divided search space; its complexity is of *O(log n)* in best case, and *O(n)* in worst case. Load balancing is supported through replication, and a Resource can be member of different domains at a time. Fault tolerance is supported through distribution and redundancy; the framework evolves with the number of things connected.

Virtual Entity Resolution location-oriented (DC3.5)

A resolution server (RS) is responsible for indexing all connected things in a certain geographical area, called indexing scope. A Catalogue server then creates the Catalogue Index of every RS' indexing scope. A resolution request is redirected towards the RS whose indexing scope intersects the search scope of the request. Large scale IoT systems are expected to have multiple administrative domains that must be handled by a federated resolution infrastructure. Different domains interact with each other by the means of a central domain directory or domain catalogue. Communication between framework domains needs to be secured. The framework performs faster through a divided search space. Indexing scope can be adjusted according to usage load. The framework scales by adding more RSs. With this approach it is impossible to retrieve things based on identifiers. Fault tolerance is achieved through data distribution and index data replication. The central domain directory is potential single point of failure. There is no theoretical limit on indexed things, but indexing scope is bound to geographic location. Only a limited number of things can share the same location.

Virtual Entity Resolution Semantic Web-oriented (DC3.6)

Semantic Web technologies are used to annotate Virtual Entity descriptions in a way machines can interpret them. This overcomes the need for exact syntactic matchmaking between resolution request and search terms in the resolution infrastructure. The search space of the resolution infrastructure is indexed by an unsupervised machine-learning technique and clustered through latent factors derived from the learning. This design is independent from the deployment of the resolution infrastructure. Distribution and replication is supported by this approach, but depends on implementation on how it is done. Semantic interoperability is

achieved through shared ontologies, after extending ontologies the training model needs to be updated.

Virtual Entity Resolution Peer-to-Peer-oriented (DC3.7)

A peer-to-peer infrastructure will maintain no centralised servers, all data is distributed in the network along with sophisticated retrieval and routing mechanisms. There are several approaches on how to distribute the data (pure, centralised indexing server, distributed hash tables). The latter approach is the recommended one for IoT Resolution infrastructures. Resolution requests result in traffic complexity of *O(n)* in worst case and *O(log n)* in best case. The framework is stable and robust through distribution and redundancy.

Virtual Entity Resolution Federation-based (DC3.8)

In a federated infrastructure Virtual Entities are clustered based on similarity. Dedicated places are in charge of the IoT Services they offer and provide their descriptions as part of a distributed resolution framework. The framework is scalable and fault tolerant because of distribution.

Virtual Entity & IoT Service Monitoring with mandatory security (DC3.9)

The VE & IoT Service Monitoring functional component is responsible for automatically finding new associations, which are then inserted into the VE resolution functional component, where it is stored in the associations storage. New associations can be derived based on existing associations, service descriptions and information about Virtual Entities [46]. If security is enforced the S&P policies of Virtual Entity need to match the policies of IoT Service to establish an association that is again compliant to the S&P policies. Both VE Resolution and IoT Service Resolution have to support S&P policies. In this case only those associations are established that are aligned with the S&P policies determined by Virtual Entity and IoT Service.

Virtual Entity & IoT Service Monitoring with optional security (DC3.10)

It cannot be guaranteed that an established association is compliant to the S&P policies of Virtual Entity or IoT Service if security is not enforced. S&P policies can only be enforced if they are supported by Virtual Entity and IoT Service Resolution.

IoT Service Resolution

The IoT Service Resolution framework offers lookup and discovery functions for IoT Services. The same design choices as in VE Resolution can be applied to IoT Resolution [34].

IoT Service Resolution with mandatory security (DC4.1)

The IoT Service Resolution will discover only those services that are compliant to the S&P policies determined in its service description. For doing so S&P policies must be supported by IoT Services as well.

IoT Service Resolution with optional security (DC4.2)

The IoT Service Resolution will discover all services regardless of the IoT Services' S&P policies unless security is enforced. Also unsecure services are provided if security is not enforced. S&P policies can only be enforced though if they are supported by IoT Services.

IoT Service

IoT Service with mandatory security (DC4.3)

The IoT Service provides an interface to IoT users by utilising capabilities of IoT Resources. The software running on the Resource implementing the IoT Service must apply the S&P policies

described in its service description. If so only those IoT Services are provided that implement the respective S&P policies.

IoT Service with optional security (DC4.4)

The IoT Service provides an interface to IoT users by utilising capabilities of IoT Resources. It cannot be assured that the software running on the Resource implementing the IoT Service applies any S&P policies if security is not enforced. S&P policies can only be enforced if they are implemented on IoT Resources.

Identification and Authentication

In the frame of the IoT, subjects must be identified for functional reasons. In some cases, after cautious investigation, services might be designed for anonymous access if this doesn't raise privacy and safety issues. When identification is needed though, depending on the reason for which it is needed, authentication might be required to exclude that malicious users impersonate other subjects.

Identifier based identification (DC5.1)

In this case, the identifier is sent in clear over the communication channel for identification purposes. This solution must never be used as basis for authorization as these messages can easily be replayed or even faked. It could be implemented when the identification is needed for the customization of the service. Even in this case, the fact that the identity could be forged has to be taken into account. For example a malicious user could feign to be user A in order to gain knowledge about user A's customization settings. This solution has the advantage of consuming only a reduced amount of battery power.

Crypto -based authentication over open channel (DC5.2)

In this scenario, the party requesting authentication initiates a challenge-response process in which the subject to be authenticated answers with a signed version of the challenge. The fact that the communication is performed on an open channel will not affect the authentication process itself, as a malicious user would not be able to derive the secret cryptographic material used for the authentication. This solution unfortunately is subjects to replay and similar attacks, aimed at (where applicable) depleting the battery resources of the device hosting the authenticated party. Man in the middle attacks would theoretically be possible albeit not effective. Depending on the implementation, the identity (or pseudonym) of the user might also be provided in clear, in which case an overhearing entity might learn who the user is and about his habits. Usually this kind of solution needs the support of an Authentication infrastructure functional component.

Authentication over encrypted channel (DC5.3)

This is the most secure of the solutions where even the content of the message exchanged between the two parties in encrypted. In this scenario though a great amount of battery is used as cryptographic functions must be applied both for the authentication process itself and for the encryption of the communication.

Service Access Control

Service Access Control means providing control over which user can interact with a given service.

Unrestricted access to service (DC6.1)

This scenario presents no architectural requirements. It is taken as reference for what concerns communication, battery performance and availability in DC6. No interoperability issues due to authentication and authorization.

On the downside, privacy issues might rise and resilience could suffer due to malicious behaviour of not authenticated consumer.

Authentication based service access (DC6.2)

This choice requires the implementation of Authentication component.

Communication performance slightly decreased. Life-time of battery-powered nodes decreased. Scalability issues regarding the number of authenticable users.

Authentication and authorization process carried out on constrained nodes can lead to unavailability. Unconstrained nodes are not affected otherwise.

Requestor and provider endpoints should both support at least one common Identity Framework OR the Authentication implementation component should provide mediation.

Policy-based service access (DC6.3)

This choice requires the implementation of Authentication and Authorization components. Communication performance might be affected negatively as well as the life-time of battery-powered nodes. Scalability issues in common with DC6.2

The availability of highly requested services to authorized nodes in peak moments can result increased. Resilience generally increased.

Requestor and provider endpoints should both support at least one common Identity Framework OR the Authentication implementation component should provide mediation. The Authorization component implementation used by service provider should store requestor's access right OR be able to retrieve them from federated components.

Capability-based service access (DC6.4)

Authentication component is not a requirement for Authorization, while each node must be able to perform capability validation. Capabilities provisioning falls into the problem of Key Exchange. No Privacy issues arise. Communication performance will result slightly decreased as well as the life-time of battery-powered nodes. Scalability is not affected by the number of users.

Can result in increased availability of highly requested services to authorized nodes in peak moments. Resilience generally increased compared to DC6.1.

As authentication based service access, but also the Authorization component used by service provider should store client's access right OR be able to retrieve them.

Sharing Public Keys/Certification

Sharing Public Keys is needed in case of asymmetric cryptography to initiate a communication, to send secure messages or to validate an identity (DC5.2). Generally it is a mechanism to establish the authenticity of the binding between a public key and its owner.

Manually/Out of Band shared Public Keys (DC7.1)

The Public Key is shared before initializing the communication out of band.

S&P No concerns

P&S Faster bootstrap. Scalability is affected by the fact all public keys needs to be permanently stored in the device.

A&R No concerns

E&I Evolution is complicate, as much as the initial provisioning. No concerns regarding Interoperability.

Leap-of-faith shared Public keys (DC7.2)

The Public Key is shared during the first association with no prior trust.

S&P The bootstrap should be performed in a secure environment or the whole life cycle is endangered.

P&S No concerns

A&R No concerns

E&I No concerns regarding Interoperability. If the set of keys is changed a manual invalidation is required.

Certificate Authority (DC7.3)

S&P Certificate Authorities need to be trusted.

P&S It requires interaction with the Certificate Authorities.

A&R Additional point of failures.

E&I No concerns regarding Interoperability.

Web of Trust (DC7.4)

Web of Trust is a decentralised trust model to establish a web of peers that trust each other.

S&P: No concerns

P&S: Performance depends on the locality of the web of trust

A&R: No concerns

E&I: No concerns regarding Interoperability. Certificate revocation is not easy.

Public Keys exchange format

Public keys exchange format means in which format public keys are exchanged.

Raw Public Keys (DC8.1)

P&S: Very compressed format

E&I: The key format should be known beforehand and supported.

Explicit Certificate (DC8.2)

Explicit Certificates bundle identity, the public key and CA signature (which could be the owner itself).

P&S: Explicit Certificates tend to be verbose and redundant. In case of LLNetworks several packets are required to transport an explicit certificate.

E&I: Explicit Certificates are a widespread Interoperable form

Implicit Certificate (DC8.3)

Implicit Certificates embeds the signature and the public key in a length comparable to the one of the public key itself. Implicit certificates require a Certificate Authority (DCS3.3).

P&S: Implicit Certificates avoid being verbose. Faster operations required to have a validated key.

E&I: Implicit Certificates requires prior knowledge of Algorithms as well as a compliant Certificate Authority.

Communication Confidentiality

This section is concerned with the topic of securing communication channels against eavesdropping.

No encryption (DC9.1)

No encryption means the data flow is in clear text. This design choice offers no confidentiality but the best performances. That is used as a goodput benchmark.

End-to-end Encryption (DC9.2)

End-to-end encryption means the two communication endpoints are the only ones supposed to be able to decode the message. It requires the same algorithm and keys are shared and supported by both ends. The trade-off is between inability of constrained objects to support strong encryption and weak security.

Hop-to-hop Encryption (DC9.3)

Hop-to-hop encryption means that every hop (according to the conceptual communication layer we are referring to) decode and re-encode the message to the following hop. That means that hops have to share keys and support for protocols two by two. That means also that every hop knows the content of the message.

Onion routing-like encryption (DC9.4)

In this case every packet is re-encoded (without decoding) for every (arbitrary chosen) hop. It can introduce additional anonymity and can help to support stronger cryptography even in case of constrained nodes involved.

Tunnelling (DC9.5)

Secure network access provides a network connection or service access only if the device is authorized.

Secure communications functions include authenticating communicating peers, ensuring confidentiality and integrity of communicated data, preventing repudiation of a communication transaction, and protecting the identity of communicating entities.

Secure storage mandates confidentiality and integrity of sensitive information stored in the system.

Content security enforces the usage restrictions of the digital content stored or accessed by the system.

Availability ensures that the system can perform its intended function and service legitimate users at all times, without being disrupted by denial-of-service attacks.

Identity Scope

Local Identity (DC10.1)

A local identity is scoped and valid only among a restricted party.

This increase performances for validation and ease the support.

Interoperability is negatively affected by the reduced scope of the Identity and overcoming this limitation would require additional solution, among them identity mapping.

Global Identity (DC10.2)

A global identity is scoped in the whole Internet. This can introduce infrastructure burdens and exploits. This can also introduce privacy leaks.

Interoperability is affected positively, identities being under the same global namespace. In the other hand Extensibility has some drawbacks, for instance the propagation of identity updates.

Scalability is not a real concern, given a number of solutions in the literature, but still is suitable to pay attention to it.

5.3.1.2 Information view

This section presents design choices on how to specify the design of the IoT Information Model to be used in IoT architectures.

Table 8: Design choices information view

Topic	Design Choice	Impact on			
		Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
Storage of Information History	DC11.1 Storage of History locally	+	-	-	0
	DC11.2 Storage of History remotely	+/-	+	+	0
	DC11.3 Storage of History locally and remotely	+/-	+	+	0
Implementation of	DC12.1 Implementation of Semantics in RDF	0	+/-	-	+
	DC12.2 Implementation of Semantics in OWL	0	-	-	+

semantics	DC12.3 Implementation of Semantics in RDFa	0	+	-	+

Storage of history

IoT architectures can store information that has been gathered before from IoT Resources to be used later for further processing. Another reason to provide the information history as cache for the IoT Resource is to avoid excessive use of IoT devices to minimise energy consumption on the constraint IoT devices. There are several choices on where to store the information history: locally, remotely, or a combination of both.

Storage of history locally (DC11.1)

The information history is stored on the IoT device that has produced the information. History needs to be secured in the same way as the present information. Constrained IoT devices are used every time the history is queried; the storage size of history as well as the performance is limited on IoT devices. Having a single storage place for history information is against good scalability. The availability of information history depends on the availability of the IoT device hosting the history. There is no impact on Evolution and Interoperability.

Storage of history remotely (DC11.2)

The information history is not stored on the IoT device that has produced the information, but on a different IoT Resource, to which the information is uploaded to. The additional history resource needs to be secured too with either the same S&P policies as the original IoT Resource or different policies. A history resource in the cloud can perform better than IoT devices; the replication of information allows load balancing between history and present information which contributes to better scalability. The Information history still exists when the respective IoT device becomes unavailable; fault tolerance is achieved by data replication. There is no impact on Evolution and Interoperability.

Storage of history locally and remotely (DC11.3)

The information history is stored on the IoT device that has produced the information as well as on a different IoT Resource replicating the information. History information that exceeds the capabilities of the hosting IoT device capabilities can be offloaded to high performance devices. This design choice contributes to high scalability as well as higher performance since the remotely stored history information is a replication of the locally stored information. Replicating information is cheaper to achieve by the device than retrieving ‘fresh’ information for every replication. This approach provides a reference-information on the local device that can be used in case a remotely stored history information resource fails. Thus this design choice leads to higher Availability and better Resilience. There is no impact on Evolution and Interoperability.

Implementation of semantics

This paragraph presents design choices on the description formats for semantic annotation of information. Presented are three choices that have been analysed in [53]

Implementation of semantics in RDF (DC12.1)

The Resource Description Framework (RDF) [47] provides a standard to express relationships between objects, like things or location, defined as URL’s. The RDF documents can be stored in ‘triple stores’. Those stores form databases designed for semantically structured data that can be queried by semantic query languages, like SPARQL [48]. When using RDF for information annotation URLs have to be accommodated in the information descriptions. Those URLs can be

very long and require some size of memory on the hosting device. Thus performance of restricted devices might be to low to handle RDF documents. In those cases the RDF documents should be handled by more powerful gateways in the network. Interoperability is achieved by using URLs as unique identifiers for Things. Those Thing-URLs have to be available over the network. If a device associated to a Thing is not able to handle the URLs of the Thing it is associated to a gateway has to translate the RDF-URLs to its respective device specific identifier. The translation has a negative impact on the performance, but putting the burden onto more powerful gateways will solve the problem and leads to better scalability if one gateway handles address translation for several devices. If semantic information is not replicated it will be unusable as soon as the device managing the device becomes unavailable. There is no impact on Security and Privacy.

Implementation of semantics in OWL (DC12.2)

The Web Ontology Language (OWL) [33] is a standard to express taxonomies of things in a descriptive language. OWL is based on RDF and XML. Like RDF documents, OWL documents can be stored and queried in triple stores too, but additionally OWL reasoning tools allow inferring additional knowledge based on the information that is asserted in the documents explicitly. Due to more expressiveness OWL documents can be bigger than RDF documents. Hence the impact on Performance is worse than in DC12.1. The impact in other perspectives is the same as in DC12.1.

Implementation of semantics in RDFa (DC12.3)

RDF-in-attributes (RDFa) [49] provides a way to inject semantic concepts into non-semantic documents. The URLs in the documents identify semantic concepts like Virtual Entities or its attributes, but save the syntactic overhead required in pure RDF/XML or OWL documents. Smaller documents allow constraint devices to handle the information onboard which results in higher performance. The impact in other perspectives is the same as in DC12.1.

5.3.1.3 Deployment and operation view

Table 9: Design choices deployment and operation view

Topic	Design Choice	Impact on			
		Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
Smart object connectivity	DC13.1 Sensor and actuator networks	-	+/-	+/-	+
	DC13.2 RFID and smart tags	+	-	-	+
	DC13.3 WiFi connectivity	+	+	+	+
	DC13.4 Cellular network connectivity	+	+/-	+	+
"Last mile" communication protocols	DC14.1 IoT-A protocol suite	+	+	0	+/-
	DC14.2 Ad hoc proprietary stack	+/-	+	0	-

	DC14.3 Other standards not in the IoT-A suite	+/-	+	0	-
Service hosting	DC15.1 on smart objects	+/-	+	+	0
	DC15.2 on gateways	+	-	+/-	0
	DC15.3 in the cloud	+	+/-	+	0
Service engine	DC16.1 internal	+	+	+/-	+
	DC16.2 external provider	+/-	+	+	+
Information storage	DC17.1 local only	+	-	+/-	0
	DC17.2 web only	-	+	+/-	0
	DC17.3 local and web cached	+/-	+/-	+/-	0

Smart object connectivity

Smart objects may have different functionalities and capabilities depending on the specific service they are designed to provide. It is important to choose the correct category of device in order to satisfy the requirements identified. This design choice concerns the connectivity adopted to integrate the smart objects into the IoT. We addressed in particular Sensor and Actuator Networks, RFID, WiFi, and cellular network.

Sensor & Actuator networks (DC13.1)

Sensor and Actuator Networks (S&AN), either wireless or wired, are usually characterized by low data rate, low power consumption, low cost and low processing power. Opting for such a connectivity type will limit the complexity of the service offered by the device, thus making unfeasible some of the advanced functionalities in all the categories. However, S&AN devices are mostly chosen for their small size and low cost, thus offering a lightweight solution for simple service implementations.

RFID and smart tags (DC13.2)

The most cost effective solution for object identification. Smartness is provided by the most advanced protocols only, such as NFC, which establish a two-way communication with the reader. This type of connectivity calls for an additional tier of network interconnecting the readers.

WiFi (DC13.3)

Whenever high performance is required, WiFi technology is the best solution. The downside relies on the higher power consumption. The more mature standards are an additional benefit of WiFi technology.

Cellular networks (DC13.4)

If the primary requirements for smart objects are the availability and the mobility, cellular networks technology offers the widest coverage among the connection type. However, it is the most expensive solution and has high energy consumption.

"Last mile" communication protocols

One of the key points of the IoT is the interoperability among the different smart objects. While in the Web the HTTP/IP paradigm is predominant, smart objects are usually realized using many different communication stack solutions. This is due to two main reasons: either the service needs to comply to regulation that force the implementers to choose a particular solution, or performance is preferred instead of interoperability, hence the communication is designed for the specific service only.

IoT-A protocol suite (DC14.1)

This project aims at providing guidelines and best practices in order for system providers and integrators to realize networks of smart objects, which are seamlessly interoperable and thus can form a unique IoT.

Ad hoc proprietary stack (DC14.2)

As anticipated above, when the performance is the first requirement, usually interoperability is more difficult and needs added complexity in the form of gateways and communication wrappers.

Other standards not in the IoT-A suite (DC14.3)

A possible reason not to opt for the more versatile IoT protocol suite may arise from specific regulation applying to specific domains. In such a case, gateways and wrappers may still save the day.

Service hosting

Services are the primary active software in our IoT Domain Model. However, it is left to the system designer to choose where to deploy them. Possible solutions are: on the smart objects themselves, on gateways or in the Web.

On smart objects (DC15.1)

This is the most intuitive among the choices offered as the service is deployed directly on the device which is providing a given functionality. However, depending on the hardware, this choice may carry along issues concerning the capability of the device. If complex services are needed in constrained environment, this is not the way to go.

On gateways (DC15.2)

Usually, the devices bridging two different networks are equipped with more computational power and storage space. Thus they allow the system integrator to deploy more complex services on them. However, once the network complexity increases, this choice may become the most difficult to maintain.

In the cloud (DC15.3)

Smart objects, in this case, are less smart as their capabilities are deployed in the Web. However, this choice offers great advantages in terms of availability and resilience. The worst drawback here is that the actual communication from the user to the smart object may be quite slower than DC9.1.

Service engine

The core service engine is the software in charge of discovering, retrieving and associating the services. It is quite a complex piece of software and might require powerful dedicated machines

to run. Here the choice left to the system integrator is whether to deploy it internally or to rely on third parties.

Internal (DC16.1)

Obviously, if the service engine is available internally, the system can be customized more deeply according to the service needs. However, this solution requires specific expertise which is not always available neither in the most advanced IT companies.

External provider (DC16.2)

This will be the most popular design choice for Small and Medium Enterprise, which need to rely on robust service providing granted availability and scalability.

Information storage

The last design choice concerning the deployment and operation view is related to where and how to store information. It is not always straightforward to decide whether to maintain internal databases or to store data in the cloud.

Local only (DC17.1)

According to this design choice, data are only available within the network where they belong. On the one hand this solution is appropriate for enforcing security and privacy, but on the other hand availability and resilience might be degraded.

Web only (DC17.2)

On the contrary, data may also be stored only in the cloud. Benefits and issues are reversed here than in the previous choice regarding security and privacy as well as availability and resilience. In fact, the system integrator will have to trade longer transaction times between users and smart objects with higher availability.

Local and web cached (DC17.3)

This solution combines the benefits offered by DC11.1 and DC11.2, but is exposed to both choices issues depending on the particular implementation of the system.

5.3.2 Risk analysis

The risk analysis carried out in this section aims at assessing risks pertaining to the IoT-A architecture and to classify them relatively to the underlying mechanisms they involve, the elements they affect and the overall criticality they present.

Risk analysis traditionally begins with a definition of the elements that have to be protected. Then an analysis of possible threats is provided. How identified threats may actually affect elements to be protected leads to the definition of risks. These latter have to be categorized, taking into account parameters such as criticality or probability of occurrence.

Multiple risk analysis methods have been proposed in the literature, such as the French EBIOS [50] or OCTAVE [51]. The methodology for risk analysis that has been chosen in IoT-A and that is used in this section is based on Microsoft STRIDE / DREAD [52]. This choice is based on two reasons: first, this methodology is designed to assess risks in the field of communications and information systems. Second, it largely bases on the analysis of architecture models and communications flows (instead of, for example, partly relying on experts interviews such as in EBIOS), which makes it perfectly suitable to be used within IoT-A architecture work package, in the current maturity status of the project, in which architecture and relationships between elements have largely been defined.

This section is organized as follows: first, a list of elements to be protected is provided. Then, the threats that may affect these elements (risk sources) are reviewed, following the STRIDE classification. A table of identified risks follows, each risk being assessed in accordance with DREAD methodology/metric.

Eventually, this risk analysis is intended to be used as input for subsequent evolutions of IoT-A architecture, on order to make it more resilient against the most critical risks.

5.3.2.1 Elements to protect

Elements to protect generally depend on the considered scenario. In the current analysis, it is worth, though, to provide a generic overview that encompasses all elements whose protection must be ensured in IoT-A. Therefore all IoT scenarios that were identified in IoT-A deliverable D1.2 were considered as inputs for the risk analysis performed in this section. These scenarios are:

- Transportation / Logistics
- Smart home
- Smart city
- Smart factory
- Retail
- eHealth
- Environment (Smart Grid)

The following elements to protect were identified:

- **Physical person**. This represents the Human User. Threats affecting the Human User are usually qualified as relating to 'safety' instead of 'security'. Such threats may arise if a critical service is diverted (e.g. returns wrong information, or even information specifically shaped to produce hazardous results) or made unavailable by an attacker. The eHealth scenario is the most prone to such attacks, even though their criticality depends on its level of automation. It is likely that most critical decisions will still require the involvement of a human operator.
- **Subject's privacy**. This element represents all information elements that a subject (either a User or a Service in the Domain Model terminology) does not explicitly agree to make publicly available, or whose availability should be restrained to a controlled set of subjects only.
- **Communications channel**. The communication channel itself has to be protected, especially to prevent attack regarding the integrity (tampering and replay attacks) or the confidentiality (eavesdropping) of the data that are exchanged over it. The communication channel should also be protected from attacks to the routing functionality (black/worm-hole, depletion,...).
- **Leaf devices**. IoT-A leaf devices represents the wide variety of Internet of Things element that the common IoT-A infrastructure interconnects. Tags, readers, sensors, actuators are part of this category of elements. Various protection schemes relevant to their object class are to be implemented. These should ensure the integrity of the software, hardware and location of these devices.
- **Intermediary devices**. Intermediary devices provide services to IoT-A leaf devices and, enable the novel IoT-A communication paradigm. The advanced gateway that is designed to interconnect constrained and unconstrained domains is an example of such intermediary entity. Disabling or tampering critical intermediary devices can lead to denial of service attacks against the global service infrastructure, and is considered within this scope. However, attacks against specific intermediary devices that offer non-critical facilitating functions have to be considered per se, hence mentioning intermediary entities within the list of elements to protect.

- **Backend services**. Backend services represent server-side applicative elements (e.g. data collection server communicating with sensor nodes). Compromising this software or the devices they are deployed on generally represents a critical threat against specific application systems and has to be prevented.
- **Infrastructure Services**: Discovery, Lookup and Resolution Services are the most critical services as they provide worldwide fundamental functionalities to IoT-A systems. In the same way, Security Services (Authorization, Authentication, Identity Management, Key Management and Trust and Reputation) are essential for a secure interaction between subjects.
- **Global systems / facilities**. This last category of elements to protect considers entire services in a global manner. For example, there might be a risk that an attack against the smart home scenario results in the complete disruption of the service, e.g. through the disruption of underlying communications between devices.

5.3.2.2 Risk Sources

The risk sources are categorized following the STRIDE classification, which is a widely used way of classifying threats that relate to information systems. STRIDE stands for **S**poofing Identity, **T**ampering with Data, **R**epudiation, **I**nformation Disclosure, **D**enial of Service, **E**levation of Privilege. These categories are quickly reviewed below:

- **Identity spoofing** means that one peer illegitimately uses the identity of another peer. Spoofing attacks can happen with respect to all kind of identifiers, irrespective of whether they are used to designate physical persons, devices or communication flows, for example.
- **Data tampering** means that an attacker is able to alter the content of data exchanged between two or more peers. Data tampering may involve subtle attack schemes, wherein the attacker is able to trigger specific behaviours at recipients by finely modifying original data.
- **Repudiation** relates to attacks in which an attacker performs illegitimate action and may afterwards deny having performing it, such that other nodes are unable to prove that the attacker actually behaved maliciously.
- **Information disclosure** means that information is disclosed to unauthorized peers. It is related to the existence of an authorization model that defines for each information element a set of peers that are authorized to access it (possibly under some specific conditions).
- **Denial of service** attack is carried out to disable a service offered to legitimate users (as opposed, for example, to more subtle schemes wherein the attacked service can be altered, e.g. making a search service return false results, without the legitimate users can notice it).
- **Elevation of privilege** may occur in systems that feature different classes of users, each class being mapped to a specific set of rights. Illegitimate elevation of privilege occurs when an attacker manages to acquire rights that would normally be granted to more privileged class(es) only. In the most critical case, an attacker may obtain administration rights on the entire system, or part of it, which means that he may perform arbitrary actions on the elements he has access to, thereby being able to destroy the system or entirely change its behaviour.

The risk sources that are considered in the following are restricted according to the following rules:

- Non-human risk sources either global (flood, lightning, fire, electrical, heat) or local (individual device failure) are not considered. Only human risk sources are.
- Among human risk sources, only theft/loss and hacker-initiated attacks are considered. Technical staff errors or accidents are not considered.

The STRIDE classification is used below to identify risks, as intersections between a STRIDE item and an element to protect.

	Spoofing Identity	**Tampering with Data**	**Repudiation**	**Information Disclosure**	**Denial of Service**	**Elevation of Privilege**
Physical person		Attack alters data so that wrong data is supplied to a critical monitoring system	Human Users might use unattended electronic devices leaving no digital trace		A service critical for user's safety is disabled	
User's privacy	User's identity is spoofed User is involved in transactions with a malicious peer			Attacker gains knowledge of user private parameters Attacker gains knowledge of user's location		
Communications channel		Alteration of the invocation of a service Alteration of the return value upon service invocation	Jamming wireless communication channels lead to local DoS attacks that can be repudiated	Attacker gains knowledge of sensitive exchanged data	Attacker disrupts communications	Wrong authorization information propagating from one server to another
Leaf devices	Loss or theft of physical device used for authentication Attacker changes the association between a Virtual Entity and the corresponding Physical Entity	Attacker gains control of an actuator Attacker alter leaf device content so that a user will eventually be redirected to a malicious content Attacker alter sensor		Disclosure of device configuration information Device identification Loss or theft of physical device containing private information	Attacker physically disables leaf device (local) Attacker physically disables leaf device (remote) Attacker prevents proper communication to an actuator	

		device so that monitoring of a Physical Entity fails				
Intermediary devices		Compromised intermediary devices alter traversing data	Intermediary devices behave maliciously and clients are not able to report the fact		Assisting intermediary devices are no longer usable	
Backend Services	Administrator role usurpation Backend account hacked			Massive disclosure of collected data	Backend Service is made unavailable	
Infrastructure Services	Attacker impersonates infrastructure Services, compromising IoT functionalities and/or other dependent infrastructure Services	Attacker poisons infrastructure databases or alter outgoing information		Disclosure of private Services (existence & description) Disclosure of access policies Disclosure of Identities and cryptographic material	Attacker denies legitimate users access to Infrastructure Services	
Global systems / facilities				Massive disclosure of users personal information	Disruption of a global service	

5.3.2.3 Risk Assessment

Identified risks were assessed using the DREAD methodology & (simplified) metrics. DREAD, defines scoring methodology and metric that help to evaluate the criticality of an identified threat. DREAD stands for **D**amage Potential, **R**eproducibility, **E**xploitability, **A**ffected Users, **D**iscoverability. It defines the criteria along which a threat is evaluated. Each criteria is given a rating between 0 and 10. Eventually, the threat can be globally rated (sum of D, R, E, A, D individual ratings) or can be described along with its individual ratings, to allow for more precise analysis. A simpler scheme for DREAD, used in what follows, consists in only affecting ratings in the form of L (Low), M (Medium), H (High) for each D, R, E, A, D rating.

Note that a 'High' rating for Exploitability means that it is easy for an attacker to carry out an attack leading to the identified threat, whereas a 'High' rating in Discoverability means that it is difficult to discover the threat. This is to ensure a coherent approach, in which 'Low' ratings decrease the overall criticality of a risk, whereas 'High' ratings increase it.

The DREAD methodology and metric is used below to evaluate identified risks. In addition to the DREAD rating, the table below provides initial information on detailed threats that may lead to the occurrence of the identified risk. In addition to this information, initial elements on threat mitigation are provided.

An evolution of this table will consist in enhancing this initial information to more complete attack trees, such that each risk can be mapped to a probabilistic review of the attack scenarios that can lead to it.

Risk	D/R/E/A/D rating	Examples of Causes	Mitigation
Attack alters data so that wrong data is supplied to a critical monitoring system	H/L/M/L/L ➔enforce **strong** security		*Data integrity protection* provided as part of **protocol security**.
Human Users might use unattended electronic devices leaving no digital trace	L/L/H/L/L ➔enforce weak security		Not specifically targeted. Addressable through proper (local / remote) user authentication scheme, which would be a function of the **Authentication** component.
A service critical for user's safety is disabled	H/M/M/L/L ➔enforce medium security		Not specifically targeted. Critical services have to be protected through redundancy of their key elements. Malicious actions are prevented through dedicated access control policies (security management). Communication medium between user and critical service has to be made robust against DoS attacks at all layers.
User's identity is spoofed	L/H/H/L/M ➔enforce medium security	Credential theft, Credential brute-forcing, Registration procedure vulnerable to man-in-the-middle attack	Robust user authentication procedure preventing MitM attacks, with proper credentials management policy provided by **Authentication** component.
User is involved in transactions with a malicious peer	L/H/H/M/L ➔enforce **strong** security	Redirection to advertising or malicious content, which may be cause by data tempering on communication channel or leaf node compromising	Trustworthy discovery / resolution / lookup system. Trustworthiness of the entire system is guaranteed through its security components (especially **Authentication** and **Trust and Reputation**) as well as its global robustness (security by design).
Attacker gains knowledge of user private parameters	M/M/M/L/H ➔enforce medium security	Characterization of a user as requiring certain data (discovery, lookup, resolution) Characterization of a user as providing certain data	Enforcement of a robust pseudonymity scheme ensuring both anonymity and unlinkability and provided by the **Identity Management** security

		Traceability (this path, hence this user)	component.
Attacker gains knowledge of user's location	L/H/M/L/H ➔enforce weak security		User's location can be hidden through reliance on pseudonyms provided by the **Identity Management** component.
Alteration of the invocation of a service	L/L/M/L/L ➔enforce weak security		End-to-end integrity protection of signalling to access a service (*Data integrity protection* is provided as part of **protocol security**).
Alteration of the return value upon service invocation	L/L/M/L/L ➔enforce weak security		End-to-end integrity protection of signalling to access a service (*Data integrity protection* is provided as part of **protocol security**).
Jamming wireless communication channels can lead to local DoS attacks that can be repudiated	M/H/L/M/M ➔enforce medium security		Not specifically targeted. Dealing with jamming-obfuscated DoS attacks would rather be addressed through physical means and start with the localization of the jammer.
Attacker gains knowledge of sensitive exchanged data	M/L/M/L/L ➔enforce medium security		End-to-end *confidentiality protection* of exchanged data, offered through **protocol security**.
Attacker disrupts communications	M/H/L/H/L ➔enforce medium security		Various DoS prevention schemes, which applicability depends on the communication technology (anti-jamming, enforced MAC, etc.). They are offered through security by design of the communication stack.
Wrong authorization information propagating from one server to another	M/L/L/H/M ➔enforce medium security		Strong security for server to server communications that leverages on individual credentials (e.g. certificates) instead of group keys, and allows for revocation (security by design, adequate management policies).
Loss or theft of a physical device used for authentication	M/L/H/L/L ➔enforce weak security		Two-factor authentication, when applicable.
Attacker changes the association between a Virtual Entity and the corresponding Physical Entity	M/L/M/H/L ➔enforce medium security	Wrong tag on a device Compromising of the resolution system	Secured discovery/ resolution/ lookup system.

Attacker gains control of an actuator	M/M/M/L/M ➔enforce medium security		Proper authorization scheme as offered by the **Authorization** component. End-to-end integrity protection, provided as part of **protocol security**.
Attacker alter leaf device content so that a user will eventually be redirected to a malicious content	M/M/H/M/L ➔enforce medium security		Not specifically targeted. Addressable through a proper URI verification system on user device.
Attacker alter sensor device so that monitoring of a Physical Entity fails	L/M/L/L/H ➔enforce weak security		Not specifically targeted. Sensitive physical values may be monitored by a large number of sensors, or sensor integrity can be remotely verified.
Disclosure of device configuration information	L/L/L/L/H ➔enforce weak security		Not specifically targeted. Unlinkability between different actions of the same device, provided by the **Identity Management** component, will mitigate the criticality of this threat.
Device identification	L/M/M/L/H ➔enforce medium security	Attacker bypasses in-place pseudonymity scheme and identifies a device as providing access to certain data	Adequate protection scheme requiring partial pre-knowledge of each other before a tag can be read by a reader.
Loss or theft of physical device containing private information	M/L/H/L/L ➔enforce medium security		Physical protection of stored credentials (e.g. security vault) – readability of a device only upon fulfilment of certain conditions (e.g. known reader).
Attacker physically disables leaf device (local)	L/H/H/L/L ➔enforce weak security	E.g. tag destruction	Not specifically targeted – typically addressable through physical investigation.
Attacker physically disables leaf device (remote)	M/H/L/H/L ➔enforce weak security	E.g. tag destruction by remote electromagnetic means	Not specifically targeted – typically addressable through physical investigation.
Attacker prevents proper communication to an actuator	M/H/L/M/L ➔enforce medium security		DoS detection / reaction scheme (security by design).
Compromised intermediary devices alter traversing data	M/H/M/M/L ➔enforce medium security		End-to-end security scheme put in place by the **Key Exchange and Management** component, and enforced by the relevant **Protocol Security** function. Remote monitoring of intermediary devices can

			be another means of dealing with this threat, through identification of compromised devices.
Intermediary devices behave maliciously and clients are not able to report the fact	M/M/L/M/H ➔enforce weak security		Remote monitoring of intermediary devices. Depending on the malicious action performed by intermediary devices, client nodes may mitigate it by applying end-to-end security schemes (**Key Exchange and Management** + **Protocol Security**).
Assisting intermediary devices are no longer usable	L/M/H/H/L ➔enforce medium security	Exhaustion attacks, Various specific attacks against the involved assistance mechanisms	DoS detection / reaction scheme.
Administrator role usurpation	H/M/L/H/L ➔enforce medium security	Administrator credentials disclosed / hacked / brute-forces	Not specifically targeted. Addressable through security management, credentials management policies.
Backend account hacked	M/M/L/H/M ➔enforce medium security		Not specifically targeted. Addressable through security management, credentials management policies.
Massive disclosure of collected data	H/M/L/H/L ➔enforce medium security		Not specifically targeted. Addressable through security management (databases).
Backend service is made unavailable	L/M/M/H/L ➔enforce medium security		DoS detection / reaction scheme.
Attacker impersonates infrastructure Services, compromising IoT functionalities and/or other dependent infrastructure Services	H/M/L/H/M ➔enforce medium security		Prevention of impersonation techniques through proper use of authentication / authorization procedures (enforced by the respective **Authentication** and **Authorization** components).
Attacker poisons infrastructure databases or alters outgoing information	H/H/L/H/M ➔enforce **strong** security		Proper authorization scheme put in place by the **Authorization** component mitigates this attack. Enforcement of a trust model (**Trust and Reputation** component) protects against blind acceptation of erroneous data.

Disclosure of private services (existence & description)	L/H/H/M/M ➔enforce medium security		Masking the belonging of multiple services to a single entity is another form of unlinkability, which can be provided through reliance on pseudonyms provided by the **Identity Management** component.
Disclosure of access policies	L/H/H/M/M ➔enforce medium security		Security management of infrastructure prevents global disclosure of access policies from the decision point. Probe discovery of access policies are more subtle, and have to be dealt with through adaptive security – this second type of attacks is not specifically targeted.
Disclosure of identities and cryptographic material	M/H/H/M/L ➔enforce **strong** security		Addressable through security management (databases).
Attacker denies legitimate users access to Infrastructure Services	M/H/L/M/L ➔enforce medium security		Exclusion of the attacker, once identified as such through the **Trust and Reputation** security component.
Massive disclosure of users personal information	H/L/L/H/L ➔enforce **strong** security		Secure storage of users personal data with dedicated protection architecture (e.g. firewall diodes) and access control rules – this is part of security management.
Disruption of a global service	H/M/L/H/L ➔enforce **strong** security		Reliance on all functional security components + proper security management.

6 Conclusions and Outlook

This second public version (v1.5) of the IoT-A Architectural Reference Model builds upon the first release in 2011 presented during the IOT Week 2011 in Barcelona. Following this presentation a long feedback process was started which led to this version. Therefore this version 1.5 is not only a great improvement to version 1 but also a consolidated version that took into account 300+ received comments from external stakeholders and internal partners involved into the other technical Work Packages of IoT-A.

In a nutshell the technical improvements touch all models of the RM and provide more explanations on the logical existing between the models of the RM and between the models of the RM and some views of the RA. As far as the RA is concerned many views and perspectives were added to the ones existing in ARM version 1 (D1.2). Finally a large chapter (Section 5) is fully dedicated to making this ARM useful to IoT system developers, by providing Best Practice guidance and a large set of Design Choices that provides the system architects with concrete option when designing a concrete architecture out of the ARM.

The ARM is not a “Style exercise” aiming at staying on the corner of someone’s desk. In order to fully reach its objective, the ARM needs to be used, challenged, squeezed, criticised by you “the reader”, in order to be improved by us. Only then it will reach its full maturity.

Following the IoT week 2012 – where a long session on Day#1 was dedicated to the ARM v1.5 presentation (D1.3) - a new feedback process has started and naturally, the ARM v2 (coming end of October 2012) will leverage on this feedback. However we have anticipated in our internal roadmap to improve some of the models of the RM (security/privacy/trust and communication), to improve all perspectives of the RA, to improve Management and Security aspects of the Functional view, and to allocate a large effort to the Best Practice and Design Choices section, which we consider as critical in order to reach our audience – you IoT System Architects- , and in order to ensure that our results are beneficial to the development of IoT.

References

[1] G. Mueller, "A Reference Architecture Primer", http://www.gaudisite.nl/referencearchitectureprimerpaper.pdf (accessed Nov. 1, 2010), 2008

[2] J. Miller and J. Mukerji (Ed.), MDA Guide Version 1.0.1, Object Management Group, Framingham, Massachusetts, June 2003.

[3] "Bio Instruments s.r.l.", http://phyto-sensor.com/FI-LP-FI-MP-FI-sp.en (accessed: 2012-04-05).

[4] Internet of Things - Architecture, "Terminology", http://www.IoT-A.eu/public/terminology (accessed 2011-06-14), 2011

[5] The Open Group, TOGAF™, 9th edition, 3rd impression ed.: Van Haren Publishing, Zaltbommel, 2009.

[6] C. M. MacKenzie, K. Laskey, F. McCabe, P. F. Brown, R. Metz, B. A. Hamilton (Ed's), "Reference Model for Service Oriented Architecture 1.0", OASIS, http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf, 2006

[7] ANSI/IEEE, "ANSI/IEEE 1471-2000 Standard for Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems," ANSI/IEEE, 2000

[8] E. Woods and N. Rozanski, "Using Architectural Perspectives", Fifth Working IEEE/IFIP Conference on Software Architecture, 2005

[9] A. Pastor, E. Ho, A. Salinas Segura, R. Kernchen, S. Meyer, J. Riedl, and A. Bassi, "Project Deliverable D6.1 - Requirements List", http://www.IoT-A.eu/public/public-documents/project-deliverables/1/1/IoT-A_Deliverable_6.1.pdf/at_download/file (accessed 2011-06-09), November 2010.

[10] Nicola Bui (Ed.), "Project Deliverable D1.1 - SOTA report on existing integration frameworks/architectures for WSN, RFID and other emerging IoT related Technologies", http://www.IoT-A.eu/public/public-documents/project-deliverables/1/1/110304_D1_1_Final.pdf/at_download/file (accessed 2011-06-09), 2011

[11] N. Rozanski and E. Woods, "Software Architecture with Viewpoints and Perspectives", http://www.viewpoints-and-perspectives.info/doc/spa191-viewpoints-and-perspectives.pdf (accessed 2011-06-15), 2005

[12] N. Rozanski and E. Woods, ”Software Systems Architecture: Working With Stakeholders Using Viewpoints and Perspectives”, Addison-Wesley Longman, 2011

[13] N. Rozanski and E. Woods, ”Applying Viewpoints and Views to Software Architecture“, Whitepaper, http://www.viewpoints-and-perspectives.info/vpandp/wp-content/themes/secondedition/doc/VPandV_WhitePaper.pdf (accessed 2012-05-23)

[14] P. Shames and T. Yamada, “Reference Architecture for Space Data Systems”, JPL TRS 1992+, http://trs-new.jpl.nasa.gov/dspace/handle/2014/7485 (accessed 2011-06-14), 2003

[15] T. Usländer (Ed.), “Reference Model for the ORCHESTRA Architecture (RM-OA) V2”, Open Geospatial Consortium Inc., OGC 07-024, 2007

[16] S. Tamblyn, H. Hinkel, D. Saley, “NASA CEV Reference GN&C Architecture”, 30th Annual AAS Guidance and Control Conference, AAS 07-071, http://ntrs.nasa.gov/archive/nasa/casi.ntrs.nasa.gov/20070005131_2007004869.pdf (accessed 2011-06-14), 2007

[17] Boehm B, "A Spiral Model of Software Development and Enhancement", ACM SIGSOFT Software Engineering Notes", "ACM", 11(4):14-24, August 1986

[18] The Consultative Committee for Space Data Systems, “Information Architecture Reference Model”, CCSDS_312.0-G-0, http://cwe.ccsds.org/sea/docs/SEA-IA/Draft%20Documents/IA%20Reference%20Model/ccsds_rasim_20060308.pdf (accessed: 2011-06-15), February 2006

[19] Open GeoSpatial Consotrium, “The OpenGIS abstract specification Topic 12: the OpenGIS Service architecture”, http://portal.opengeospatial.org/files/?artifact_id=1221 (accessed 2011-06-14), 2002

[20] C. Vicente-Chicote, B. Moros, and A. Toval, “REMM-Studio: an Integrated Model-Driven Environment for Requirements Specifications, Validation and Formatting”, International Journal of Object Technology, Vol. 6, No. 9, pp. 437-454, 2007.

[21] E. Gamma, R. Helm, R. E. Johnson, and J. Vlissides, “Design Patterns: Elements of Reusable Object-Oriented Software”, Addison-Wesley Professional, Part of the SEI Series in Software Engineering, October 1994

[22] R. Giffinger, C. Fertner, H. Kramar, R. Kalasek, N. Pichler-Milanovic, E. Meijers, "Smart cities - Ranking of European medium-sized cities". www.smart-cities.eu/download/smart_cities_final_report.pdf (accessed 2011-06-14), , 2007

[23] R. Nicholson, "Smart Cities: Proving Ground for the Intelligent Economy", http://www.slideshare.net/rlnicholson2/smart-cities-proving-ground-for-the-intelligent-economy (accessed 2011-06-15), 2010

[24] Field, A., "As Wal-Mart expands its requirements for RFID, others find new uses for the technology", Journal of Commerce, Vol. 9 Issue 16, p.21, 2008.

[25] P. Oldfield, "Domain Modelling" http://www.aptprocess.com/whitepapers/DomainModelling.pdf, (accessed Dec. 15, 2010), 2002

[26] Haller, S., "The Things in the Internet of Things", Poster at the Internet of Things Conference, Tokyo (IoT 2010), http://www.IoT-A.eu/public/news/resources/TheThingsintheInternetofThings_SH.pdf, (accessed Jan. 24, 2011), 2010.

[27] Serbanati, A., Madaglia, C.M., Ceipidor, U.B, "Building Blocks of the Internet of Things: State of the Art and Beyond", in RFID / Book 3, ISBN 979-953-307-026-0, InTech, 2011

[28] K. Römer, F. Mattern, T. Dübendorfer, J. Senn, "Infrastructure for Virtual Counterparts of Real World Objects", Technical Report ETHZ, http://www.inf.ethz.ch/vs/publ/papers/ivc.pdf (accessed 2011-06-09), 2002

[29] EPCglobal, "EPC Information Services (EPCIS) Version 1.0.1 Specification", September 2007, http://www.gs1.org/gsmp/kc/epcglobal/epcis/epcis_1_0_1-standard-20070921.pdf (accessed 2012-04-30)

[30] Martín, G. (Ed.), "Resource Description Specification", IoT-A deliverable D2.1, 2012, (not yet online!!!)

[31] NGSI Context Management Specification, Open Mobile Alliance http://www.openmobilealliance.org/Technical/release_program/docs/NGSI/V1_0-20101207-C/OMA-TS-NGSI_Context_Management-V1_0-20100803-C.pdf (accessed Jun. 11, 2012), 2010

[32] Furness, A., "Ontology for Identification", in CASAGRAS Final Report, Annex C, 2009, http://www.grifs-project.eu/data/File/Casagras_Final%20Report.pdf (accessed 2012-04-18)

[33] OWL 2 Web Ontology Language Definition, http://www.w3.org/TR/owl2-overview/ (accessed 2011-06-14), 2009

[34] Ricardo de las Heras (Ed.), “Project deliverable D4.1 - Concepts and Solutions for Identification and Lookup of IoT Resources”, December 2011

[35] SAP Research “what is USDL and why do we need it", http://www.internet-of-services.com/index.php?id=264&L=0 (accessed 2012-05-31), 2012

[36] N. Gruschka (Ed), IoT-A D4.2: Deliverable Concepts and Solutions for Privacy and Security in the Resolution infrastructure, http://www.IoT-A.eu/public/public-documents/documents-1/1/1/d4.2/at_download/file retrieved on May 2012

[37] D. Gambetta, Can We Trust Trust?, in Trust: Making and Breaking Cooperative Relations, D. Gambetta, pp. 213-237, Basil Blackwell, Oxford, 2000, http://www.loa.istc.cnr.it/mostro/files/gambetta-conclusion_on_trust.pdf, retrieved in May 2012

[38] M. Rossi (Ed), IoT-A D3.3 Deliverable: Protocol Suite, available from http://www.IoT-A.eu/public/public-documents, retrieved on ……..

[39] Ashton, K., “That ‘Internet of Things’ Thing”, RFID Journal, June 2009, http://www.rfidjournal.com/article/view/4986 (accessed 2012-04-30)

[40] IoT-I, ”Survey on IoT scenarios”, private communication, 2011.

[41] “Demonstration of the electrochemical fatigue sensor system at the transportation technology center facility”, http://www.arema.org/files/library/2007_Conference_Proceedings/Demonstration-Electochemical_Fatigue_Sensor_System_TTC_2007.pdf (accessed: 2012-04-05).

[42] Philip Levis, “TinyOS Programming”, http://www.tinyos.net/tinyos-2.x/doc/pdf/tinyos-programming.pdf (accessed 2012-04-05).

[43] “Apache HBase”, http://hbase.apache.org/ (accessed: 2012-04-05).

[44] Joseph Polastre, Robert Szewczyk, and David Culler, “Telos: Enabling Ultra-Low Power Wireless Research,” in Proc. of the International Conference on Information Processing in Sensor Networks (IPSN), CA, USA, April 2005, pp. 364–369.

[45] Brucker, Achim D., Isabelle Hang, Gero Lückemeyer, and Raj Ruparel. “SecureBPMN: Modeling and Enforcing Access Control Requirements in Business Processes”. ACM symposium on access control models and technologies (SACMAT), 2012, New York, NY, USA.

[46] Suparna De (Ed.), “Project deliverable D4.3 - Concepts and Solutions for Entity-based Discovery of IoT Resources and Managing their Dynamic Associations”, March 2012

[47] Resource Description Framework (RDF), http://www.w3.org/RDF/ (accessed 2012-06-01), 2004

[48] SPARQL Query Language for RDF, http://www.w3.org/TR/rdf-sparql-query/ (accessed 2012-06-01), 2008

[49] RDFa Primer, http://www.w3.org/TR/xhtml-rdfa-primer/ (accessed 2012-06-01), 2008

[50] Ebios 2010 - expression of needs and identification of security objectives. Technical report, Agence Nationale de la Sécurité des Systèmes d"Information (ANSSI), 2010.

[51] OCTAVE (Operationally Critical Threat, Asset, and Vulnerability Evaluation), http://www.cert.org/octave/, retrieved on May 2012.

[52] Microsoft, The STRIDE Threat Model, http://msdn.microsoft.com/en-us/library/ee823878, retrieved on May 2012.

[53] Payam M. Barnaghi, Stefan Meissner, Mirko Presser, and Klaus Moessner, "Sense and Sens’ability: Semantic Data Modelling for Sensor Networks", *In Proceedings of the ICT Mobile Summit 2009*, June 2009.

Appendix

A Terminology

This glossary aims at defining the new terminology introduced by this document and at updating the existing terms if their meaning has been refined during the editing of this document. The changes introduced by this document will be applied to the IoT-A terminology webpage at http://www.IoT-A.eu/public/terminology. Please, always refer to the online version of the glossary, since it contains a more complete version of the glossary and, after new deliverables got released, new definitions may have superseded these ones.

Also, the terms written in *italic* have a specific definition in the table.

Term	Definition	Source
Active Digital Artefact	*Active Digital Artefacts* are running software applications, agents or *Services* that may access other *Services* or *Resources*.	Internal
Active Digital Entity	Any type of active code or software program, usually acting according to a *Business Logic*. Obsolete: the term to be used is *Active Digital Artefact*	Internal
Actuators	Special *Device* that executes a change in the physical state of one or more *Physical Entities.*	Internal
Address	An address is used for locating and accessing – "talking to" – a *Device*, a *Resource*, or a *Service*. In some cases, the ID and the Address can be the same, but conceptually they are different.	Internal
Application Software	"Software that provides an application *service* to the *user*. It is specific to an application in the multimedia and/or hypermedia domain and is composed of programs and data".	[ETSI- ETR173]
Architectural Reference Model	The IoT-A architectural reference model follows the definition of the IoT reference model and combines it with the related IoT reference architecture. Furthermore, it describes the methodology with which the reference model and the reference architecture are derived, including the use of internal and external stakeholder requirements.	Internal
Architecture	"The fundamental organization of a *system* embodied in its components, their relationships to each other, and to the environment, and the principles guiding its design and evolution".	[IEEE-1471-2000]
Architecture Vision	"A high-level, aspirational *view* of the target *architecture*."	[TOGAF9]
Aspiration	"*Stakeholder* Aspirations are statements that express the expectations and desires of the various *stakeholder*s for the *service*s that the final [*system*]	[E-FRAME]

	implementation will provide.”	
Association	An association establishes the relation between a *service* and *resource* on the one hand and a *Physical Entity*on the other hand.	Internal
Augmented Entity	The composition of a *Physical Entity* and its associated *Virtual Entity*.	Internal
AutoID and Mobility Technologies	“Automatic Identification and Mobility (AIM) technologies are a diverse family of technologies that share the common purpose of identifying, tracking, recording, storing and communicating essential business, personal, or product data. In most cases, AIM technologies serve as the front end of enterprise software *system*s, providing fast and accurate collection and entry of data. AIM technologies include a wide range of solutions, each with different data capacities, form factors, capabilities, and "best practice" uses. AIM technologies also include mobile computing devices that facilitate the collection, manipulation, or communication of data from data carriers as well as through operator entry of data via voice, touch screens or key pads. Each member of the AIM technology family has its own specific benefits and limitations -- meaning there is no "best" technology. Rather, applications may be best served by one or more AIM technologies. Multiple AIM technologies are often used in combination to provide enterprise-wide solutions to business issues. Most AIM technologies are defined by international and national technical standards. International, national or industry application standards also exist to define the use of AIM technologies.”	[AIMglobal]
Business Logic	Goal or behaviour of a system involving *Things.* *Business logic* serves a particular business purpose. *Business logic* can also define the behaviour of a single or multiple Physical Entities, or a complete business process.	Internal
Controller	Anything that has the capability to affect a *Physical Entity*, like changing its state or moving it.	Internal
Device	Technical physical component (hardware) with communication capabilities to other IT systems. A *device* can be either attached to or embedded inside a *Physical Entity*, or monitor a *Physical Entity* in its vicinity.	Internal
Digital Artefact	*Virtual Entities* are *Digital Artefacts* that can be	Internal

	classified as either active or passive.	
Digital Entity	Any computational or data element of an IT-based system. Obsolete: the new term to be used is *Digital Artifact*.	Internal
Discovery	Discovery is a *service* to find unknown *resources/services* based on a rough specification of the desired result. It may be utilized by a human or another *service*. Credentials for authorization are considered when executing the discovery.	Internal
Domain Model	"A domain model describes objects belonging to a particular area of interest. The domain model also defines attributes of those objects, such as name and identifier. The domain model defines relationships between objects such as "instruments produce data sets". Besides describing a domain, domain models also help to facilitate correlative use and exchange of data between domains".	[CCSDS_312.0-G-0]
Energy-harvesting Technologies	"*Energy-harvesting* (also known as power harvesting or energy scavenging) is the process by which energy is derived from external sources (e.g., solar power, thermal energy, wind energy, salinity gradients, and kinetic energy), captured, and stored. Frequently, this term is applied when speaking about small, wireless autonomous *device*s, like those used in wearable electronics and wireless *sensor network*s. Traditionally, electrical power has been generated in large, centralized plants powered by fossil fuels, nuclear fission or flowing water. Large-scale ambient energy, such as sun, wind and tides, is widely available but technologies do not exist to capture it with great efficiency. Energy harvesters currently do not produce sufficient energy to perform mechanical work, but instead provide very small amount of power for powering low-energy electronics. While the input fuel to large scale generation costs money (oil, coal, etc.), the "fuel" for energy harvesters is naturally present and is therefore considered free. For example, temperature gradients exist from the operation of a combustion engine and in urban areas, there is also a large amount of electromagnetic energy in the environment because of radio and television broadcasting".	[Wikipedia_EH]
Entity of Interest (EoI)	Any physical object as well as the attributes that describe it and its state that is relevant from a *user* or application *perspective*. The term is obsolete in the IoT-A reference model: the term *Physical Entity* should be used instead	Internal
Gateway	A Gateway is a forwarding element, enabling various	Internal

	local networks to be connected. Gateways can be implemented in *Device* that provides protocol translation between peripheral trunks of the IoT that are provided with lower parts of the communication stacks. For efficiency purposes, *gateways* can act at different layers, depending on which is the lowest layer in a common protocol implementation. *Gateways* can also provide support for security, scalability, service discovery, geo-localisation, billing, etc.	
Global Storage	*Storage* that contains global information about many *entities of interest*. Access to the *global storage* is available over the *Internet*.	Internal
Human	A *Human* that either physically interacts with *Physical Entities* or records information about them, or both.	Internal
Identifier (ID)	Artificially generated or natural feature used to disambiguate things from each other. There can be several *IDs* for the same *Physical Entity*. This set of *IDs* is an attribute of a *Physical Entity*.	Internal
Identity	Properties of an entity that makes it definable and recognizable.	Internal
Information Model	"An *Information Model* is a representation of concepts, relationships, constraints, rules, and operations to specify data semantics for a chosen domain of discourse. The advantage of using an information model is that it can provide sharable, stable, and organized structure of information requirements for the domain context. The *Information Model* is an abstract representation of entities which can be real objects such as devices in a network or logical such as the entities used in a billing system. Typically, the *Information Model* provides formalism to the description of a specific domain without constraining how that description is mapped to an actual implementation. Thus, different mappings can be derived from the same *Information Model*. Such mappings are called data models."	[AutoI]
Infrastructure Services	Specific services that are essential for any IoT implementation to work properly. Such services provide support for essential features of the IoT.	Internal
Interface	"Named set of operations that characterize the behaviour of an entity."	[OGS]
Internet	"The *Internet* is a global *system* of interconnected computer networks that use the standard *Internet* protocol suite (TCP/IP) to serve billions of *user*s worldwide. It is a network of networks that consists of	[Wikipedia_IN]

	millions of private, public, academic, business, and government networks of local to global scope that are linked by a broad array of electronic and optical networking technologies. The *Internet* carries a vast array of information *resource*s and *service*s, most notably the inter-linked hypertext documents of the World Wide Web (WWW) and the infrastructure to support electronic mail. Most traditional communications media, such as telephone and television *service*s, are reshaped or redefined using the technologies of the *Internet*, giving rise to *service*s such as Voice over *Internet* Protocol (VoIP) and IPTV. Newspaper publishing has been reshaped into Web sites, blogging, and web feeds. The *Internet* has enabled or accelerated the creation of new forms of *human* interactions through instant messaging, *Internet* forums, and social networking sites. The *Internet* has no centralized governance in either technological implementation or policies for access and usage; each constituent network sets its own standards. Only the overreaching definitions of the two principal name spaces in the *Internet*, the *Internet*-protocol address space and the domain-name *system*, are directed by a maintainer organization, the *Internet* Corporation for Assigned Names and Numbers (ICANN). The technical underpinning and standardization of the core protocols (IPv4 and IPv6) is an activity of the *Internet* Engineering Task Force (IETF), a non-profit organization of loosely affiliated international participants that anyone may associate with by contributing technical expertise.”	
Internet of Things (IoT)	The global network connecting any smart object.	Internal
Interoperability	“The ability to share information and services. The ability of two or more systems or components to exchange and use information. The ability of systems to provide and receive services from other systems and to use the services so interchanged to enable them to operate effectively together.”	[TOGAF 9]
IoT Service	Software component enabling interaction with *resources* through a well-defined interface, often via the Internet. Can be orchestrated together with non-IoT services (e.g., enterprise services).	Internal
Local Storage	Special type of *Resource* that contains information about one or only a few *Entities* in the vicinity of a *device*.	Internal
Location Technologies	All technologies whose primary purpose is to establish and communicate the location of a *device* e.g. GPS,	Internal

	RTLS, etc.	
Look-up	In contrast to *Discovery*, *Look-up* is a *Service* that addresses exiting known *Resources* using a key or *Identifier*.	Internal
M2M (also referred to as machine to machine)	"The automatic communications between *device*s without *human* intervention. It often refers to a *system* of remote *sensor*s that is continuously transmitting data to a central *system*. Agricultural weather sensing *system*s, automatic meter reading and *RFID* tags are examples."	[COMPDICT-M2M]
Microcontroller	"A *microcontroller* is a small computer on a single integrated circuit containing a processor core, memory, and programmable input/output peripherals. Program memory in the form of NOR flash or OTP ROM is also often included on chip, as well as a typically small amount of RAM. *Microcontroller*s are designed for embedded applications, in contrast to the microprocessors used in personal computers or other general purpose applications. *Microcontroller*s are used in automatically controlled products and *device*s, such as automobile engine control *system*s, implantable medical *device*s, remote controls, office machines, appliances, power tools, and toys. By reducing the size and cost compared to a design that uses a separate microprocessor, memory, and input/output *device*s, *microcontroller*s make it economical to digitally control even more *device*s and processes. Mixed signal *microcontroller*s are common, integrating analog components needed to control non-digital electronic *system*s".	[Wikipedia_MC]
Network-based resource	*Resource* hosted somewhere in the network, e.g., in the cloud.	Internal
Next-Generation Networks (NGN)	"Packet-based network able to provide telecommunication *service*s and able to make use of multiple broadband, QoS-enabled transport technologies and in which *service*-related functions are independent from underlying transport-related technologies"	[ETSI_TR_102_477]
Observer	Anything that has the capability to monitor a *Physical Entity*, like its state or location.	Internal
On-device Resource	*Resource* hosted inside a *Device* and enabling access to the *Device* and thus to the related *Physical Entity*.	Internal
Passive Digital Artefact	*PDigital Artefactassive Digital Artefacts* are passive software elements such as data-base entries or other digital representations of the *Physical Entity*.	Internal

Passive Digital Entities	A digital representation of something stored in an IT-based system. Obsolete: the term to be used is *Passive Digital Artefact*.	Internal
Perspective (also referred to as architectural perspective)	"Architectural perspective is a collection of activities, checklists, tactics and guidelines to guide the process of ensuring that a *system* exhibits a particular set of closely related quality properties that require consideration across a number of the *system*'s architectural *view*s."	[Rozanski, 2005]
Physical Entity	Any physical object that is relevant from a user or application perspective.	Internal
Reference Architecture	A reference architecture is an architectural design pattern that indicates how an abstract set of mechanisms and relationships realises a predetermined set of requirements. It captures the essence of the architecture of a collection of systems. The main purpose of a reference architecture is to provide guidance for the development of architectures. One or more reference architectures may be derived from a common reference model, to address different purposes/usages to which the Reference Model may be targeted.	Internal
Reference Model	"A reference model is an abstract framework for understanding significant relationships among the entities of some environment. It enables the development of specific reference or concrete architectures using consistent standards or specifications supporting that environment. A reference model consists of a minimal set of unifying concepts, axioms and relationships within a particular problem domain, and is independent of specific standards, technologies, implementations, or other concrete details. A reference model may be used as a basis for education and explaining standards to non-specialists."	[OASIS-RM]
Requirement	"A quantitative statement of business need that must be met by a particular *architecture* or work package."	[TOGAF9]
Resolution	*Service* by which a given ID is associated with a set of *Addresses* of information and interaction *Services*. Information *Services* allow querying, changing and adding information about the thing in question, while interaction services enable direct interaction with the thing by accessing the *Resources* of the associated *Devices*. Resolution is based on a priori knowledge.	Internal
Resource	Heterogeneous, generally system-specific, software components that store or process data or information about one or more *Physical Entities*, or that provide access to measurements and actuations in the case of	Internal

	Sensors and *Actuators* respectively.	
RFID	"The use of electromagnetic or inductive coupling in the radio frequency portion of the spectrum to communicate to or from a tag through a variety of modulation and encoding schemes to uniquely read the *identity* of an RF Tag."	[ISO/IEC 19762]
Sensor	Special *Device* that measures physical characteristics of one or more *Physical Entities*.	Internal
Service	Platform-independent computational entity that can be used in a platform-independent way.	Internal
Stakeholder (also referred to as system stakeholder)	"An individual, team, or organization (or classes thereof) with interests in, or concerns relative to, a system."	[IEEE-1471-2000]
Storage	Special type of *Resource* that stores information coming from *Resources* and provides information about *Entities*. They may also include *Services* to process the information stored by the *Resource*. As *Storages* are *Resources*, they can be deployed either on-device or in the network.	Internal
System	"A collection of components organized to accomplish a specific function or set of functions."	[IEEE-1471-2000]
Tag	Label or other physical object used to identify the *Physical Entity* to which it is attached.	Internal
Thing	Generally speaking, any *physical object* in combination with its *digital representation*. In other words, it denotes the same concept as an *Augmented Entity*.	Internal
User	A *Human* or some *Active Digital Entity* that is interested in interacting with a particular physical object.	Internal
View	"The representation of a related set of concerns. A *view* is what is seen from a *viewpoint*. An architecture *view* may be represented by a model to demonstrate to stakeholders their areas of interest in the architecture. A *view* does not have to be visual or graphical in nature".	[TOGAF 9]
Viewpoint	"A definition of the perspective from which a *view* is taken. It is a specification of the conventions for constructing and using a *view* (often by means of an appropriate schema or template). A *view* is what you see; a *viewpoint* is where you are looking from - the vantage point or perspective that determines what you see".	[TOGAF 9]
Virtual Entity	Computational or data element representing a *Physical Entity*. Virtual Entities can be either Active or Passive	Internal

	Digital Entities.	
Wireless communication technologies	"Wireless communication is the transfer of information over a distance without the use of enhanced electrical conductors or "wires". The distances involved may be short (a few meters as in television remote control) or long (thousands or millions of kilometres for radio communications). When the context is clear, the term is often shortened to "wireless". Wireless communication is generally considered to be a branch of telecommunications."	[Wikipedia_WI]
Wireless Sensors and Actuators Network	"Wireless *Sensor* and *Actuator* Networks (WS&ANs) are networks of nodes that sense and, potentially, control their environment. They communicate the information through wireless links enabling interaction between people or computers and the surrounding environment."	[OECD2009]
Wireline communication technologies	"A term associated with a network or terminal that uses metallic wire conductors (and/or optical fibres) for telecommunications."	[Setzer-Messtechnik, 20102010]

- [AIMglobal] Association for Automatic Identification and Mobility, online at: http://www.aimglobal.org/
- [AutoI] Information Model, Deliverable D3.1, Autonomic Internet (AutoI) Project. Online at: http://ist-autoi.eu/autoi/d/AutoI_Deliverable_D3.1_-_Information_Model.pdf
- [CCSDS_312.0-G-0] Information architecture reference model. Online at: http://cwe.ccsds.org/sea/docs/SEA-IA/Draft%20Documents/IA%20Reference%20Model/ccsds_rasim_20060308.pdf
- [COMPDICT-M2M] Computer Dictionary Definition, online at: http://www.yourdictionary.com/computer/m2-m
- [E-FRAME] E-FRAME project, available online at: http://www.frame-online.net/top-menu/the-architecture-2/faqs/stakeholder-aspiration.html
- [EPCglobal] EPC Global glossary (GS1), online at: http://www.epcglobalinc.org/home/GS1_EPCglobal_Glossary_V35_KS_June_09_2009.pdf
- [ETSI-ETR173] ETSI Technical report ETR 173, Terminal Equipment (TE); Functional model for multimedia applications. Available online: http://www.etsi.org/deliver/etsi_etr/100_199/173/01_60/etr_173e01p.pdf
- [ETSI_TR_102_477] ETSI Corporate telecommunication Networks (CN); Mobility for enterprise communication, online at: http://www.etsi.org/deliver/etsi_tr/102400_102499/102477/01.01.01_60/tr_102477v010101p.pdf
- [IEEE-1471-2000] IEEE 1471-2000, "IEEE Recommended Practice for Architectural Description of Software-Intensive Systems"
- [ITU-IOT] the Internet of Things summary at ITU, online at: http://www.itu.int/osg/spu/publications/internetofthings/InternetofThings_summary.pdf
- [ISO/IEC_2382-1] Information technology -- Vocabulary -- Part 1: Fundamental terms, online at: http://www.iso.org/iso/iso_catalogue/catalogue_tc/catalogue_detail.htm?csnumber=7229

- [OGS] Open GeoSpatial portal, the OpenGIS abstract specification Topic 12: the OpenGIS Service architecture. Online at: http://portal.opengeospatial.org/files/?artifact_id=1221
- [OASIS-RM] Reference Model for Service Oriented Architecture 1.0 http://docs.oasis-open.org/soa-rm/v1.0/soa-rm.pdf
- [OECD2009]: “Smart Sensor Networks: Technologies and Applications for Green Growth”, December 2009, online at: http://www.oecd.org/dataoecd/39/62/44379113.pdf
- [Sclater, 2007] Sclater, N., Mechanisms and Mechanical Devices Sourcebook, 4th Edition (2007), 25, McGraw-Hill
- [Setzer-Messtechnik, 2010] Setzer-Messtechnik glossary, July 2010, online at: http://www.Setzer-Messtechnik, 2010.at/grundlagen/rf-glossary.php?lang=en
- [TOGAF9] Open Group, TOGAF 9, 2009
- [Wikipedia_EH] Energy harvesting page on Wikipedia, online at: http://en.wikipedia.org/wiki/Energy_harvesting
- [Wikipedia_IN] Internet page on Wikipedia, online at: http://en.wikipedia.org/wiki/Internet
- [Wikipedia_MC] Microcontroller page at Wikipedia, online at: http://en.wikipedia.org/wiki/Microcontroller
- [Rozanski, 2005] Software Architecture with Viewpoints and Perspectives, online at: http://www.viewpoints-and-perspectives.info/doc/spa191-viewpoints-and-perspectives.pdf
- [Wikipedia_WI] Wireless page on Wikipedia, online at: http://en.wikipedia.org/wiki/Wireless

B Requirements

The purpose of this section is to describe the process in which requirements were created and refined, so that they could serve as inputs for developing the views, perspectives and the functional decomposition shown in this document (see Section 4.2.2). Since the time of publication of D1.2, new internal requirements were introduced while an update of the requirements coming from stakeholders was integrated in D6.2. After D6.2 a major rework was done regarding the existent set of requirements lists. The result is a single *unified requirements* list merging the unified requirements with the internal requirements, as well as recently emerged security requirements in order to provide a unique list with a consistent numbering scheme. Henceforward in the project this integrated list will be mentioned as the *unified requirements list*. The whole development process of the requirements from the beginning until D1.3 is described in Figure 44.

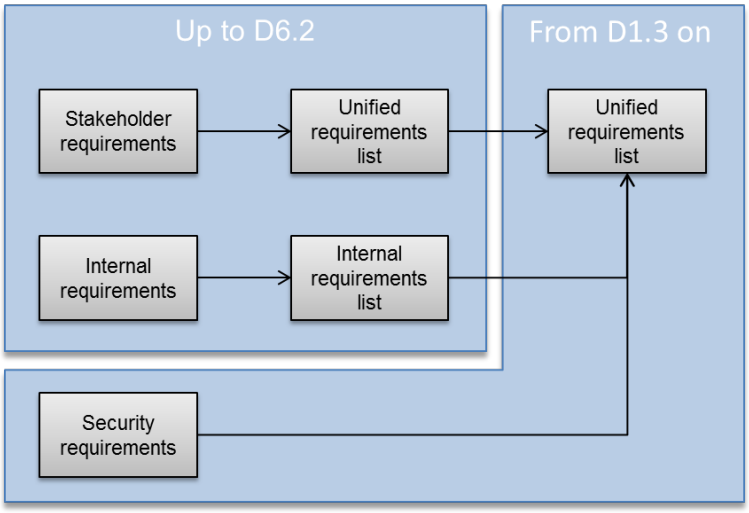

Figure 44: Evolution of requirements lists towards unified requirements list in D1.3

B.1 Requirements Gathering Methodology

B.1.1 Gathering external requirements from stakeholders

B.1.1.1 First Stakeholder Workshop (SW1)

The process began with collecting requirements from 7 stakeholders during the first stakeholder workshop in Paris, October 2010. The members of the stakeholder group were representatives of a wide range of business domains with an interest on the Internet of Things: Logistics, Healthcare, Technology Integration, Retail, Automotive, Service Integrators, Telecom Operators, Law, Standardization and Veterinary Medicine. These stakeholder aspirations were then reviewed individually by WP1 and WP6, each providing input relevant to their respective work packages. In WP1, after the requirements were reviewed, they were used to develop the views and functional decomposition for a first Draft Initial Architecture. The inputs of WP1 and WP6 were then combined, so that a unified set of requirements were obtained (as shown in Figure 45) The underlying unification process consisted of aligning the notations in order to

achieve consistency and of a generalization of the partially specific stakeholder objectives. This resulting set of requirements was then used to refine the views and functional decomposition as found in D1.2 Initial Architecture Reference Model.

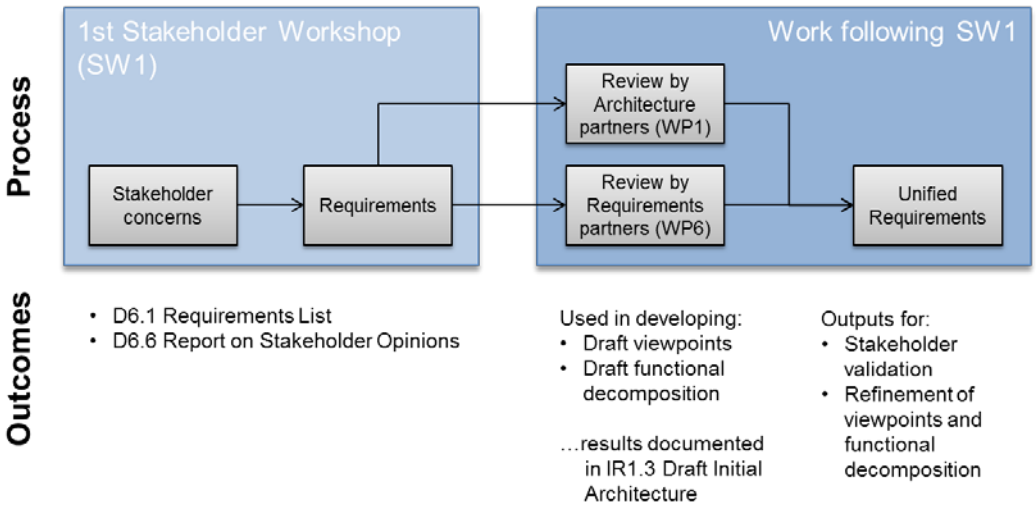

Figure 45: Development process for Requirements

B.1.1.2 Second and third Stakeholder Workshop 2 (SW2 + SW3)

Within IoT week 2011 in Barcelona the second stakeholder workshop took place. Its purpose was on the one hand to discuss the outcomes of SW1 and on the other hand to present the initial concept of the validation approach.

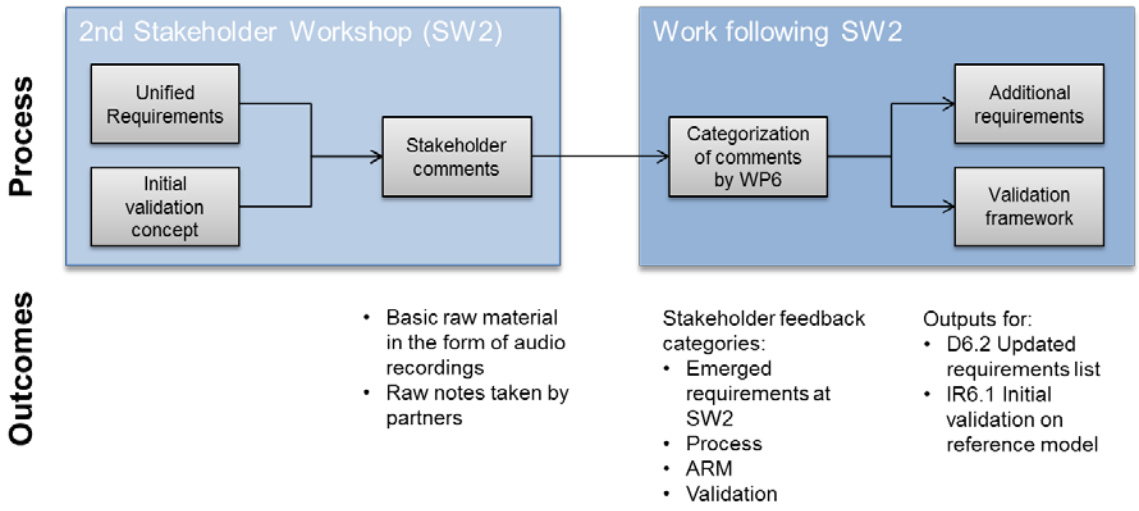

Figure 46: Further development of requirements and validation approach

Both inputs led to stakeholder comments which, in turn, have been categorized using four different feedback categories to identify new requirements, get feedback to the requirements distillation process and the ARM in general, as well as the feasibility of the presented validation approach. After that, a validation framework has been generated considering the improvement suggestions from the stakeholders and potential new requirements which were implemented in D6.2 Updated requirements list. The overall approach undertaken for SW2 is shown in Figure 46.

Stakeholder workshop 3 (SW3) was held one day before IoT International Forum in Berlin, 22nd of November 2011. During this event the objectives included to conduct open discussions with the goal to obtain new requirements, as well as to discuss the further developed validation process. By means of showing a set of visionary IoT videos the stakeholders were encouraged to evaluate the feasibility for the future and to identify possible issues (e.g. security or privacy) to infer new requirements based on their statements. The post-processing was similar to the post-processing of SW2 that is all stakeholder comments were collected, extracted and transcribed in order to categorize them into the categories mentioned above.

B.1.2 Gathering internal requirements

A set of technical requirements were acquired from the partners spanning the entire IoT-A project, in all of IoT-A's different aspects: this includes specialists in orchestration, communication, discovery & lookup, and in IoT objects and platforms (topics which roughly map onto the project work package structure).

The approach taken was to ask each work package (which corresponded roughly to the topics cited above) to analyse the state-of-the-art work which they carried out in D1.1, and formulate best practices by writing requirements for the IoT-A ARM.

Additionally upon completion of the system use cases (see Annex C), each work package was requested to extract the requirements for certain functionalities which an IoT system should have. It should be noted that some of these internal requirements were considered as too detailed or too implementation specific for the RA-level unified requirement list – those were removed from this list although they remain valuable input at the corresponding WP level.

Finally, Security requirements evolved from the fact that in almost each IoT scenario one has to deal with security or privacy issues. As a result, IoT-A created a cross work package security task force in charge of these issues. In particular, this task force compiled a list of requirements containing only security-related requirements from internal partners.

B.1.3 Unification process

Gathered requirements were aggregated and carefully reviewed to produce a comprehensive list of requirements for the IoT-A Reference Architecture and Reference Models. In the process, some external requirements which were considered too specific (e.g. to a specific use case scenario) or too vague had to be rewritten, and in some cases discarded. Similarly, some internal requirements were deemed as too implementation specific and removed from this list, although they remain valuable input for the technical WP aiming at having components implemented.

B.2 Unified requirements list

For the reader reference, the following (simplified) unified requirement list and the mapping to the corresponding view, perspective and relevant concepts from the Reference Model are presented below. The reader is advised that this list is work in progress, but gives a good preview of the Final Requirement List that will be released in D6.3.

Only a subset of the full requirements list fields is used in the following to ease the reading - each requirement consists of a unique ID, a requirement type and a requirement description. The reason why the requirement exists turns out in the “Rationale”. The traceability to the RA is provided in the two columns “View” and “Perspective”, whereby it should refer to only one of the two. The column “Reference Model” follows the same purpose of traceability where each requirement is assigned to components of the RM. The different fields are depicted in more detail in the table below.

Out of scope requirements have been excluded from the document. Thus, the continuous numbering in the ID column is interrupted in some spots.

Field	Description
ID	Each requirement is uniquely identified by a three-digit number: UNI.*klm*. All requiremnts with *klm* > 200 are either based on state of the art or about our understanding of the IoT domain.
Requirement Type	Type of requirement among the three categories Functional Requirements / Non-Functional Requirements / Design Constraints, abbreviated as FR/NFR/DC.
Description	The description is the intent of the requirement. It is a statement about what the system has to fulfil according to the *rationale*.
Rationale	The rationale is the reason behind the requirement’s existence. It explains why the requirement is important and how it contributes to the system’s purpose. It typically refers to direct stakeholder input for stakeholder-originated requirements, or an explanation/reference for internally-originated requirements.
View	One or several views to which the requirement is related.
Functionality Group	One or several functionality groups in the functional decomposition to which the requirement is related.
Functional Component	One or several components in the functional decomposition to which the requirement is related. These functional components are part of the groups listed in the functionality-group field.
Domain Model	One or several domain-model entities to which the requirement is related.
Perspective	One or several perspectives to which a requirement is related.

UNI ID	Type	Description	Rationale	View	Perspective	Functional Group	Functional Component	Domain Model
UNI.001	NFR	The system shall provide a means to allow people to use Internet of Things services anonymously	Citizens want to protect their private data	(none)	Security and Privacy	Security	Identity Management	User, Service, Resource, Device
UNI.002	NFR	Users have control how their data is exposed to other users	Citizens want to protect their private data	(none)	Trust, security and privacy	Security	Authorisation	Human User, Service, Resource
UNI.003	NFR	The system shall enable the provision and exchange of semantics between services in order to support the design of new applications	I would like a way to create and exchange semantics between objects in order to design new applications	(none)	Evolution and Interoperability	(none specific)	(none specific)	Service, Resource
UNI.004	NFR	The system shall enable the semantic description of Physical Entities	I would like a way to create and exchange semantics between objects in order to design new applications	Information	(none)	(none specific)	(none specific)	(none)
UNI.005	FR	The system shall support event-based, periodic, and/or autonomous communication	The remote monitoring device gathers patient measurements, data and or events. Data may be communicated each time the device gathers the data, accumulated measurements may be communicated periodically (e.g., hourly, daily), or data may be delivered upon request or upon certain events	Functional	(none)	IoT Service	IoT Service	(none)

UNI.008	NFR	The system shall be able to run applications and services in an interoperable manner	The problem is to provide a framework, a set of scenarios where these applications could be developed in harmony, in an interoperable way and in a way that responses to the real needs of organization and people	(none)	Performance and Scalabiltiy	(none specific)	(none specific)	Service
UNI.010	NFR	The system shall enable autonomous goal-driven (task-driven) collaboration between devices or services	"I would expect that the traffic lights collaborate for a goal" - Smart objects should collaborate in order to realize a common goal (such as traffic lights in order to reduce traffic or pollution).	(none)	Evolution and Interoperability	(none specific)	(none spedific)	Device, Service
UNI.012	NFR	The system shall be able to handle interference between IoT devices (avoidance and detection)	In order to achieve a reliable eHealth service the system must be interference-free	(none)	Evolution and Interoperability	Communication	Error detection & correction	Service, Device
UNI.014	FR	The system shall support devices to activate themselves into a collaboration	The remote monitoring device is prepared for use and communication by the action of the patient or clinician. This may involve physically attaching or placing the device, registering the device, setting up the communications channels to M2M application entities, setting up the communications capabilities of the device and providing for secure communications.	Deployment	(none)	Management	Device Manager	Device, Service, Resource
UNI.015	FR	Devices shall have the possibility to be remotely controlled and configured	The remote monitoring device may be configured by via the M2M network by the M2M application entities. The configuration capability could span simple parametric changes, such as, reporting rates, event or alarm trigger levels, and dosing levels to downloading and securely restarting new operating software	Deployment, Operation	(none)	(none specific)	(none spedific)	Device, Service, Resource
UNI.016	FR	The system shall support Physical Entity location tracking (geo spatial and/or logical location)	High value assets need to be tracked in order to avoid theft and also to know where they are currently located	Information, Functional	(none)	Virtual Entity	VE Resolution, VE & IoT Service Monitoring, VE Service	Augmented Entity, Resource, Service

UNI.018	FR	The system shall support data processing (filtering, aggregation/fusion, ...) on different IoT-system levels (for instance device level)	The remote monitoring device gathers patient measurements, data and or events. Data may be communicated each time the device gathers the data, accumulated measurements may be communicated periodically (e.g., hourly, daily), or data may be delivered upon request or upon certain events	Functional	(none)	IoT Service	IoT Service	Service
UNI.019	FR	The system shall support user-initiated communication	Providers can initiate communication with the patients health monitoring device for a number of reasons. Examples of this include a provider querying the device for a reading or for configuring such a device	Functional	(none)	Communication	Communiction Trigger	(none)
UNI.020	FR	The system shall support real-time monitoring of radio usage of devices and gateways	The application knows the current radio transmission activity of the M2M device	Functional	(none)	Communication	Error detection & correction	(none)
UNI.021	FR	The user shall be able to control the radio activity of the system	The application can control the radio transmission	Functional	(none)	Communication	Error detection & correction	Device
UNI.022	FR	The system shall provide end users with secure access to resources	Patients are able to initiate communication to the providers Electronic Medical Record (EMR) or health database application using the secure messaging tool for a variety of purposes. Examples include providing manually gathered information on existing self-monitoring and/or chronic care regiments.	Functional	(none)	IoT Service, Security	IoT Service, Key Exchange & Management	(none)

UNI.023	FR	The system shall provide access to external information sources, e.g. health databases	Patients are able to initiate communication to the providers Electronic Medical Record (EMR) or health database application using the secure messaging tool for a variety of purposes. Examples include providing manually gathered information on existing self-monitoring and/or chronic care regiments.	Functional	(none)	(none specific)	(none specific)	Resource, Storage
UNI.026	FR	The system shall support time-critical message handling and delivery on a second time scale.	In case of emergency the Remote Monitoring Device has to send or receive time critical messages	Functional	Performance and Scalabiltiy	Communication	QoS	(none)
UNI.027	FR	The system shall support prioritization of services	In case of time-sensitive services the system needs to assure that important services are prioritized	Functional	Performance and Scalabiltiy	(none specific)	(none specific)	(none specific)
UNI.028	FR	The system shall provide a message-prioritization mechanism	Not every message has the same priority	Functional	Performance and Scalabiltiy	Communication	QoS	(none specific)
UNI.030	FR	The system shall provide a resolution infrastructure for naming, addressing and assignment of Virtual Entities and services	A system may be provided which is operable to determine a routing node for a data object. The system can comprise an identifier generator operable to generate an identifier for the data object on the basis of data content thereof, and a lookup engine operable to compare the identifier for the data object to a routing table to determine a routing node for the data element.	Functional	(none)	Virtual Entity, IoT Service	VE Resolution, IoT Service Resolution	VE, Service, Resource

UNI.031	FR	The system shall enable centralized or decentralized automated activities (control loops)	Today, due to sub-optimal processes, a lot of time and money is wasted. This situation could be improved a lot by tracking all the items/things, providing context data on them at any time and location, allowing for automated evaluation of the collected data and reacting immediately on a dangerous situation to protect against the break down of items.	Functional	(none)	IoT Business Process Management	Business Process Modeling, Business Process Execution	Service
UNI.032	FR	The system shall enable the planning of automated tasks	Today, due to sub-optimal processes, a lot of time and money is wasted. This situation could be improved a lot by tracking all the items/things, providing context data on them at any time and location, allowing for automated evaluation of the collected data and reacting immediately on a dangerous situation to protect against the break down of items.	Functional	(none)	IoT Business Process Management	Business Process Modeling, Business Process Execution	Service
UNI.036	FR	The system shall enable the retrieval of the self-description of things	My wish is to retrieve the capacity of a thing. Thus, I can plan a change maintenance of all my bulbs if they can say when they should be changed	Functional	(none)	Virtual Entitty	VE Resolution	Service, Resource
UNI.040	NFR	The system shall provide ways to ensure security and resilience	Road users and energy providers want to avoid shortages/ blackouts	(none)	Availability and Resilience	(none specific)	(none specific)	(none specific)
UNI.041	FR	The system shall provide historical information about the Physical Entity	A method for clarification whether the Cold/Hot Chain has been violated or not is required. To be able to do this, the continuous context information (e.g., temperature) of the things needs to be collected. This is for example of major importance to avoid any damage to the pharmaceutics during the transport and storage process.	Information, Functional	(none)	IoT Service	IoT Service	Physical Entity, Storage, MetaData

UNI.042	NFR	Both user and device must be able to exchange information about their state	Both the M2M server and the M2M device must be able to provide information about the current state	(none)	Evolution and Interoperability	(none specific)	(none specific)	User, Device, Service, Resource
UNI.043	FR	The system shall enable the composition of entity-related services	The costs for complex logistics and healthcare processes need to be kept on a low level. A modular setup of the applications and services is one important ingredient to achieve this. Therefore it should be very easy to integrate things together with their atomic services into other services, and it should be easy for things to use services provided by others.	Functional	(none)	Service Organization	Service Composition, Service Orchestration	Service
UNI.047	NFR	The system must ensure interoperability between objects or between applications	As an example, CCTV system could inform traffic management of the length of the waiting queue at a crossroad. Having smart traffic lights receiving such input from the CCTV system could, could help changing the schedule of green/red light to optimize the traffic.	(none)	Evolution and Interoperability	Security	Key Exchange & Mangement	(none specific)
UNI.048	FR	The system shall provide interoperable naming and addressing	IoT-A will play a role in terms of providing a kind of novel resolution infrastructure. We need to understand how best IoT could be served by scheme regarding the naming of objects, the addressing and assigning problems.	Functional	Evolution and Interoperability	Communication	Gateway, Routing & Addressing	(none)
UNI.050	NFR	The system shall support mobility of the Physical Entity	The use of M2M Devices for monitoring health related information is not confined to the residence of the patient.	(none)	Availability and Resilience	(none specific)	(none specific)	Augmented Entity
UNI.058	NFR	The system shall provide high availability	Communication blackouts are not accepted from client side and particularly if they are paying for premium services	(none)	Availability and Resilience	(none specific)	(none specific)	(none specific)

UNI.060	NFR	The system shall support different SLA	Communication blackouts are not accepted from client side and particularly if they are paying for premium services	(none)	Availability and Resilience	Communication	QoS	Service
UNI.062	DC	The system shall provide trusted and secure communication and information management	A method for clarification whether the Cold/Hot Chain has been violated or not is required. To be able to do this, the detailed context information (e.g., temperature) of the things, which have been collected in some database need to be easily made available. This is for example of major importance to avoid any damage to the pharmaceutics during the transport and storage process.	(none)	Trust, security and privacy	IoT Service, Security	IoT Service, Trust & Reputation, Key Exchange & Management	(none)
UNI.064	NFR	The system shall provide security through resilience	Security, why? Simply because the IoT - I am sure you will demonstrate it - is a kind of critical information infrastructure which means that if ever for whatever reason there is a failure somewhere on the IoT the impact will be so high that it would be a social loss, like if we do not have more electricity.	(none)	Trust, security and privacy, Availability and Resilience	(none specific)	(none specific)	Service
UNI.065	NFR	The system shall provide reliable services	In order to accommodate certain scenario, support of a certain degree of reliability might be necessary	(none)	Availability and Resilience	(none specific)	(none specific)	Service
UNI.066	FR	The system shall provide integrity validation of Virtual Entities, devices, resources, and services	In certain life-critical applications the device may be required to perform a secure start-up procedure that includes integrity checking.	(none)	Performance and Scalabiltiy	Communication, Management	Error detection & correction, Device Manager	Virtual Entity, Service, Device, Resource
UNI.067	FR	The system shall provide different access permissions to information	Sensitive data of patients must be kept secure in order to assure trust between the patients and to allow access to certain people	Functional	(none)	Security	Authorisation	Resource

UNI.070	FR	The system shall handle semantic interoperability	I would like a way to create and exchange semantics between objects in order to design new applications	Information	(none)	(none specific)	(none specific)	Service, MetaData
UNI.071	DC	The system shall provide standardized and semantic communication between services	Standard communications between objects, from a communication channel point of view but also from a semantic point of view. (Standardization of object semantic is somehow similar to the standardization of MIB (Management Information Base) of telecommunication equipments).	(none)	Evolution and Interoperability	(none specific)	(none specific)	Service
UNI.073	FR	The system shall allow the semantic description of Physical Entities and services by a user	I would like a way to create and exchange semantics between objects in order to design new applications	Information	(none)	(none specific)	(none specific)	Virtual Entity, Service, Resource
UNI.087	FR	The system shall support service lifecycle management	Road users want to use one service over a service life cycle	Operation	(none)	(none specific)	(none specific)	Service
UNI.089	FR	The system shall support reliable time synchronization	Services which depend on a precise time need a guarantee that the devices they are communicating to have the right time.	(none)	Performance and Scalabiltiy	Communication	Error detection & correction	
UNI.092	NFR	Remote services shall be accessible by hHuman Users	The mobile phone of the consumer can and should be used for interacting with product centric services	(none)	Availability and Resilience	(none specific)	(none specific)	Service
UNI.093	NFR	The system shall be extensible for future technologies.	The reference architecture shall provide an integral approach that combines legacy aspects as well as an imaginating vision on the Internet of Things.	(none)	Evolution and Interoperability	(none specific)	(none specific)	(none)
UNI.094	NFR	The reference architecture shall support any business scenarios.	The reference architecture shall provide the building blocks in a creative way coming from a business perspective.	(none)	Evolution and Interoperability	(none specific)	(none specific)	(none)

UNI.095	DC	The system shall include an interface to IP communication protocols.	The reference architecture shall consider that we have gateways to IP everywhere, so we must have a global addressing system with protocol and so on. That would be an evolution of IPv6. Or we need an integration package for existing addressing systems.	Functional	(none)	Virtual Entity, IoT Service, Communication	VE Resolution, IoT Service Resolution, Gateway	Service, Resource, Device, VE
UNI.096	FR	The system shall support the autonomous and dynamic selection of protocols without human intervention.	Future systems implementing the reference architecture shall allow for a dynamic selection of protocols and layers without any human intervention.	Functional	Evolution and Interoperability	Communication, Service Organization	Gateway, Service Composition, Service Orchestration	Device, Service
UNI.097	FR	The system shall support information (data) lifecycle management.	Deal with the lifecycle of information (how to distinguish, if information (tag) is temporary not available or not valid any more?)	Information	(none)	(none specific)	(none specific)	(none)
UNI.098	FR	The system shall have a semantic understanding of distance and location.	"It is necessary to make the system know what defines a distance." - this is necessary to discover location-based services	Information	(none)	IoT Service, Virtual Entity	IoT Service Resolution, VE Resolution	Service, VE
UNI.099	NFR	The system shall guarantee correctness of resolutions.	When searching for a certain object you need an implemented system that actually gives you the correct result.	Functional	Trust, security and privacy	IoT Service, Virtual Entity	IoT Service Resolution, VE Resolution	(none)
UNI.100	FR	The system should include means to wake-up sleepy devices.	We must look out also for some way to wake up sleepy communications in order to manage energy consume.	Functional	(none)	Communication	Energy Optimization	(none)
UNI.101	NFR	The system should include means to manage the energy consumption of devices.	We must look out for a highly energy efficient system.	(none)	Performance and Scalabiltiy	Communication	Energy Optimization	(none)
UNI.102	NFR	The system should take into account external computing resources, e.g. 'the cloud'.	Maybe there should be some part of processing information in the cloud.	(none)	Performance and Scalabiltiy	(none specific)	(none specific)	(none)

UNI.211	FR	The process-modeling notation has to be extensible in terms of the definition of new symbols, the specification of new syntax, the definition of serialisation and execution semantics.	The reuse of an existing process-modeling notation allows to focus the effort on the IoT-extension.	Information	(none)	IoT Business Process Management	Business Process Modeling	(none)
UNI.212	FR	The process-modeling notation has to be executable.	The projects task 2.2 and 2.3 should closly work together and represent a hand in hand solution.	Functional	(none)	IoT Business Process Management	Business Process Modeling	(none)
UNI.213	NFR	The systems' process modeling notation shall be able to describe IoT-specif aspects, as, for instance, availability.	The standard established process notations cannot cope with IoT specific aspects, but in order to address IoT aware processes, one needs to be able to describe them appropriately. Reference: Sonja Meyer, Klaus Sperner, Carsten Magerkurth, Jacques Pasquier (2011): Towards Modeling Real-World Aware Business Processes. Web of Things 2011. San Francisco, USA, June 6, 2011.	Information	(none)	IoT Business Process Management	Business Process Modeling	(none)
UNI.214	NFR	The specification of the system's process-modeling notation shall include a graphical representation.	A graphical process notation offers a symbolism to easily model and document business processes.	Information	(none)	IoT Business Process Management	Business Process Modeling	(none)
UNI.215	NFR	The process-modeling notation shall adhere to a standard.	A common standard maximizes the potential application of industrial stakeholders.	Information	(none)	IoT Business Process Management	Business Process Modeling	(none)
UNI.229	NFR	The process-execution functional component shall be "easily and fastly" extendable.	The development should focus on the IoT related extension.	Information	(none)	IoT Business Process Management	Business Process Execution	Service

UNI.230	NFR	The system's process execution functional component shall be interoperable with other functional components in the same functional group or, otherwise, with other functional groups.	Non-interoperable components defy the spirit of the functional decomposition.	(none)	Availability and Resilience	IoT Business Process Management, Service Organization	Business Process Execution, Service Orchestration	Service
UNI.232	FR	The process-execution engine must support the integration with a complex-event-processing (CEP) component.	One WP central process execution engine including the CEP enables a bigger research contribution.	Functional	Availability and Resilience	IoT Business Process Management, Service Organization	Business Process Execution, Service Orchestration	(none)
UNI.233	NFR	Mobile entities shall be able to provide events to the platform	Many Physical Entities such as mobile phones, products in a retail store, etc. are mobile and IoT-A must be able to detect changes related to those entities	(none)	Availability and Resilience	(none specific)	(none specific)	Active Digital Entity
UNI.234	NFR	Events are processed on a set of distributed nodes	A distributed architecture provides more flexibility in the way events are processed, saves energy and allows minimal functionality if there is no network connectivity	(none)	Performance and Scalability	Service Organization	Service Composition, Service Orchestration	(none)
UNI.235	FR	Processing of events shall take quality of informaton (QoI) into account	In IoT the quality of information stemming from events is often questionable.	Functional	(none)	Service Organization	Service Composition, Service Orchestration	(none)
UNI.236	FR	The system shall offer services for the retrival of quality of information related to Virtual Entities.	Different devices provide information with varying quality. An application may have certain quality requirements.	Functional	(none)	IoT Service	IoT Service	Service
UNI.237	FR	The system shall offer data types for describing the quality of information related to Virtual Entities.	Different devices provide information with varying quality. An application may have certain quality requirements.	Information	(none)	(none specific)	(none specific)	Resource

UNI.239	FR	The IoT-A architecture shall provide a Storage Resource with a shared cache, in which an observable phenomenon is stored	Due to resources could not be online all the time it could be necessary to incorporate an intermediate shared memory in order to store this information, so it could be accessed by services using this information.	Functional	(none)	IoT Service	IoT Service	Service
UNI.240	FR	The system shall provide unified interfaces to access and query the resource/entity meta data	This will enable WP4 discovery and identification and also reasoning mechanisms to access the required descriptions	Functional	(none)	Virtual Entity, IoT Service	VE Service, IoT Service	Service
UNI.241	FR	The system shall provide unified interfaces to access and query the observation and measurement data emerging from resources	This will enable integration of IoT data into business layer and high-level applications.	Functional	(none)	IoT Service	IoT Service	Service
UNI.244	FR	The orchestration engine shall interpret service descriptions	Service orchestration needs to be done based on IOPE information provided in service descriptions. Reference: Bell, Michael. 2008. Service-Oriented Modeling. Chichester: John Wiley & Sons.	Functional	(none)	Service Organization	Service Composition, Service Orchestration	Service
UNI.245	FR	The service organization shall support creation of new applications	Composite services allow added value services based on simple services	Functional	(none)	Service Organization	Service Composition, Service Orchestration	Service
UNI.247	FR	The service organization shall support flexible composition	Services involved in compositions can fail and need to be replaced by some serving equal needs. Reference: Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. Computer, 36(1), 41-50.	Functional	(none)	Service Organization	Service Composition, Service Orchestration	Service

UNI.251	FR	The service organization shall provide a feedback to the user who sent a composition request	The service user needs to be informed whether or not the composition request has succeded or failed due to uncertainty of service availability. Reference: Nielsen, J. (1993). Usability Engineering. Retrieved from http://dl.acm.org/citation.cfm?id=529793	Functional	(none)	Service Organization	Service Composition, Service Orchestration	Service
UNI.252	NFR	The service organization shall provide feedback within a reasonable amount of time.	A time out must be set for request/response loops. For requests entered by hHuman Users a limit of 10 seconds could be reasonable. After that an error is assumed. Reference: Nielsen, J. (1993). Usability Engineering. Retrieved from http://dl.acm.org/citation.cfm?id=529793	Operation	(none)	Service Organization	Service Composition, Service Orchestration	Service
UNI.253	FR	The orchestration engines shall support setting preferences for selecting services involved in composition	Users can have the possibility to prefer one service over another for any reason	Functional	(none)	Service Organization	Service Composition, Service Orchestration	Services (Integration & Interoperability Layer)

UNI.401	FR	Discovery and lookup services of the system shall allow locating Physical Entities based on geographical parameters	This requirement is derived from SmartProducts (SP) requirement "A SmartProduct should be able to locate another SmartProduct in the same environment w.r.t. their environment" Reference: [SmartProduct Deliverable: "D6.3.1 & D6.4.1 & D6.5.1 Initial Smart Products Communication Middleware, Initial Sensor and Actuator Integration Framework & Initial Context and Environment Model Framework". http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D6.345.1_Final.pdf]	Functional	(none)	Virtual Entity	VE Resolution	Augmented Entity (Physical Entity +Virtual Entity)
UNI.402	FR	The system shall provide geographical-location attributes for Virtual Entities	Derived from SP requirement "A SmartProduct should be able to access the location information of other SmartProducts" Reference: [SmartProduct Deliverable: "D6.3.1 & D6.4.1 & D6.5.1 Initial Smart Products Communication Middleware, Initial Sensor and Actuator Integration Framework & Initial Context and Environment Model Framework". http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D6.345.1_Final.pdf]	Functional	(none)	Virtual Entity	VE Resolution	Virtual Entity

UNI.403	FR	The system shall support a standardized location model and location-information representation.	Derived from SP requirement "Smart products shall support a standardized location model and location-information representation." Reference: [SmartProduct Deliverable: "D6.3.1 & D6.4.1 & D6.5.1 Initial Smart Products Communication Middleware, Initial Sensor and Actuator Integration Framework & Initial Context and Environment Model Framework". http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D6.345.1_Final.pdf]	Functional	(none)	Virtual Entity	VE Resolution	Virtual Entity
UNI.404	FR	The system shall support a hybrid location model, that is, it shall support symbolic coordinates as well as local and global geometric coordinates	Derived from SP requirement "Smart products shall support a hybrid location model, that is, it shall support symbolic coordinates as well as local and global geometric coordinates" Reference: [SmartProduct Deliverable: "D6.3.1 & D6.4.1 & D6.5.1 Initial Smart Products Communication Middleware, Initial Sensor and Actuator Integration Framework & Initial Context and Environment Model Framework". http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D6.345.1_Final.pdf]	Functional	(none)	Virtual Entity	VE Resolution	Virtual Entity

UNI.405	FR	The system shall allow programmers to add new coordinate reference systems and shall support the transformation of coordinates among them	Derived from SP requirement: The location model shall allow programmers to add new coordinate reference systems and shall support the transformation of coordinates among them [SmartProduct Deliverable: "D6.3.1 & D6.4.1 & D6.5.1 Initial Smart Products Communication Middleware, Initial Sensor and Actuator Integration Framework & Initial Context and Environment Model Framework". http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D6.345.1_Final.pdf]	Functional	(none)	Virtual Entity	(none)	Virtual Entity
UNI.406	FR	The discovery service of the system shall support the following location queries: position queries, nearest neighbour queries, navigational queries, and range queries	Derived from SP requirement: "The location model shall support the following common location queries: position queries, nearest neighbour queries, navigational queries, and range querie" Reference: [SmartProduct Deliverable: "D6.3.1 & D6.4.1 & D6.5.1 Initial Smart Products Communication Middleware, Initial Sensor and Actuator Integration Framework & Initial Context and Environment Model Framework". http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D6.345.1_Final.pdf]	Functional	(none)	Virtual Entity	VE Resolution	Virtual Entity

UNI.407	FR	The look-up service of the system shall withold or grant information depending on context. Context includes application involved, requesting entity, and security permissions	Derived from BRIDGE requirement: "A broad set of data from enterprise applications MAY be requested depending on context, industry, application, etc" Reference: [BRIDGE Deliverable "D2.1 Requirements document of serial level lookup service for various industries, Section C". http://www.bridge-project.eu/data/File/BRIDGE%20WP02%20Serial%20level%20lookup%20Requirements.pdf]	Functional	Trust, security and privacy	Security	Authorisation	Augmented Entities (Physical Entity + Virtual Entity)
UNI.408	FR	The system's services shall indicate what information can be found by a discovery/look-up service	Opting out of being found in a data search was indicated in the BRIDGE requirement list and also in the IoT-A Stakeholder Opinion Report. The BRIDGE requirement was "Data that companies are willing to provide to the Discovery Services are mainly URL addresses of databases / EPCIS repositories" Reference: [BRIDGE Deliverable "D2.1 Requirements document of serial level lookup service for various industries, Section C". http://www.bridge-project.eu/data/File/BRIDGE%20WP02%20Serial%20level%20lookup%20Requirements.pdf] [IoT-A Deliverable "D6.6 Report on	Deployment	Trust, security and privacy	Virtual Entity	VE Resolution	Services

			Stakeholder Opinions" http://www.IoT-A.eu/public/public-documents/documents-1/1/1/d6.6/at_download/file]					
UNI.409	FR	The system shall allow for storage of aggregation changes	This is a main functionality of the BRIDGE system which applies to RFID/assets tracked in the EPCGlobal framework Reference: [BRIDGE deliverable: "High level design for Discovery Services". http://www.bridge-project.eu/data/File/BRIDGE%20WP02%20High%20level%20design%20Discovery%20Services.pdf]	Functional	(none)	Virtual Entity	VE Service	Virtual Entity

UNI.410	FR	The Digital Entity History Storage shall be restricted in who can delete and update it	The integrity and trust in the history storage block depends on how "unaltered" it is. The BRIDGE project justifies the present use of the "history storage" component. They expressed it as "Discovery Service security policies may be set to restrict update and delete actions on DS records" Reference: [BRIDGE Deliverable "D2.1 Requirements document of serial level lookup service for various industries, Section C". http://www.bridge-project.eu/data/File/BRIDGE%20WP02%20Serial%20level%20lookup%20Requirements.pdf]	Functional	Trust, security and privacy	Virtual Entity	VE Service	Virtual Entity
UNI.411	FR	The system shall offer a unique identification of clients requesting data via the discovery/lookup services	BRIDGE mentioned that the unique client identification at the DS is required to control access to data stored on the DS (particularly EPC number and link).	Functional	Trust, security and privacy	Security	Authentication	Users (Human, Active Digital Entitity)
UNI.412	FR	Data owners should be able to set access-control rights/ policies (set up by data owners) to their data stored on resources	This addresses privacy by putting the control in the hands of the data owners (or certain external groups)	Functional	Trust, security and privacy	Security	Authorisation	Users (Human, Active Digital Entitity)
UNI.413	Design Constraint	Access-control rights/ policies (set up by data owners) shall not be published publicly.	Access control policies themselves, if known, can give away information.	Deployment	Trust, security and privacy	(none)	(none)	Resources (Computational Element for PE Access)

UNI.414	FR	The system shall enable the dynamic discovery of Virtual Entities and their services. This is to be done based on the specification of the service and the virtual enities.	Augmented entities are the core concept proposed for IoT and to enable applications that do not have to be a-priori configured for a fixed set of augmented entities, discovery at runtime must be possible.	Functional	(none)	Virtual Entity	VE resolution	Virtual Entity
UNI.415	FR	The system shall enable the dynamic discovery of Virtual Entities and their related services based on a geographical location	Geographic location is one of the most important aspects for finding relevant Virtual Entities. Spatial relations are of prime importance in the physical world.	Functional	(none)	Virtual Entity	VE resolution	Virtual Entity
UNI.416	FR	The system shall enable the lookup of service descriptions of specified services for a Virtual Entity with the VE identifier as key for the lookup	It is important to find the services related to a Virtual Entity that may provide information about the Virtual Entity, allow to actuate the Virtual Entity, or enable interaction with the Virtual Entity.	Functional	(none)	Virtual Entity	VE resolution	Virtual Entity
UNI.417	FR	The system shall enable the resolution of service identifiers to service locators	Due to the heterogeneity, dynamicity and mobility in the Internet of Things, the communication endpoint may change or different endpoints may be suitable for different applications. Therefore, services should be uniquely identified by a service identifier, but this identifier should not be used for locating the service, so a resolution step is necessary.	Functional	Evolution and Interoperability	IoT Service	IoT Service Resolution	Services (Intergation & Interoperability Layer)

UNI.418	FR	The system shall be able to discover dynamic associations between Virtual Entities and service related to Virtual Entities	Due to the mobility of Physical Entities as well as devices whose resources are accessible through services, changing services may provide information, allow actuation or enable interaction with Physical Entities. In order to provide the currently relevant services for a corresponding Virtual Entity, the dynamic assoications must be discovered	Functional	(none)	Virtual Entity	VE & IoT Service monitoring	Augmented Entities (Physical Entity + Virtual Entity)
UNI.419	FR	The system shall be able to track dynamic associations between a Virtual Entity and services related to the Virtual Entity. This need to be done in order to determine whether they are still valid.	Due to the mobility of things, as well as devices whose resources are accessible through services, changing services may provide information, allow actuation, or enable interaction with things. In order to provide the currently relevant services for a thing, dynamic assoications must be tracked to determine whether they are still valid.	Functional	(none)	Virtual Entity	VE & IoT Service monitoring	Augmented Entities (Physical Entity + Virtual Entity)
UNI.420	FR	The IoT system shall be able to discover dynamic associations based on geographic location and other context information.	Mobility is one of the key reasons for changing associations. By monitoring both the location of Physical Entities and the service area of resources, dynamic associations can be discovered. Based on the proximity of the Physical Entity, the service area of the resource and the functionality provided by the resource, it can be determined whether the resource can provide any information about the Physical Entity or enable any actuation on the Physical Entity. If this is the case, an association between the Virtual Entity, which represents the Physical Entity in the system, and the service, which makes the functionality of the resource accessible, can be established.	Functional	(none)	Virtual Entity	VE & IoT Service monitoring	Augmented Entities (Physical Entity + Virtual Entity)

UNI.421	FR	The system shall be able to track dynamic associations between a Virtual Entity and services based on geographic loaction to determine whether they are still valid.	Mobility is one of the key aspects for changing associations. By monitoring the location of Physical Entities, e.g., using location services, it can be determined when associations become invalid due to the geographic distance of Physical Entities and the service areas of resources and possibly other and possibly other aspects.	Functional	(none)	Virtual Entity	VE & IoT Service monitoring	Augmented Entities (Physical Entity + Virtual Entity)
UNI.422	Design Constraint	The system shall enable the discovery and lookup of associations across multiple administrative domains.	The Internet of Things will consist of multiple administrative domains with different owners that generally manage their devices, resources, services Virtual Entities etc. independently. To develop its full potential interactions, including lookup and discovery, across domain boundaries must be possible.	(none)	Evolution and Interoperability	Virtual Entity	VE Resolution	Virtual Entity
UNI.423	FR	When performing discovery, resolution or lookup, the system must respect any aspect of privacy, including the possibility to retrieve information about or related to people by using (or subverting the use of) the Internet of Things. In addition some services should be accessible in an anonymous way, while others might require an explicit authentication or authorization of the user.	Privacy is a key aspect for the IoT.	Functional	(none)	IoT Services, Virtual Entity, Security	IoT Service Resolution, VE Resolution, Identity Management	Services (Intergation & Interoperability Layer)
UNI.424	FR	The system must provide privacy protection for users accessing information about Physical Entities or services	For acceptance of the Internet of Things privacy during usage must be guaranteed	Functional	Trust, security and privacy	Security	Identity Management	User, Service

UNI.425	FR	The system shall provide a service identifier, and the identifier shall use a service/resource description for retrieval.	The system must consider the description of a service/resource for the semantic indexing on which the search will be performed	Functional	(none)	IoT Service	IoT Service Resolution	Services (Intergation & Interoperability Layer)
UNI.426	FR	The system shall be able to accept and manage semantic queries from the user and return Resources/Services	Iot Service Resolution functional component has interfaces to enable the user make queries for the discovery,lookup and resolution functions.	Functional	(none)	IoT Service	IoT Service Resolution	Services (Intergation & Interoperability Layer)
UNI.427	FR	The Discovery Service in a local search is required to find service/resource based on (rough) semantic description	Users must be able to discover Services locally in their environment. This is because in many cases users a) might not be able to leverage infrastructure services b) leveraging the Infrastructure would be ineffective and c) context-awareness would be higher if information is derived from local network (e.g. in an underground garage, proximity might be measured with higher accuracy using network metrics respect to using A-GPS or cell-based localization).	Functional	(none)	IoT Service	IoT Service Resolution	Resources (Computational Element for PE Access)
UNI.428	FR	The system shall provide a service that obtains unique identifiers for associations between VE and the service.	Association between Ves and the services is one of the key parameters for the resolution functional component and association contains unique association ID for example to manage it (such as delete, insert)	Functional	(none)	Virtual Entity	VE Resolution	Virtual Entity

UNI.429	FR	The IoT resolution component shall provide a service to insert or update the operational specifications (i.e. type, description, locator) of a new IoT service into the data base that is used for discovery, lookup, and resolution.	In order for lookup and global discovery to work properly, a IoT service-resolution component must provide a way to insert and update the description of services that it will then use as a search basis.	Functional	(none)	IoT Service	IoT Service Resolution, IoT Service	Service
UNI.432	FR	The system shall provide a virtual identification system.	A universal identifier should be defined as standard ID in order to map it to the specific ID used in every type of system (TCP/IP, RFID, ...)	Functional	Evolution and Interoperability	Virtual Entity	VE Resolution	Augmented Entities (Physical Entity + Virtual Entity)
UNI.501	NFR	The system shall make it difficult to spy on communicated messages.	The confidentiality of messages must be ensured.	(none)	Security & Privacy	Security	Key Exchange & Management, Authentication	Device Tag Gateway Infrastructure services Physical entity Storage Virtual entity
UNI.502	NFR	The device (contactless card for example) must not be activated without the consent of the owner. A device is always owned by a person or an entity. For example, in a retail use case, the owner of an RFID tag can be a retailer and after the checkout the new owner should be the client. The aim is to avoid skimming attacks	The unsolicited scanning of people shall be avoided.	(none)	Security & Privacy	Security	Authorisation	Device Tag User Physical entity Virtual Entity

UNI.503	NFR	It must be possible to change the owner of a device (tag for example). A device is always owned by a person or an entity. For example, in a retail use case, the owner of an RFID tag can be a retailer and after the checkout the new owner should be the client. The aim is to avoid skimming attacks	Privacy preserving solution in RFID requires to share a secret key between tag and reader (or owner since in this case, the owner enters his key in the reader). It must be possible to change this key in tag and reader (and even in the databases where the data related to the device is stored) if the owner has changed.	(none)	Security & Privacy	Security	Authorisation, Authentication, Key Exchange & Management	Device Tag Gateway Infrastructure services Physical entity Storage Virtual entity
UNI.504	NFR	The identifier of the device (ID of an RFID tag for example) must not be tracked by unauthorised entities. To preserve privacy, only the owner of the tag must be able to read it. So, authoized persons are the owner and the persons who are authorized by the owner. The "unauthorized entities" are all the other people.	The tracking of items and then people raise the problem of privacy	(none)	Security & Privacy	Security	Autorisation, Authentication, Key Exchange & Management	Device Tag User Physical entity Virtual Entity
UNI.505	FR	Connected devices shall be able to do energy harvesting, if needed	Maintain operation in environments where power supply is not possible	Functional	(none)	Management, Communication	Device Manager, Energy Optimization	Device
UNI.506	FR	Connecting devices shall be able to communicate with each other through the network by aid of standardised communication interfaces	Use of standard interfaces will enable take up of IOTA concept on the market	Functional	Evolution & Interoperability	Communication	Gateway	Device
UNI.507	NFR	Data security&privacy should be enabled at atomic level	Security in end-to-end communication does not address security issues pertaining to the device itself.	(none)	Security & Privacy	Security	Authorisation, Key Exchange & Management	Resource, Device
UNI.508	NFR	Communication with devices must be intermittent and command-based	Avoid traffic overhead	(none)	Evolution & Interoperability	Communication	Flow Control & Reliability, Energy Optimization	(none)

UNI.509	NFR	Each IoT device shall possess a universal ID, part of it read only and part of it read/write.	Enable object regonisition and setup/configuration in the context of applications development	(none)	Evolution & Interoperability	Communication	Routing & Addressing	Resource
UNI.510	FR	Atomic-level protocols must implement only functions related to data acquisition (e.g. DSP-level), crypto and security	Atomic-level protocols are the protocols realised to carry out a particular task related to device internal functions. E.g. how data are acquired from the environmnet. How they are encoded/encrypted for transportation of unreliable networks, etc. This requirements is needed to avoid overlap with user-level communication protocols.	Functional	(none)	(none specific)	(none specific)	Device
UNI.511	NFR	The system shall be scalable, that is, usable on very tight resources with potentially reduced features (e.g., the level of security may be different depending on the underlying hardware resources)	IoT faces many issues in that complex features have been to be made available on restrained devices (memory size, CPU speed, power consumption). For example, can an OS be run on something like an 8-bit CPU, 8 KB RAM, 64 KB flash platform? Or can use of symmetric crypto algorithms (e.g., AES) be run on resource-constraint platforms w/ AES co-processor functionality?	(none)	Performance and Scalability, Security	(none specific)	(none specific)	Device, Resource
UNI.512	NFR	The application shall share information about resource usage (for instance, when will the application need to transmit a message) with other functional layers.	IoT systems are often resource constrained, especially in terms of energy consumption. Optimum energy efficency can only be achieved by cross-functional-layer optimisation, which is dependent on application needs.	(none)	Performance and Scalability, Security	Communication	Energy Optimization	Device, Resource

UNI.601	NFR	The system shall guarantee infrastructure availability	The services provided by the infrastructure should always be available, as their operation is critical to the operation of the Internet of Things. Users should thus be able to reach the infrastructure. The infrastructure services should be able to operate.	(none)	Availability and Resilience, (Trust, security and privacy)	IoT Service, Security	IoT Service Resolution, IoT Service	
UNI.602	NFR	The infrastructure services shall be trustable	The services provided by the infrastructure Services should be trustworthy.	(none)	Trust, security and privacy	IoT Service, Security	IoT Service Resolution, IoT Service	User, Service
UNI.603	FR	The infrastructure services shall comply with the infrastructure service design and operate accordingly	Infrastructure Services should operate properly according to their design.	Operation, (Deployment)	(none)	IoT Service, Security	IoT Service Resolution	
UNI.604	NFR	A service shall always be accessible to entitled users	Access to the service shall be regulated by access policies. Users entitled in access policies to envoke a given service must be able to actually envoke it.	(none)	Availability and Resilience, (Trust, security and privacy)	IoT Service, Security	IoT Service Resolution, IoT Service	Service
UNI.605	NFR	The system shall support the reversing of the pseudonymization processes in order to guarantee mutual accountability	Some scenarios require Subjects to take responsibility for their actions. Some Services could be classified or critical for their provider and could require Users to take responsibility of their action. On the other hand Users might need providers to take responsibility for the Services they provide, because relying on such Services is critical for them. The IoT should support the reversing of the Pseudonymization processes.	(none)	Trust, security and privacy	IoT Service, Security	IoT Service, Identity Management	User, IoT Service

UNI.606	NFR	The system shall make the traceability of digital activities impossible	Subjects should not be able to track the digital activities of other subjects	(none)	Trust, security and privacy	IoT Service, Security	IoT Service Resolution, IoT Service, Identity Management, Authorization	User, Service
UNI.607	FR	The system shall provide communication confidentiality	The exchange of information between Subjects (Users and Services) should be understandable only for the intended recipients. This is a generic communication requirement because it can be achieved at different layers of the the stack.	Functional	(none)	IoT Service, Security	IoT Service, Key Exchange & Management	User, Service
UNI.608	FR	The system shall support communication integrity	The messages exchanged between subjects must be delivered in a complete and coherent way. It can affect communications at different layers (MAC, NWK, TRA).	Functional	(none)	Communication, Security	Error detection & correction, Key Exchange & Management	User, Service
UNI.609	NFR	The system shall ensure Data Freshness	The system should be protected from replay attacks (message replays at Service level, packet replay at network and link layer level).	(none)	Trust, security and privacy	Security	Key Exchange & Management	Device, Service
UNI.610	NFR	The system shall provide IoT-Service availability	Services providing access to Resources must be reachable by the Users who might need to rely on them. This requirement has a specific IoT declination as the resources of many nodes will be constrained and specific ways to protect from DoS or exhaustion attacks will be needed.	(none)	Availability and Resilience, (Trust, security and privacy)	Communication, IoT Service, Security	Flow Control & Reliability, IoT Service, Authentication, Authorization, Trust & Reputation	Service, Device
UNI.611	NFR	The system shall support access control mechanisms	The control of User access to Resources must be supported and, where needed, regulated by policies. Anonymous interaction must be supported and group authorization should be supported.	(none)	Trust, security and privacy	IoT Service, Security	IoT Service, Authorisation, Identity Management	User, Service

UNI.612	NFR	The system shall support subject authentication	Subjects (Users and Services) must be able to confirm the identity of other Subjects.	(none)	Trust, security and privacy	IoT Service, Security	IoT Service, Authentication, Identity Management	User, Service
UNI.613	FR	The system shall be able to meter service reputation	As there is a high chance of nodes being compromised due to their physical availability to malicious users, a secondary mechanism for establishing trust is needed.	Functional	(none)	IoT Service, Security	IoT Service, Trust & Reputation	Service
UNI.614	FR	The system shall provide Quality of Service	In networks where nodes are constrained devices with limited communication capabilities, QoS might have a new (or extended) meaning compared to the current meaning. For example, real-time, event-triggered data with high time resolution, needs to be delivered with a higher priority than other and might need to ignore the need to sleep of some devices in the network.	Operation, (Deployment), Functional	(none)	IoT Service, Communication, Management	IoT Service Resolution, IoT Service, QoS, QoS Manager	Device
UNI.615	NFR	The system shall provide transport layer fairness	While congestion avoidance is important in any large network, in low bandwidth mesh networks this is essential.	(none)	Perfomance and Scalability	Communication	Flow control and Reliability	Device
UNI.616	NFR	The system shall ensure network availability	The network functions should be available to network endpoints. Appropriate measures should be taken to avoid network disruption.	(none)	Availability and Resilience	Communication, Security	(none specific)	Device
UNI.617	NFR	The system shall enforce correct routing	Packet routing over underlying Link Layer should be efficient and should not be subject to disruption by malicious subjects. Disruption could lead to worm/blackhole, exhaustion and DoS attacks.	(none)	Trust, security and privacy	Communication, Security	Routing & Addressing, Trust Authority, Authentication	Device

UNI.618	NFR	The system shall have a communication control for restricted usage	In some cases hop by hop communication should only be available to authenticated devices.	Functional	Trust, security and privacy	Communication	Flow Control & Reliability, Authentication	Device
UNI.619	NFR	The system shall ensure non repudiation at network level	Mobile devices should be able to join peripheral networks belonging to different provider. Devices entitled to join a given network must be able to do so.	(none)	Trust, security and privacy	Security	Authorisation, Trust & Reputation, Authentication	Device
UNI.620	NFR	The system shall provide Software Integrity	The software execution environment should preserve software integrity.	(none)	Trust, security and privacy	Security	Certification Authority	Service
UNI.622	FR	The system shall support device location identification	A node that is considered fixed should not be moved from its position. This could alter the quality of the data provided as it refers to a different position.	Functional	Trust, security and privacy	Security	Trust & Reputation	Device
UNI.623	NFR	The system shall support location privacy	The Location of a Subject should only be available to authorized Subjects. Specific methods for obscuring both network and physical location should be available.	Functional	Trust, security and privacy	IoT Service, Virtual Entity, Security	IoT Service, Autorisation, VE Resolution, IoT Service Resolution	Service
UNI.624	NFR	The system shall provide pseudonymisation mechanisms	While complete anonimity is not feasibe in an IoT scenario, pseudonimity should be supported.	(none)	Trust, security and privacy	Security	Identity Management	Device, Service, User
UNI.625	FR	The system shall provide a device security and privacy measurement	Users should be able to monitor and control the security and privacy settings of all the devices that they own.	Functional	Trust, security and privacy	Security	(none specific)	Device

UNI.626	NFR	The IoT should support secure Over-the-Air/Over-the-Network Device Management	The Execution Environment and the Services provided on a given remote device should be securely managed from remote.	(none)	Trust, security and privacy	Security, Management	Authorisation, Authentication, Device Manager	Device

C Use cases, sequence charts and interfaces

In this Appendix, the system use cases, interaction diagrams and interface definitions for the different functionality groups and/or functional components are described according to the functional view of Section 4.2.2.

The first step in the modelling of the functional components was taken in D1.2 where system use cases were introduced. This document expands on this work and introduces interaction diagrams and interface descriptions.

The modelling is of course not complete yet and will be further expanded in D1.4 and D1.5 where it will be complemented with:

- System use cases covering the components not covered in this document.
- Interaction diagrams not covered in this document
- Interface definitions between functionality groups not covered in this document.

The remainder of this Appendix is organized as follows.

First, system use cases and interaction diagrams of the “IoT Business Process Management” and “Service Organisation” functionality groups are described. Next, system use cases, interaction diagrams and interface descriptions of the “IoT Service” and “Virtual entity” functionality groups will be given.

Finally, security is applied to some of the modelling above as an example.

C.1 IoT Business Process Management and Service Organisation

The modelling presented in this section demonstrates the two primary functional components provided by WP2, namely the Io Business Process Management and Service Organization. The former functional component is located at a higher level of abstraction and stems from the world of Business Process Management. Business processes are initially modelled at a business level and then executed in a concrete technical environment in which services are resolved at design or runtime that fulfil the process steps or activities outlined in the process model. This is where the second functional component that provides Service Composition and Orchestration comes into play, when services must be found and orchestrated in order to execute business steps.

The modelling depicted on the next pages show the mechanics of these two functions.

C.1.1 IoT Business Process Management

C.1.1.1 Use Cases

The Process Execution diagram (see Figure 48) illustrates how a process model is created by a modelling application and then serialized and deployed to different execution platforms. The Isis Platform and Real World Integration Platform (RWIP) are outlined as concrete examples, as these execution platforms are used as background IP in the IoT-A project.

The use cases in the Business Process Execution component typically follow this usage pattern:

1. A domain expert starts with modelling a business process in a dedicated modelling application. While such an application for modelling IoT-Aware processes is not strictly part of the IoT-A reference architecture, IoT-A will provide a respective tool within WP2

as Deliverable D2.5. The graphical modelling environment provides stencils and other components following the IoT-A concepts of an entity based domain model as it is outlined in this deliverable. Stencils are graphical shapes commonly found in CAD applications that can frequently also be provided separately from websites such as http://www.bpm-research.com/downloads/bpmn-stencils/.

2. The graphical model is then serialized to an executable form. The preliminary analysis of process execution languages and notations that is part deliverable D2.2 indicates that BPMN2.0 will most probably be the preferred output format for IoT-Aware processes. A technical expert will use this serialization to deploy the process to an execution environment in which the process is to be run.

3. The actual process execution is then IoT-specific in the sense that it delegates certain activities or process steps to IoT execution platforms such as RWIP or Isis. What happens there is that service capabilities and activity requirements are aligned in order to allow for choosing appropriate services that are capable of providing the required IoT-specific service qualities, such as e.g. a process might require a certainty of information provided by a sensor service of at least 80%, so that only a subset of the available sensor services might be suitable for executing the process. At this stage, the Service Composition and Orchestration components within the Service Organisation FG become relevant, as certain quality and capability parameters might not be met by individual services, but only by an orchestration of individual services. The lower part of the diagram is thus explained in more detail with the next figure in the next section.

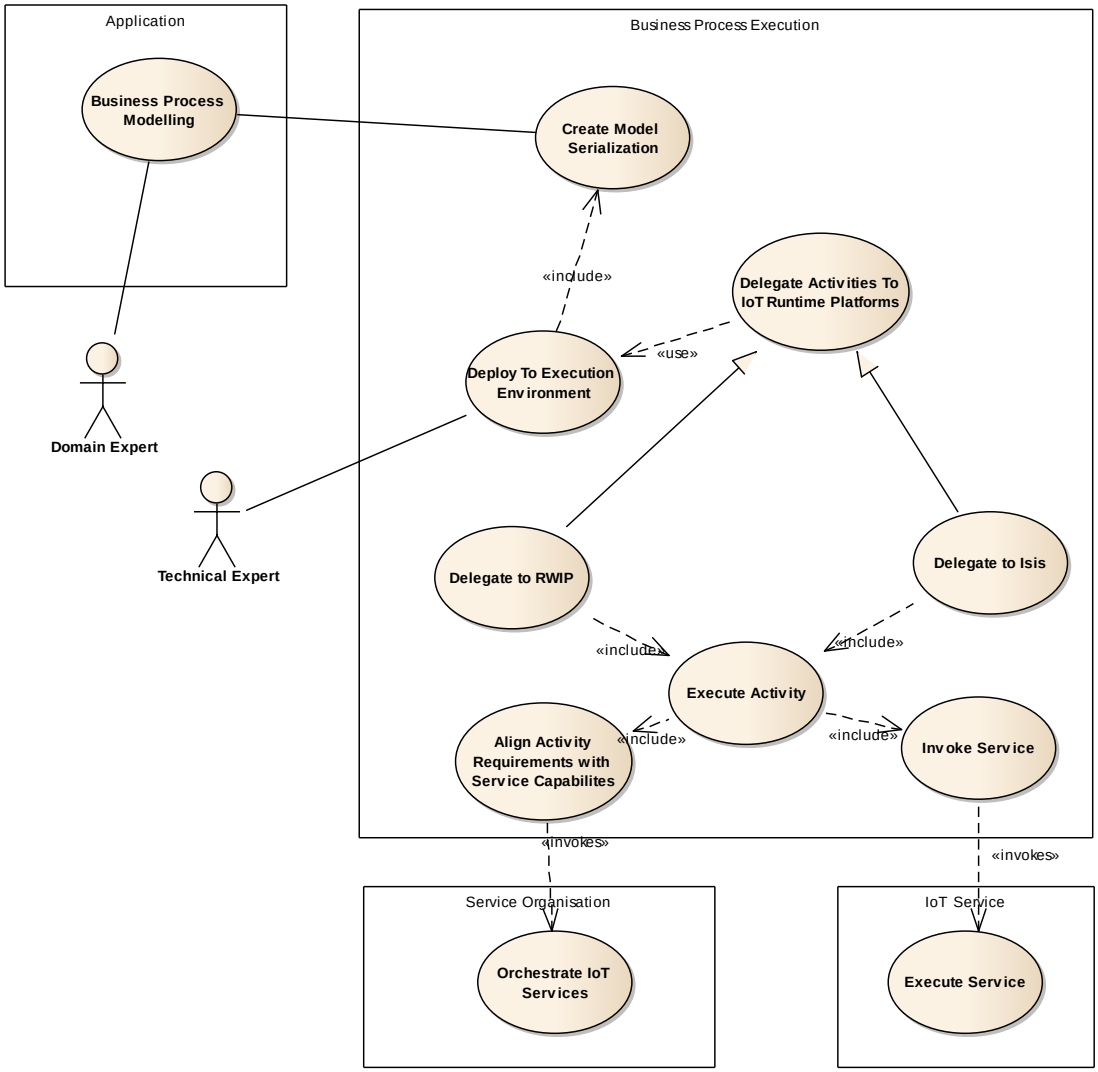

Figure 47: Business Process Execution.

C.1.2 Service Organisation

C.1.2.1 Use Cases

When the individual processes, steps or activities need to be executed, the Service Organisation diagram gains the focus. Here, the detailed and principal steps for service composition are shown in a domain agnostic way. The general principle is always that a mapping of services and Virtual Entities (VEs) must be found by aligning information from Virtual Entity Resolution and IoT Service Resolution and that these services are then orchestrated.

As we typically enter the Service Composition and Orchestration components from the Business Process Execution component (i.e. services are typically orchestrated in the execution of a process activity), the Process Execution component is shown as an actor starting the composition activities. The diagram features the relationships to other functional components,

which we briefly outline below. For this section we however focus on the main responsibilities of the component:

- Increase Quality of Information
  Service Composition can increase the Quality of Information by fusing information from different sources. This relates to the example given in the previous section. While a single sensor service might not be able to guarantee a certain level of accuracy for the respective sensor information, fusing several similar services might increase information quality considerably, as errors are mitigated and faulty sensors compensated.

- Support semantic Service Composition
  IoT Services can be composed of other services providing higher level functionality. The composition is flexible because the composition is not made of particular services but of services able to provide equal functionality. If one service fails it can be replaced by a similar one. This is closely related to the previous aspect and actually further contributes to an increase of information quality, as dynamic changes in the available services are taken into account.

- Orchestrate IoT Services
  The Business Process Execution component delegates service orchestration (executing the services appropriate to the process activity) to the Service Composition and Orchestration function. This is the actual interface between the process execution and the service resolution infrastructure (the latter being discussed in the following sections). In essence, the Process Execution component conveys the service requirements needed for executing the respective activity to the Service Composition and Orchestration component which in turn utilizes the IoT Service Resolution in order to Service Manager in order find and resolve appropriate services and create a suitable orchestration from them, if necessary. Then, the Service Manager is used for actually invoking the services and eventually delivering service results back to the Process Execution component.

The most important link to other functional components relates to the IoT Service resolution in a first step of finding respective services appropriate for being bound to the process execution as part of composite services. Here, service identifiers are both resolved to URLs used for executing them, but also service descriptions / capabilities are evaluated in terms of matching the requirements of the process within a service composition. Before the individual services resolved in the IoT Service resolution are actually executed or subscribed to (this depends on the nature of the service), the Virtual Entity resolution comes into play, as – in accordance with the domain model – the respective services usually need to target a specific entity, so that the associations between entities and services must be evaluated, for which the Virtual Entity resolution is utilized. It must be noted that it is important to look for the Virtual Entity and the required aspect first (instead of potential services), thereby finding possible services and only then they would be picked up in the IoT Service Resolution. Trying to discover services first and then match them with the result of the VE Discovery/Look-up may result in a huge amount of services in the first step that then need to be discarded as they pertain to the wrong Virtual Entities. This “early exclusion” principle is well known in the query optimizers of traditional database systems and is relevant here as well.

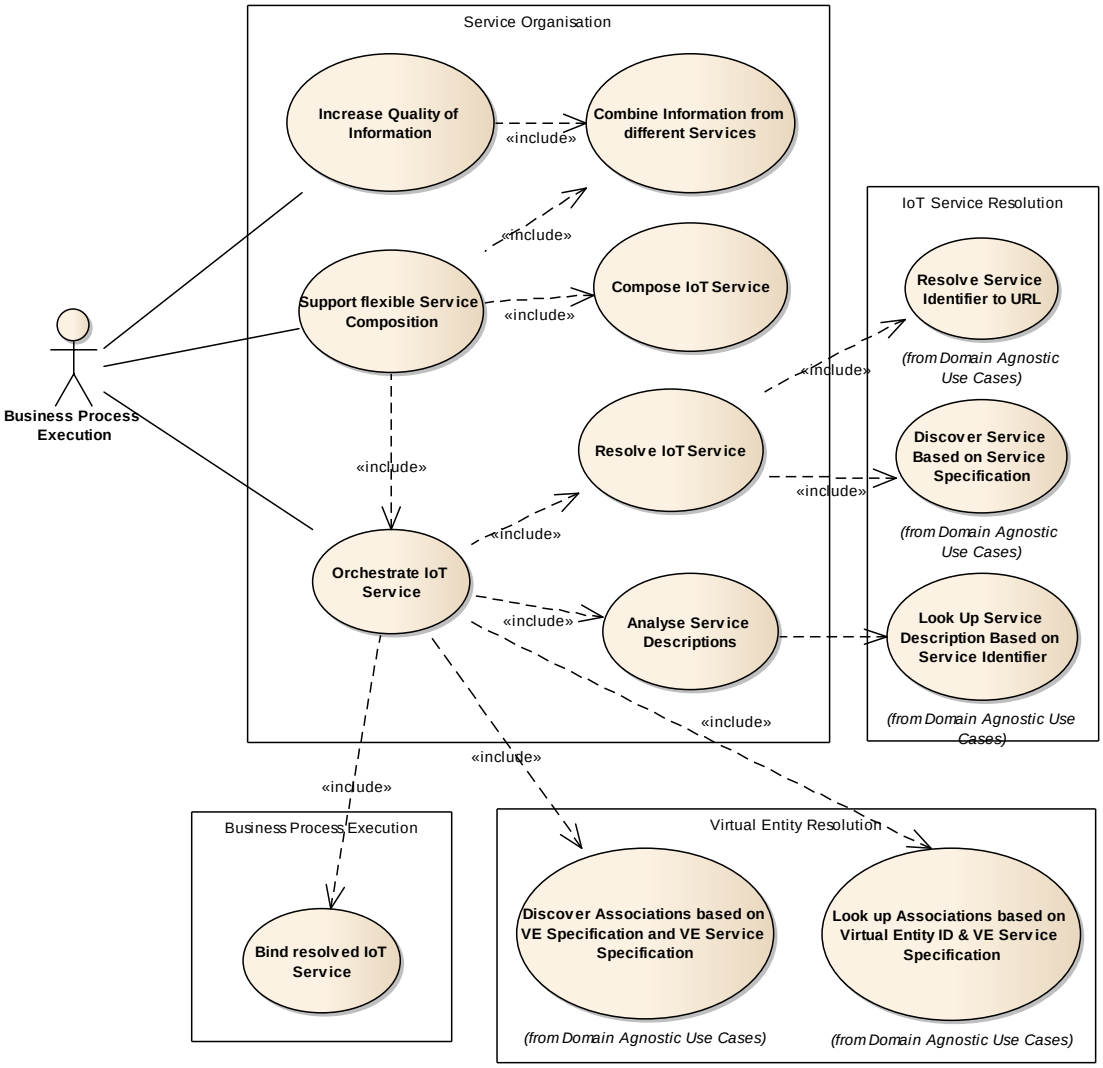

Figure 48: Service Organization.

C.1.2.2 Interaction Diagrams

The Interaction diagram related to the use cases of the Service Composition and Orchestration functional component in the Service Organisation are depicted below. They are not yet complete, as e.g. Virtual Entity Resolution is not involved yet, although, as discussed in the previous section, it is vital for the formulation of abstract service requests addressing properties of entities without knowing concrete Iot services associated to these entities a priori.

Interaction Diagram: Orchestrate Service

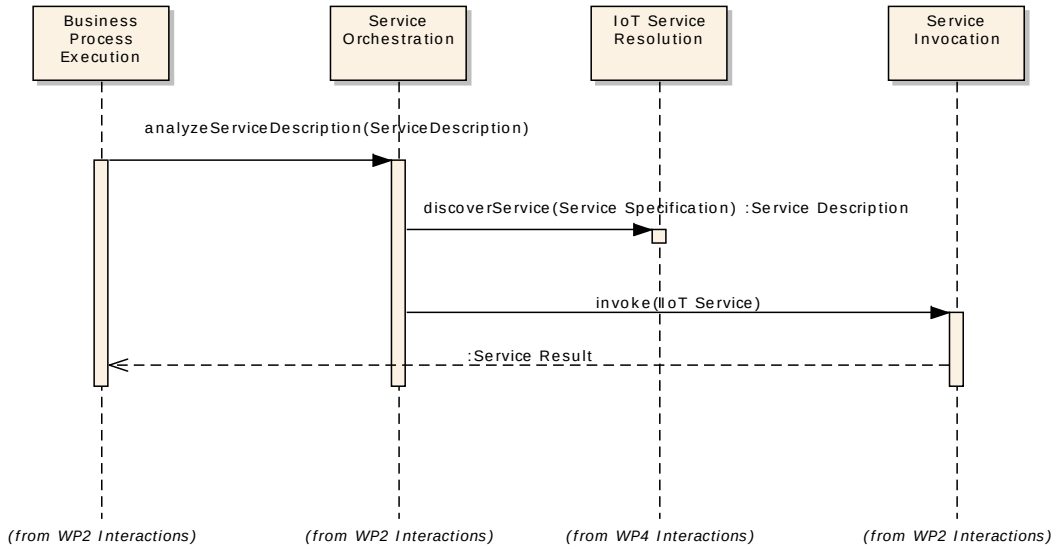

Figure 49: Orchestrate Service.

Interaction Diagram: Decompose Composite Service

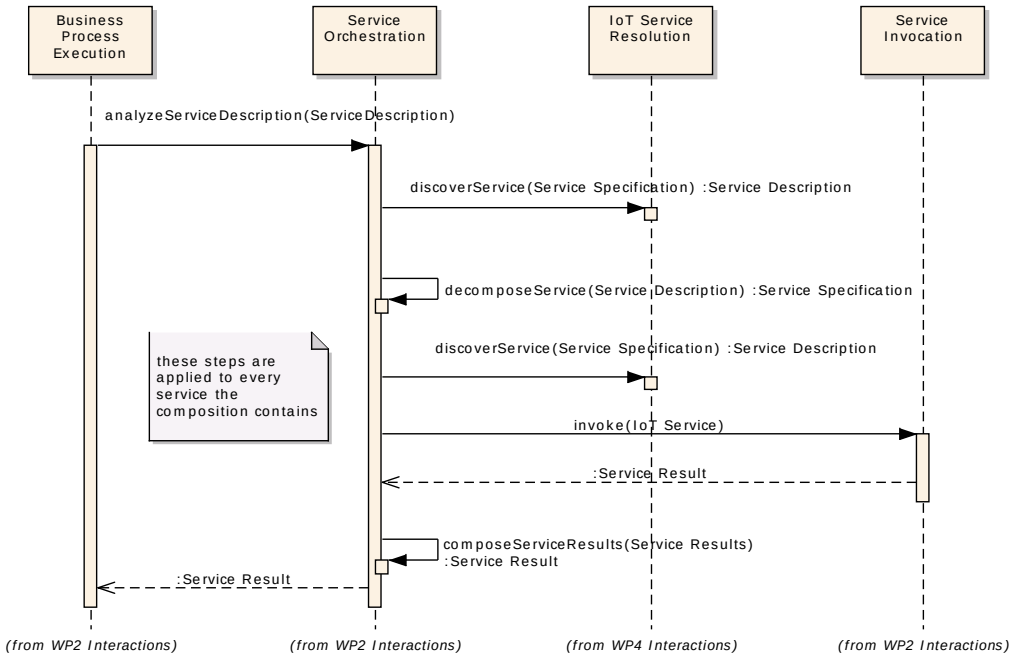

Figure 50: Decompose Composite Service.

C.2 IoT Services

C.2.1 IoT Service Resolution functional component

C.2.1.1 Use Cases

The use cases of this section cover the IoT Service Resolution functional component as identified in the functional view (see Section 4.2.2.5). They provide a service/resource abstraction level, i.e., service descriptions can be discovered and looked up, but there is no relation to Virtual Entities (and thus Physical Entities) being modelled. Associations between Virtual Entities and services are handled by the Resolution of Virtual Entities component (see Section 4.2.2.4).

The following use cases are depicted in Figure 51.

- Resolve Service Identifier to URL/Address
  - The use case is initiated by a user of the system, i.e., a Human User or an Digital Artefact. The user wants to have the URL or address of a service for interacting with the service.
  - The assumption is that the user already knows a unique identifier of the service.
  - In this use case, the IoT Service Resolution resolves the service identifier to a URL or address.
  - If the resolution step is successful, the user can contact the service.
- Subscribe to resolution of Service Description based on Service Identifier

- The use case is initiated by a user of the system, i.e., a Human User or an Active Digital Artefact. The user wants to be asynchronously notified about the URL or address of a service for interacting with the service. A new notification will be sent whenever the URL of the service changes.
- The assumption is that the user already knows a unique identifier of the service.
- In this use case, the IoT Service Resolution asynchronously notifies the subscribing user about the URL and sends a new notification whenever the URL changes.
- If the subscription is successful, the user will always receive the current URL for contacting the service.

- Unsubscribe to resolution of Service Description
  - The use case is initiated by a user of the system. The user has previously subscribed to receive notifications about the current URL of a service identified by a service identifier.
  - The assumption is that the user knows the subscription identifier of the subscription assigned by the IoT Service Resolution.
  - In this use case, the subscription to the IoT Service Resolution identified by the subscription identifier is cancelled.
  - If the unsubscription is successful, the user will no longer receive notifications concerning the URL of the identified service.
- Look up service description based on Service Identifier
  - This use case is initiated by a user of the system. The user wants to have a full description of the service, including a description of the interface and the URL or address for interacting with the service.
  - The assumption is that the user already knows a unique identifier of the service.
  - In this use case, the IoT Service Resolution looks up the service description based on the service identifier. The service description contains all information necessary for interacting with the service (including URL). This interaction is then based on service identifier.
  - If the lookup step is successful, the user has all the information needed for interacting with the service.
- Subscribe to look-up of Service Description based on Service Identifier
  - The use case is initiated by a user of the system, i.e., a Human User or an Active Digital Artefact. The user wants to be asynchronously notified about the service description of a service, which includes a description of the URL or address for interacting with the service. A new notification will be sent whenever the service description of the service changes.
  - The assumption is that the user already knows a unique identifier of the service.
  - In this use case, the IoT Service Resolution asynchronously notifies the subscribing user about the service description and sends a new notification whenever the service description changes.

- If the subscription is successful, the user will always receive the current service description of the service.

- Unsubscribe to look-up of Service Description
  - The use case is initiated by a user of the system. The user has previously subscribed to receive notifications about the current service description of a service identified by a service identifier.
  - The assumption is that the user knows the subscription identifier of the subscription assigned by the IoT Service Resolution.
  - In this use case, the subscription to the IoT Service Resolution identified by the subscription identifier is cancelled.
  - If the unsubscription is successful, the user will no longer receive notifications concerning the service description of the identified service.

- Discover service based on service specification
  - This use case is initiated by a user of the system. The user wants to discover a service that can provide certain functionality.
  - The assumption is that the user knows what kind of service he needs, but does not know the specific service instances available.
  - In this use case, the IoT Service Resolution discovers services that fit the service specification, which can contain information about the type of service, its requirements, and also scope information, e.g., the geographic area for which the service provides information.
  - If the discovery step is successful, i.e., services fitting the specification are found, the user gets the service descriptions of these services.

- Subscribe to discovery of Service Descriptions based on Service Specification
  - The use case is initiated by a user of the system, i.e., a hHuman User or an Active Digital Artefact. The user wants to be asynchronously notified about the service descriptions of all services fitting a given service specification. A new notification with a service description will be sent whenever a service description changes, a new service description fitting the service specification has become available or when a fitting service description was deleted from the IoT Service Resolution.
  - The assumption is that the user knows what kind of service he needs, but does not know the specific service instances currently available.
  - In this use case, the IoT Service Resolution asynchronously notifies the subscribing user about the service descriptions fitting the give service specification and sends a new notification whenever a service description changes, a new service description fitting the service specification has become available or when a fitting service description was deleted from the IoT Service Resolution..
  - If the subscription is successful, the user will always be informed about changes in the service descriptions fitting the provided service specification.

- Unsubscribe to discovery of Service Descriptions

- The use case is initiated by a user of the system. The user has previously subscribed to receive notifications about all the service descriptions fitting a given service specification.
- The assumption is that the user knows the subscription identifier of the subscription assigned by the IoT Service Resolution.
- In this use case, the subscription to the IoT Service Resolution identified by the subscription identifier is cancelled.
- If the unsubscription is successful, the user will no longer receive notifications concerning service descriptions fitting the given service specification.

- Manage service resolution and service descriptions (insert, update, delete)
  - This use case is initiated by a service (or an entity managing a service).
  - The assumption is that a service description needs to be inserted, updated or deleted due to a new service becoming available, an aspect of a service changing (e.g. due to mobility), or a service no longer being available.
    - This use case is about the management of service descriptions in the IoT-service resolution, and the association of service identifiers to URLs / addresses.
    - The service (or an entity-managing a service) inserts a new service description, so that it can be looked up and discovered and so that the service identifier can be resolved as a URL/address.
    - The service (or an entity managing a service) updates an existing service description, which may include the update of the mapping of a service identifier to a URL/address.
    - The service (or an entity managing a service) deletes an existing service description, so that a service is no longer available.
    - If the management of a service description is successful, the service descriptions can be looked up or discovered, and/or reflect the status as reported by the services.

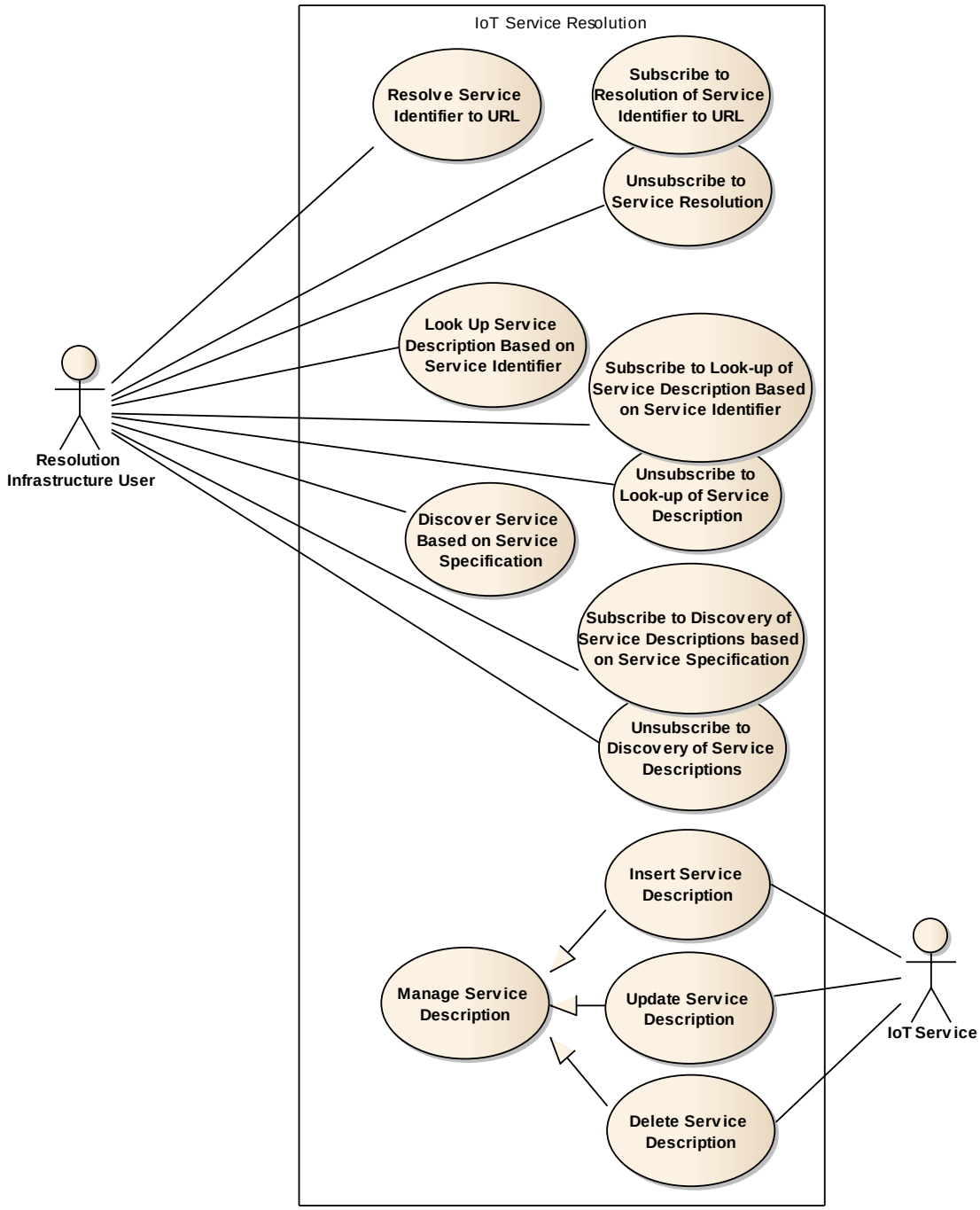

Figure 51: Use case IoT Service Resolution.

C.2.1.2 Interaction Diagrams

The Interaction diagram related to the use cases of the IoT Service Resolution functional component are depicted below.

Interaction Diagram: Resolution

For the resolution of a Service Identifier to the URL through which the service can currently be accessed, an IoT-Service Client synchronously calls the IoT Service Resolution component, using the resolveService operation with the ServiceID of the service as parameter. The IoT Service Resolution resolves the ServiceID, providing the requested URL as the return value.

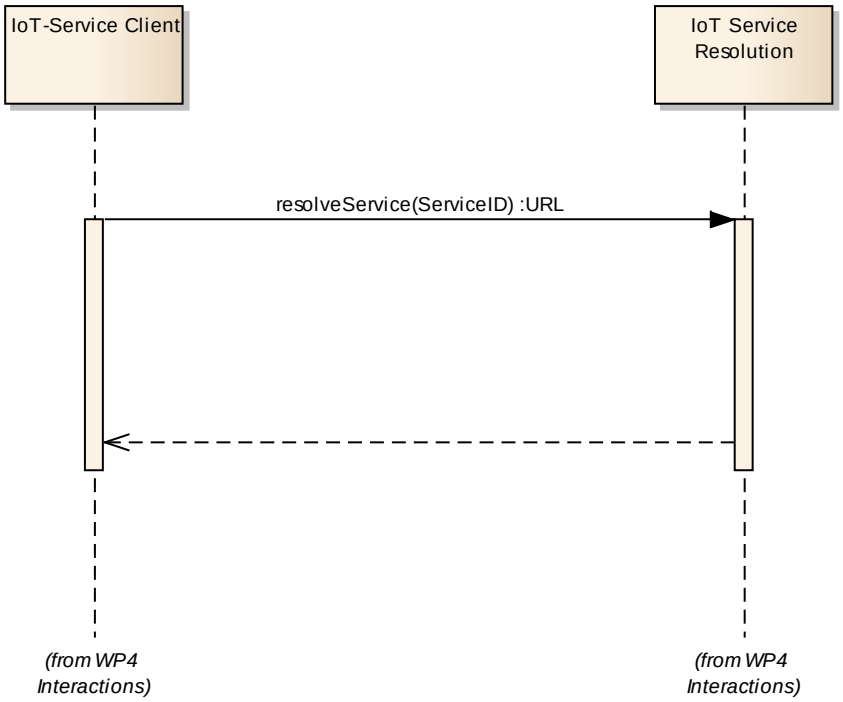

Figure 52: Resolve Service Identifier to URL.

Interaction Diagram: Subscribe to Resolution

For subscribing to asynchronously receive notifications about the current service URL of a service identified by its service identifier, an IoT Service Client synchronously calls the IoT Service Resolution, using the subscribeServiceResolution operation with the Service ID of the service and the notification callback, to which notifications are to be sent, as parameters. The notification callback identifies the endpoint on the IoT Service Client side that implements the notifyServiceResolution operation. The IoT Service Resolution returns the subscription identifier that can be used to map an incoming notification to the subscription it belongs to.

Subsequently, the IoT Service Resolution will call the notifyServiceResolution operation of the IoT Service client, providing the service URL and the subscription ID as parameters.

When the IoT Service Client is no longer interested in receiving notifications pertaining to the subscription, it will call the unsubscribeServiceResolution operation of the IoT Service

Resolution using the subscription identifier as parameter. As a result, the IoT Service Resolution will stop sending notifications pertaining to the identified subscription.

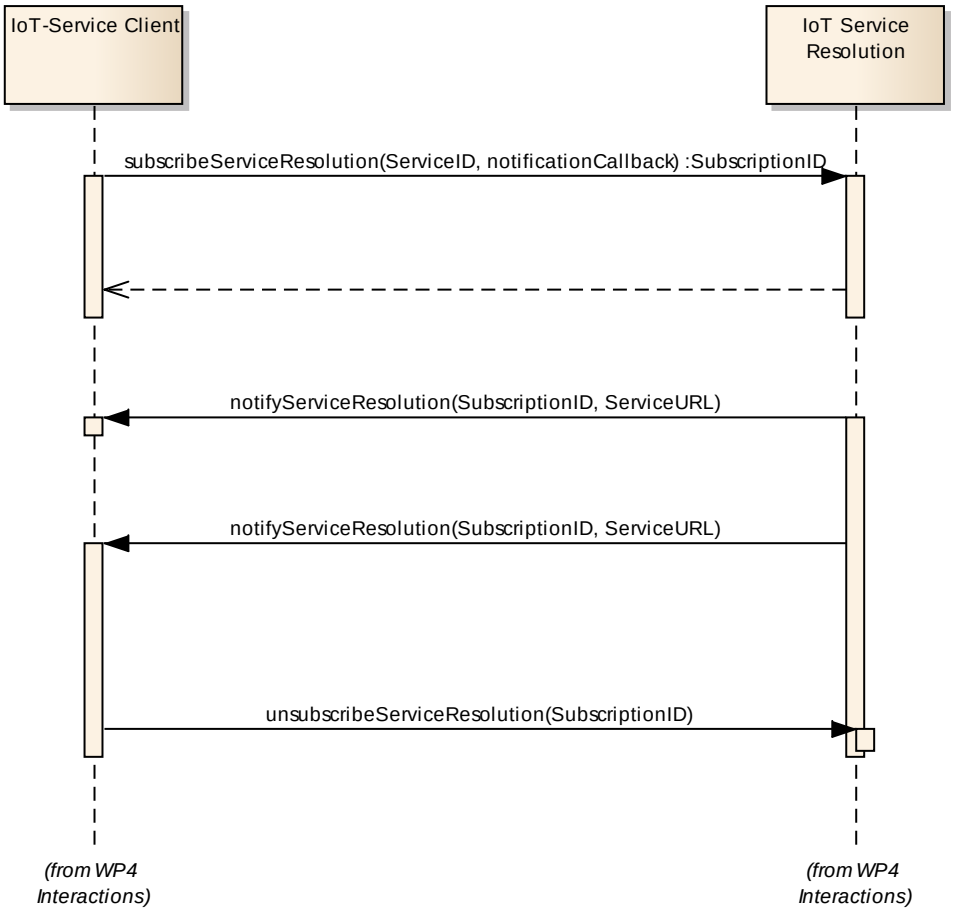

Figure 53: Subscribe Resolution of Service Identifier to URL

Interaction Diagram: Lookup

For the lookup of a Service Description based on a Service Identifier, an IoT Service Client synchronously calls the IoT Service Resolution component, using the lookupService operation with the ServiceID of the service as parameter. The IoT Service Resolution looks up the Service Description based on the ServiceID and provides it as the return value.

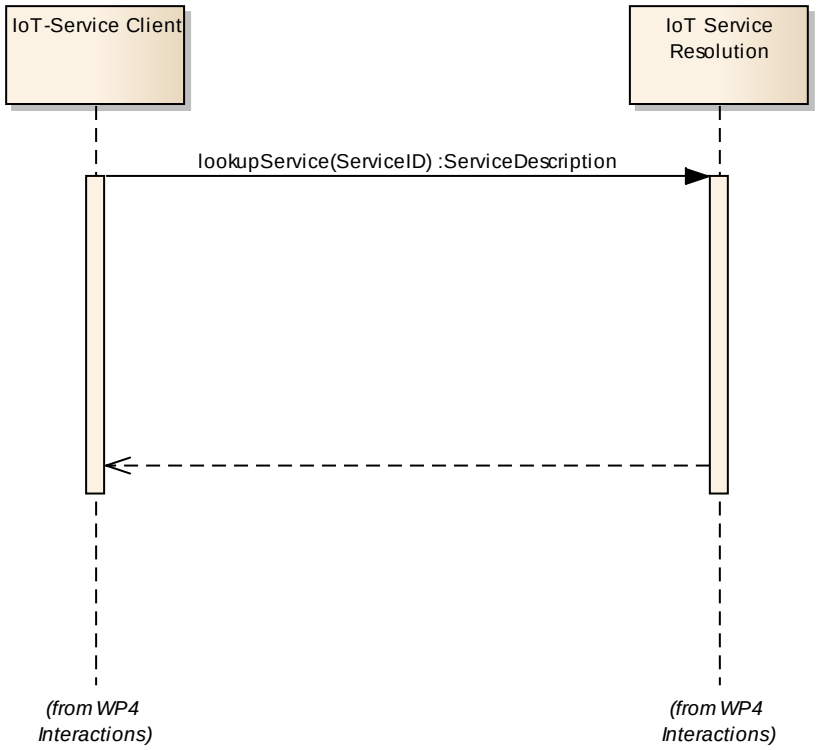

Figure 54: Lookup Service Description based on Service Identifier.

Interaction Diagram: Subscribe to Lookup

For subscribing to asynchronously receive notifications about the current service description of a service identified by its service identifier, an IoT Service Client synchronously calls the IoT Service Resolution, using the subscribeServiceLookup operation with the Service ID of the service and the notification callback, to which notifications are to be sent, as parameters. The notification callback identifies the endpoint on the IoT Service Client side that implements the notifyServiceLookup operation. The IoT Service Resolution returns the subscription identifier that can be used to map an incoming notification to the subscription it belongs to.

Subsequently, the IoT Service Resolution will call the notifyServiceLookup operation of the IoT Service client, providing the service description and the subscription ID as parameters.

When the IoT Service Client is no longer interested in receiving notifications pertaining to the subscription, it will call the unsubscribeServiceLookup operation of the IoT Service Resolution using the subscription identifier as parameter. As a result, the IoT Service Resolution will stop sending notifications pertaining to the identified subscription.

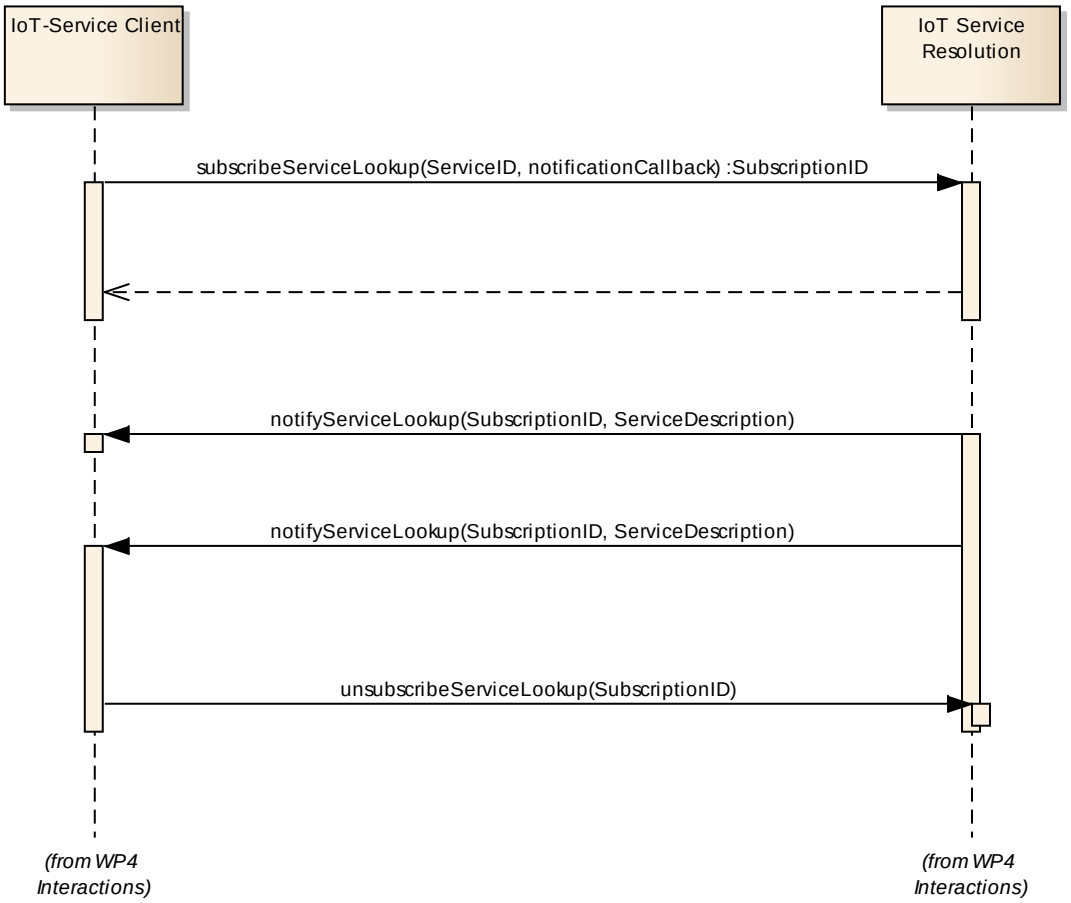

Figure 55 Subscribe Look-up of Service Description based on Service Identifier

Interaction Diagram: Discovery

For the discovery of suitable services, an IoT Service Client synchronously calls the IoT Service Resolution component, using the discoverService operation with the ServiceSpecification of the service as parameter. The ServiceSpecification contains a specification of aspects the service must fulfil, i.e., the type of service, the output to be provided, the post-conditions that need to be valid, the inputs to be provided, the pre-conditions required, the geographical location covered etc. [The details of the service specification have not been defined yet and will be covered as part of future work.] The IoT Service Resolution finds the Service Descriptions fitting the Service Specification and returns them in an array.

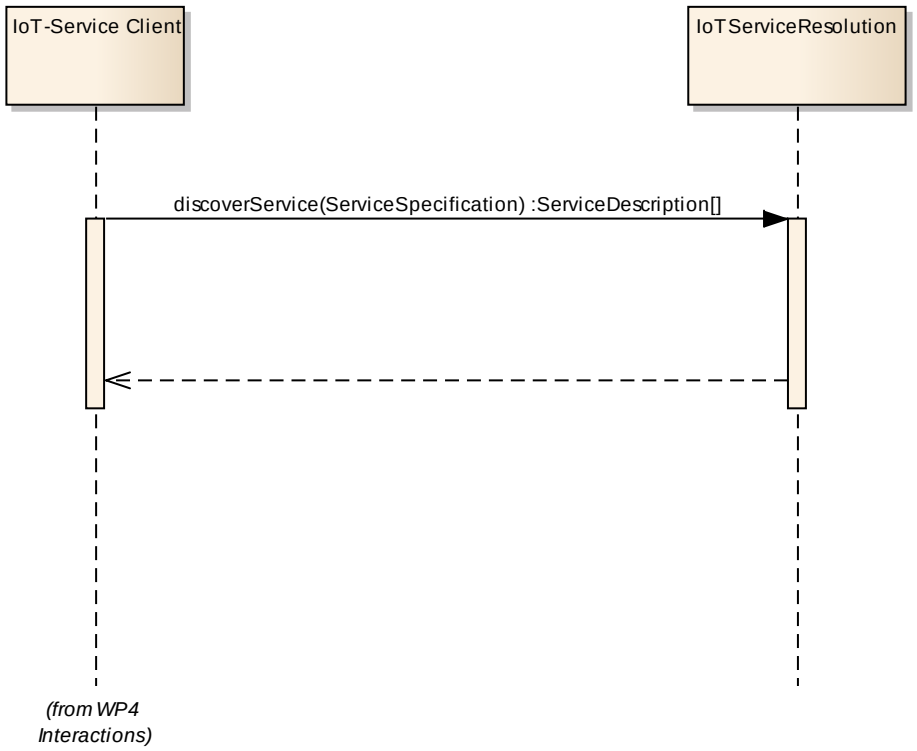

Figure 56: Discover Service based on Service Specification.

Interaction Diagram: Subscribe to Discovery

For subscribing to asynchronously receive notifications about the current service descriptions of services fitting a given service specification, an IoT Service Client synchronously calls the IoT Service Resolution, using the subscribeServiceDiscovery operation with the service specification of the service and the notification callback, to which notifications are to be sent, as parameters. The notification callback identifies the endpoint on the IoT Service Client side that implements the notifyServiceDiscovery operation. The IoT Service Resolution returns the subscription identifier that can be used to map an incoming notification to the subscription it belongs to.

Subsequently, the IoT Service Resolution will call the notifyServiceDiscovery operation of the IoT Service client, providing the service descriptions and the subscription ID as parameters. A notification will be sent whenever a previously provided service description changes or is

deleted. A notification will also be sent if a new service description fitting the given service specification becomes available.

When the IoT Service Client is no longer interested in receiving notifications pertaining to the subscription, it will call the unsubscribeServiceDiscovery operation of the IoT Service Resolution using the subscription identifier as parameter. As a result, the IoT Service Resolution will stop sending notifications pertaining to the identified subscription.

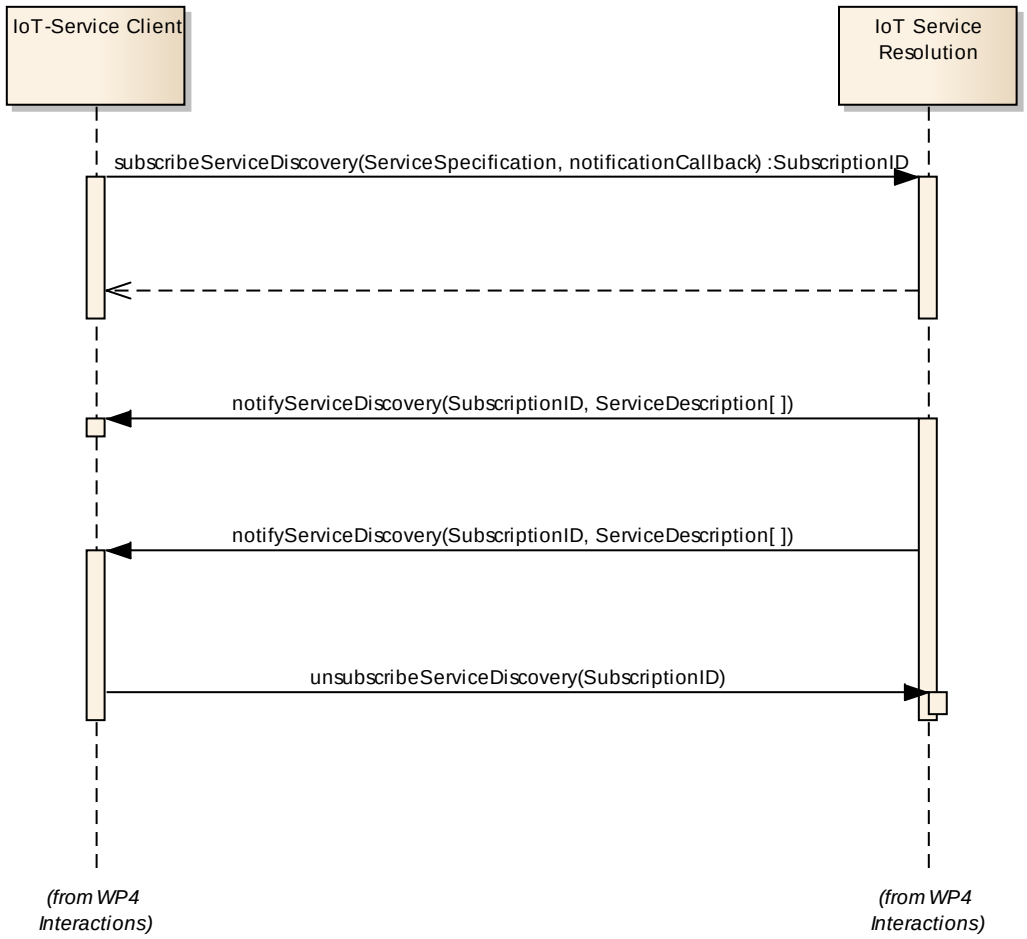

Figure 57 Subscribe Discovery of Service Descriptions based on Service Specification

Interaction Diagram: Insert

An IoT Service inserts its Service Description into the IoT Service Resolution component. The Service synchronously calls the IoT Service Resolution component using the insertServiceDescription operation with its ServiceDescription as parameter. The IoT Service Resolution component inserts the Service Description into its internal information base and returns the ServiceID that uniquely identifies the stored Service Description. As a result, the information required for resolution, lookup and discovery can efficiently be found.

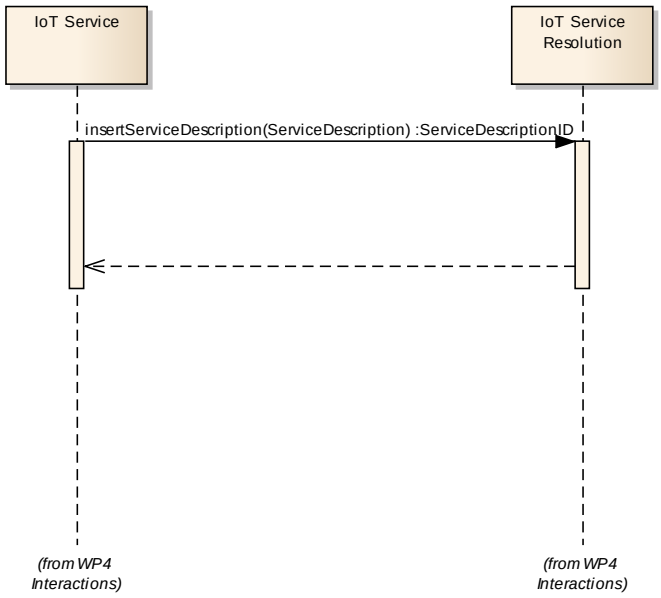

Figure 58: Insert Service Description.

Interaction Diagram: Update

An IoT Service updates its Service Description in the IoT Service Resolution component. The Service asynchronously calls the IoT Service Resolution component using the updateServiceDescription operation with its updated ServiceDescription as parameter. The IoT Service Resolution component updates the Service Description in its internal information base, so the updated information required for resolution, lookup and discovery can efficiently be found. The call to updateServiceDescription always returns with an OK status code, as the processing of the ServiceDescription is done asynchronously.

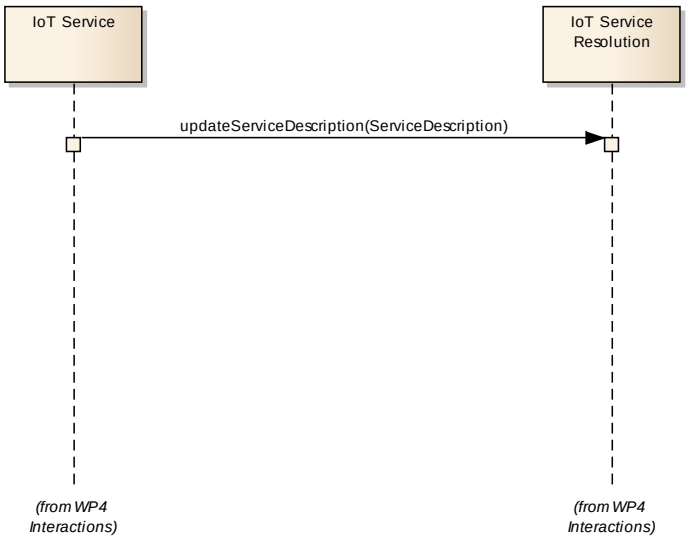

Figure 59: Update Service Description.

Interaction Diagram: Delete

An IoT Service deletes its Service Description from the IoT Service Resolution component. The Service asynchronously calls the IoT Service Resolution component using the deleteServiceDescription operation with the ServiceID (which is part of the ServicDescription) as parameter. The IoT Service Resolution component deletes the Service Description identified by the ServiceID from its internal information base. The call to deleteServiceDescription always returns with an OK status code, as the processing of the deletion is done asynchronously.

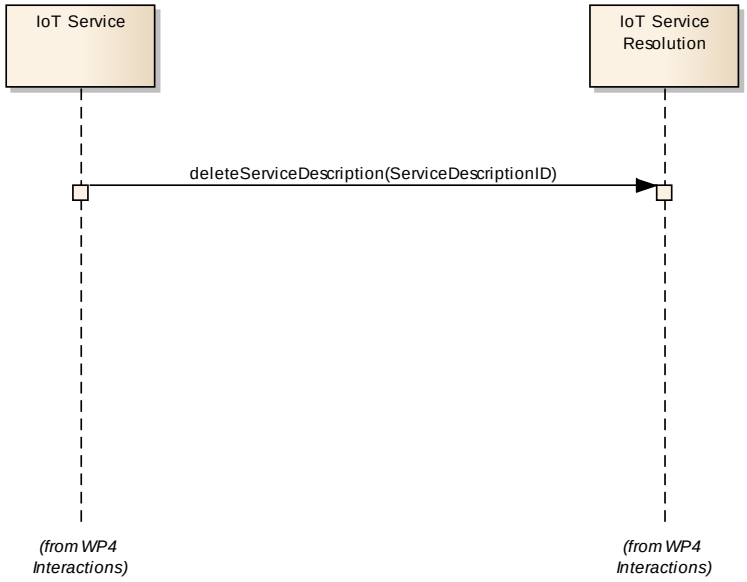

Figure 60: Delete Service Description.

C.2.1.3 Interface definitions

In this subsection we present the operations of the IoT Service Resolution functional component, i.e., resolve service, look up service, discover service, insert service, update service and delete service.

Interface Definition: Resolve Service

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Resolve Service Identifier to URL" Use Case	IoT Service Client	IoT Service Resolution	resolveService: given the ServiceID provide the URL required for accessing the service	ServiceID	ServiceID available

Output:

Functionality Output	Impacted Components	Return value	Exception
URL of the Service	-	Service URL	Service URL not available

Interface Definition: Subscribe Service Resolution

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Subscribe to Resolution of Service Identifier to URL" Use Case	IoT Service Client	IoT Service Resolution	subscribeService-Resolution: given the ServiceID asynchronously notify the IoT Service Client about the service URL required for accessing the service as a result of the subscription and on any change	ServiceID Notification-Callback	ServiceID available Notification-Callback and notifyService-Resolution implemented on the IoT Service Client side

Output:

Functionality Output	Impacted Components	Return value	Exception
Subscription identifier	-	SubscriptionID	Subscription failed

Interface Definition: Notify Service Resolution

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Subscribe to Resolution of Service Identifier to URL" Use Case	IoT Service Resolution	IoT Service Client	notifyService-Resolution: the current service URL required for accessing the service is provided. The subscription to which the notification pertains is identified by the subscription identifier	SubscriptionID, ServiceURL	Notification-Callback available and IoT Service Client reachable using the Notification-Callback

Interface Definition: Unsubscribe Service Resolution

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Unsubscribe to Service Resolution" Use Case	IoT Service Client	IoT Service Resolution	Unsubscribe-Service-Resolution: given the SubscriptionID cancel the respective subscription	SubscriptionID	SubscriptionID available IoT Service Resolution reachable

Interface Definition: Lookup Service

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Look Up Service Description Based On Service Identifier" Use Case	IoT Service Client	IoT Service Resolution	lookupService: given the ServiceID provide the Service Description of the service	ServiceID	ServiceID available

Output:

Functionality Output	Impacted Components	Return value	Exception
Service Description of the Service	-	ServiceDescription	Service Description not available

Interface Definition: Subscribe Service Look-up

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Subscribe to Look-up of Service Description based on Service Identifier" Use Case	IoT Service Client	IoT Service Resolution	subscribeService-Lookup: given the ServiceID asynchronously notify the IoT Service Client about the service description as a result of the subscription and on any change of the service description	ServiceID Notification-Callback	ServiceID available Notification-Callback and notifyService-Lookup implemented on the IoT Service Client side

Output:

Functionality Output	Impacted Components	Return value	Exception

Subscription identifier	-	SubscriptionID	Subscription failed

Interface Definition: Notify Service Look-up

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Subscribe to Look-up of Service Description based on Service Identifier" Use Case	IoT Service Resolution	IoT Service Client	notifyService-Lookup: the current service description is provided. The subscription to which the notification pertains is identified by the subscription identifier	SubscriptionID, ServiceDe-scription	Notification-Callback available and IoT Service Client reachable using the Notification-Callback

Interface Definition: Unsubscribe Service Look-up

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Unsubscribe to Look-up of Service Description" Use Case	IoT Service Client	IoT Service Resolution	Unsubscribe-Service-Lookup: given the SubscriptionID cancel the respective subscription	SubscriptionID	SubscriptionID available IoT Service Resolution reachable

Interface Definition: Discover Service

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Discover Service Based On	IoT Service Client	IoT Service Resolution	discoverService: given the Service	Service Specification	Service Specification

Service Specification" Use Case			Specification provide the Service Descriptions of fitting services		available

Output

Functionality Output	Impacted Components	Return value	Exception
Service Descriptions of Services fitting the Service Specification	-	Array of ServiceDescription	- [no fitting services is a normal case]

Interface Definition: Subscribe Service Discovery

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Subscribe to Discovery of Service Descriptions based on Service Specification" Use Case	IoT Service Client	IoT Service Resolution	subscribeService-Discovery: given a service specification asynchronously notify the IoT Service Client about the fitting service descriptions as a result of the subscription and on any change regarding the set of fitting service descriptions as well as the content of a previously notified service description	Service-Specification Notification-Callback	Service-Specification available Notification-Callback and notifyService-Discovery implemented on the IoT Service Client side

Output:

Functionality	Impacted	Return value	Exception

Output	Components		
Subscription identifier	-	SubscriptionID	Subscription failed

Interface Definition: Notify Service Discovery

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Subscribe to Discovery of Service Descriptions based on Service Specification" Use Case	IoT Service Resolution	IoT Service Client	notifyService-Discovery: the changed service descriptions fitting the service specifications are provided. The subscription to which the notification pertains is identified by the subscription identifier	SubscriptionID, ServiceDe-scription[]	Notification-Callback available and IoT Service Client reachable using the Notification-Callback

Interface Definition: Unsubscribe Service Discovery

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Unsubscribe to Discovery of Service Descriptions" Use Case	IoT Service Client	IoT Service Resolution	Unsubscribe-Service-Discovery: given the SubscriptionID cancel the respective subscription	SubscriptionID	SubscriptionID available IoT Service Resolution reachable

Interface Definition: Insert Service

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Insert Service Description" Use Case	IoT Service	IoT Service Resolution	insertService Description: insert the given service description into the information base of the IoT Service Resolution	Service Description	Service Description available

Output:

Functionality Output	Impacted Components	Return value	Exception
Service description identifier that uniquely identifies the stored service description	-	ServiceDescriptionID	Service Description could not be inserted

Interface Definition: Update Service

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Update Service Description" Use Case	IoT Service	IoT Service Resolution	updateService Description: update the given service description in the information base of the IoT Service Resolution	Service Description	Service Description available, Service Description to be updated stored in the IoT Service Resolution information base

Interface Definition: Delete Service

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Delete Service Description" Use Case	IoT Service	IoT Service Resolution	deleteService: given the Service-DescriptionID delete the Service Description from the information base of the IoT Service Resolution	Service-DescriptionID	Service-DescriptionID available, Service Description with Service-DescriptionID available in the information base of IoT Service Resolution

C.3 Virtual Entity (VE)

C.3.1 Virtual Entity Resolution functional component

C.3.1.1 Use Cases

In this section, the Virtual Entity Resolution functional component, as identified in the functional view (see Section 4.2.2), is described. It provides a Virtual Entity abstraction level, i.e., Virtual Entities, which are the digital counterparts of Physical Entities, are modelled on this level. Virtual entities and services are linked together using *associations*. Services provide access to information about the corresponding Physical Entities through the resources, to which the services are associated. The Virtual Entity service specification allows the specification of the relation between a Virtual Entity and a service. Notice that the service is part of the association. For example, a room and a temperature service may be related through the relation (e.g., modelled as an attribute) indoorTemperature. The association would contain the virtual identifier of the room, the type of room, the relation indoorTermperature, and the identifier of the service.

The following use cases are depicted in Figure 61.

- *Look up associations for Virtual Entity and Virtual Entity service specification.*
  - This use case is initiated by a user of the system, i.e. a hHuman User or an active digital artefact like a software agent. The user wants to look up associations that associate the identifier of the Virtual Entity with a service providing specific information or allowing executing an actuation affecting the corresponding Physical Entity.
  - The assumption is that the user already knows the identifier of the Virtual Entity.
  - In this use case, the Virtual Entity Resolution looks up the associations corresponding to the identifier and filters them according to the Virtual Entity service specification. As a result, the user receives associations containing identifiers of relevant services.

- If the lookup is successful, the user gets the associations containing the identifiers of the relevant services whose description can then be looked up through the IoT Service Resolution.

- *Subscribe to look-up of Associations based on VE Identifier and VE Service Specification*
  - The use case is initiated by a user of the system, i.e., a hHuman User or an Active Digital Artefact. The user wants to be asynchronously notified about Associations between the Virtual Entity identified by the VE Identifier and services fitting the VE Service Specification. A new notification will be sent whenever a new fitting association becomes available, is removed or there is a change to an Association that was previously sent.
  - The assumption is that the user already knows the VE Identifier of the Virtual Entity.
  - In this use case, the VE Resolution asynchronously notifies the subscribing user about fitting Associations and sends a new notification whenever a new fitting Association has become available, an Association has been removed or a previously sent Association has changed.
  - If the subscription is successful, the user will always get an updated set of Associations fitting the subscription.

- *Unsubscribe to look-up of Associations*
  - The use case is initiated by a user of the system. The user has previously subscribed to receive notifications about Associations between the Virtual Entity identified by its VE Identifier and services fitting the VE Service Specification.
  - The assumption is that the user knows the subscription identifier of the subscription assigned by the VE Resolution.
  - In this use case, the subscription to the VE Resolution identified by the subscription identifier is cancelled.
  - If the unsubscription is successful, the user will no longer receive notifications concerning the Associations between the Virtual Entity identified by its VE Identifier and services fitting the VE Service Specification.

- *Discover associations based on Virtual Entity specification and Virtual Entity service specification*
  - This use case is initiated by a user. The user wants to discover Physical Entities through their corresponding Virtual Entities. These Virtual Entities can provide information about the Physical Entity or trigger actuations on the physical counterpart of the Virtual Entity.
  - The assumption is that the user does not know the virtual identities of these Virtual Entities, but knows what kind of Virtual Entities and what kind of associated services are required.
  - In this use case, Virtual Entity Resolution enables the user to discover relevant associations. Virtual entities are specified through a virtual-entity specification, and the requirements for the associated service are specified in the virtual-entity-service specification. As a result, the user then receives fitting associations.

- If the discovery is successful, the use gets the virtual identities of fitting Virtual Entities together with the identifiers of required services, whose description can then be looked up through the IoT Service Resolution.

- *Subscribe to discovery of Associations based on VE Specification and VE Service Specification*
  - The use case is initiated by a user of the system, i.e., a hHuman User or an Active Digital Artefact. The user wants to be asynchronously notified about Associations between the Virtual Entities fitting the VE Specification and services fitting the VE Service Specification. A new notification will be sent whenever a new fitting association becomes available, is removed or there is a change to an Association that was previously sent.
  - The assumption is that the user does not know the virtual identities of these Virtual Entities, but knows what kind of Virtual Entities and what kind of associated services are required.
  - In this use case, the VE Resolution asynchronously notifies the subscribing user about fitting Associations and sends a new notification whenever a new fitting Association has become available, an Association has been removed or a previously sent Association has changed.
  - If the subscription is successful, the user will always get an updated set of Associations fitting the subscription.

- *Unsubscribe to discovery of Associations*
  - The use case is initiated by a user of the system. The user has previously subscribed to receive notifications about Associations between the Virtual Entities fitting the VE Specification and services fitting the VE Service Specification.
  - The assumption is that the user knows the subscription identifier of the subscription assigned by the VE Resolution.
  - In this use case, the subscription to the VE Resolution identified by the subscription identifier is cancelled.
  - If the unsubscription is successful, the user will no longer receive notifications concerning the Associations between the Virtual Entities fitting the VE Specification and services fitting the VE Service Specification.

- *Manage Virtual Entity/service associations (insert, update, delete)*
  - The use case is initiated by a service or the Virtual Entity & IoT-service Monitoring.
  - The assumption is that an association between a virtual identity and a service needs to be inserted, updated, or deleted.
  - The use case is about the management of associations in the Virtual Entity Resolution.
    - A service or the Virtual Entity & IoT Service Monitoring unit inserts a new association, so that it can be looked up and discovered.
    - A service or the Virtual Entity & IoT Service Monitoring unit updates an existing association, so that any changes are reflected.

- A service or the Virtual Entity & IoT Service Monitoring unit deletes an existing association, indicating that the formerly associated service does no longer provide the specified functionality.

If the management of associations is successful, the associations that can be looked up or discovered reflect the status as reported by the services or the Virtual Entity & IoT Service Monitoring.

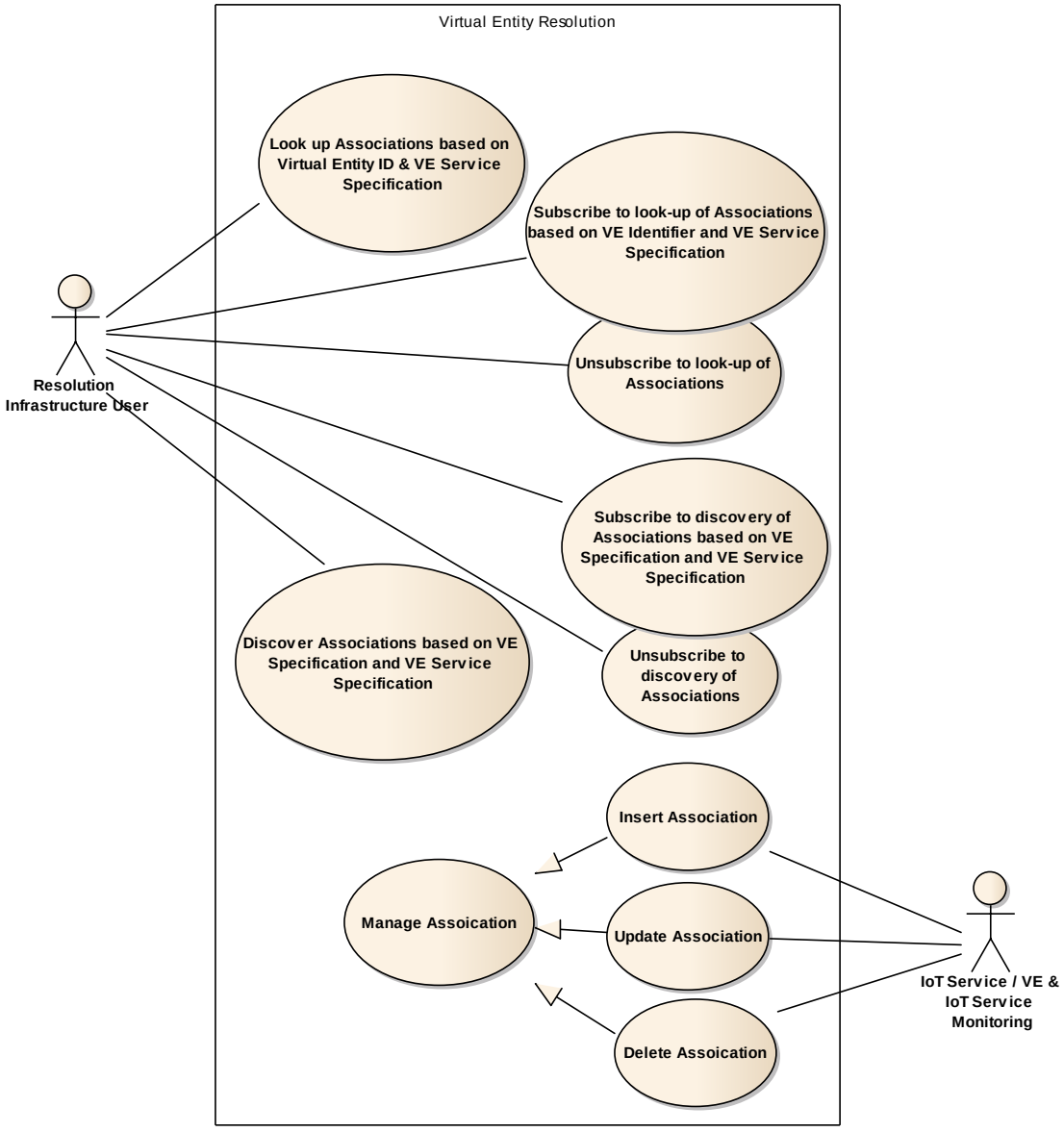

Figure 61: Virtual Entity Resolution.

C.3.1.2 Interaction Diagrams

The Interaction diagram related to the use cases of the Virtual Entity Resolution functional component are depicted below.

Interaction Diagram: Lookup Associations

For the lookup of associations based on the identifier of the Virtual Entity and the specification of the service associated with the Virtual Entity, an IoT Service Client synchronously calls the Virtual Entity Resolution component, using the lookupAssociations operation with the VE-ID and the VEServiceSpecification as parameters. The VEServiceSpecification contains the attribute of the Virtual Entity with which the required service needs to be associated and potentially other information, i.e., if the value of the attribute should be returned by the service or if the service should influence this value as in the case of actuation. An association is the relation between a VE-ID and a Service Identifier and is described by the attribute name and additional information. The Virtual Entity Resolution looks up fitting associations based on the VE-ID and the VEServiceSpecification and provides the resulting array as the return value.

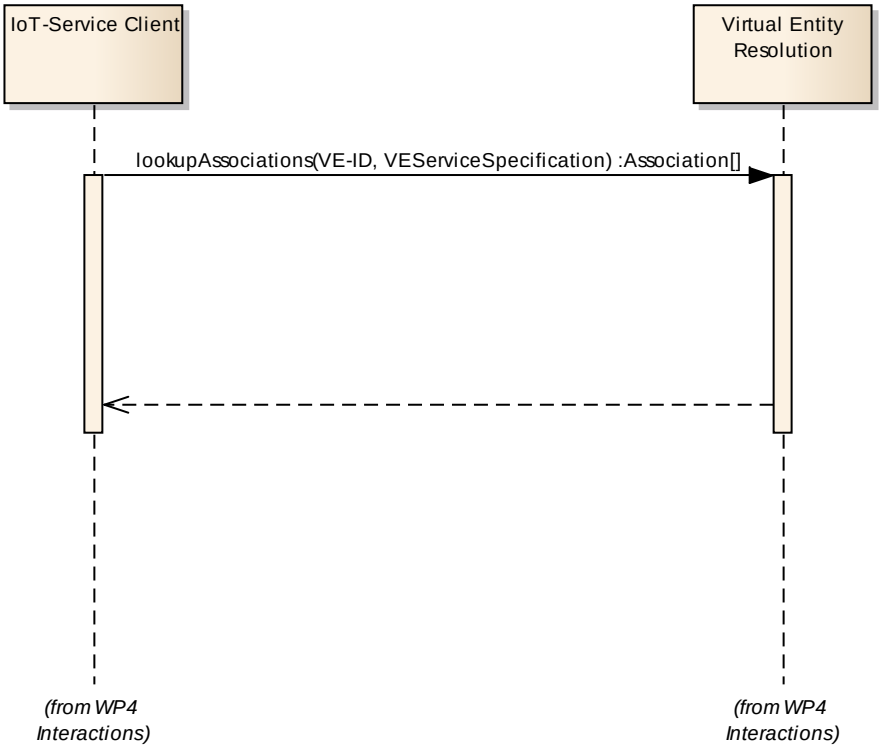

Figure 62: Look up Associations based on VE-ID and VEServiceSpecification.

Interaction Diagram: Subscribe Associations Look-up

For subscribing to asynchronously receive notifications about Associations between a Virtual Entity and services fitting the given VE Service Specification, an IoT Service Client synchronously calls the VE Resolution, using the subscribeAssociationsLookup operation with the Virtual Entity ID, the VE Service Specification and the notification callback, to which notifications are to be sent, as parameters. The notification callback identifies the endpoint on

the IoT Service Client side that implements the notifyAssociationLookup operation. The IoT Service Resolution returns the subscription identifier that can be used to map an incoming notification to the subscription it belongs to.

Subsequently, the VE Resolution will call the notifyAssociationLookup operation of the IoT Service client, providing the Associations and the subscription ID as parameters.

When the IoT Service Client is no longer interested in receiving notifications pertaining to the subscription, it will call the unsubscribeAssociationLookup operation of the VE Resolution using the subscription identifier as parameter. As a result, the VE Resolution will stop sending notifications pertaining to the identified subscription.

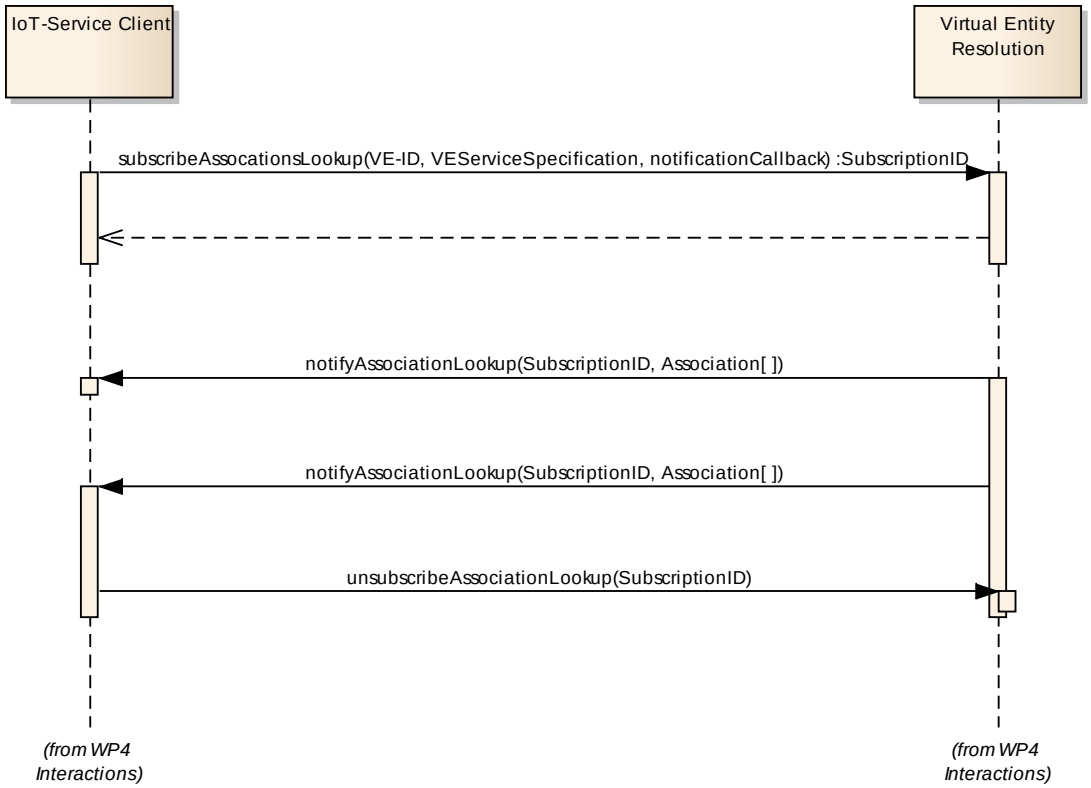

Figure 63 Subscribe Look-up of Associations for VE Identifier and VE Service Specification

Interaction Diagram: Discover Associations

For the discovery of associations based on a specification of the Virtual Entity and the specification of the service associated with the Virtual Entity, an IoT Service Client synchronously calls the Virtual Entity Resolution component, using the discoverAssociations operation with the VESpecification and the VEServiceSpecification as parameters. The VESpecification specifies the Virtual Entities that are of interest. The VEServiceSpecification contains the attribute of the Virtual Entity with which the required service needs to be associated and potentially other information, i.e., if the value of the attribute should be returned by the service or if the service should influence this value as in the case of actuation. An association is the relation between a VE-ID and a Service Identifier and is described by the attribute name and

additional information. The Virtual Entity Resolution discovers fitting associations based on the VESpecification and the VEServiceSpecification and provides the resulting array as the return value. All fitting associations must refer to a Virtual Entity that fits the VESpecification and for this Virtual Entity, the VEServiceSpecification has to fit as well.

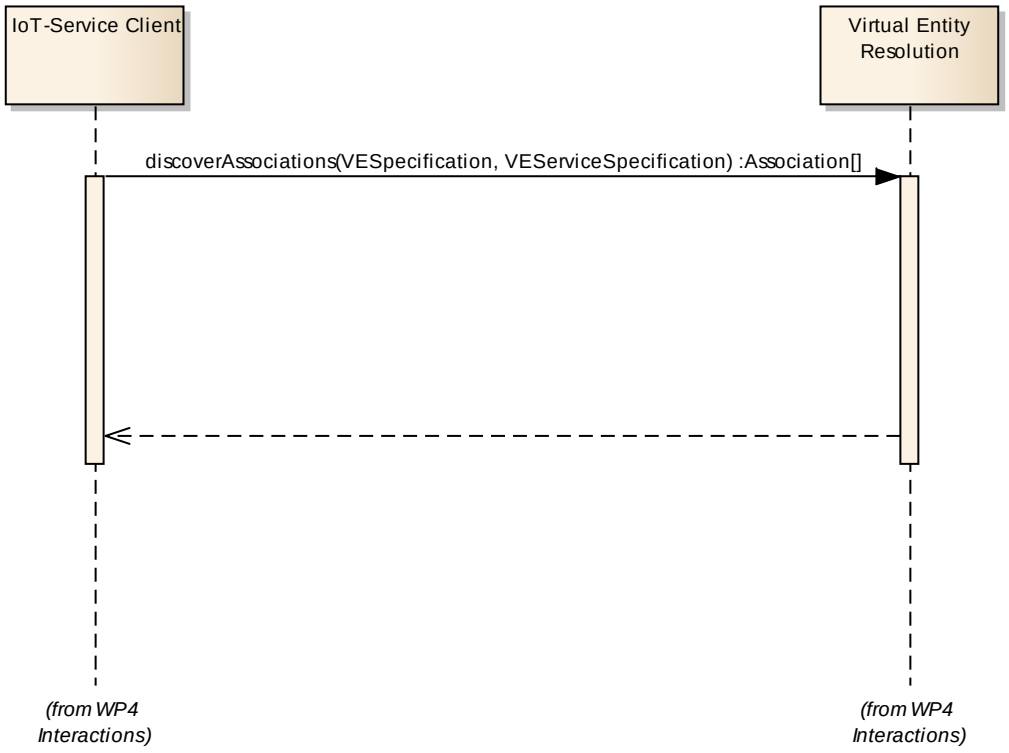

Figure 64: Discover Associations based on VE Specifications and VEServiceSpecifications.

Interaction Diagram: Subscribe Associations Discovery

For subscribing to asynchronously receive notifications about the current set of associations fitting a VE Specification and a VE Service Specification, an IoT Service Client synchronously calls the VE Resolution, using the subscribeAssociationDiscovery operation with the VE Specification specifying the Virtual Entities of interest, the VE Service Specification identifying the services associated to the Virtual Entities that are of interest and the notification callback, to which notifications are to be sent, as parameters. The notification callback identifies the endpoint on the IoT Service Client side that implements the notifyAssociationDiscovery operation. The VE Resolution returns the subscription identifier that can be used to map an incoming notification to the subscription it belongs to.

Subsequently, the VE Resolution will call the notifyAssociationDiscovery operation of the IoT Service client, providing the associations and the subscription ID as parameters. A notification will be sent whenever a previously provided association changes or is deleted. A notification will also be sent if a new association fitting the given VE Specification and VE Service Specification becomes available.

When the IoT Service Client is no longer interested in receiving notifications pertaining to the subscription, it will call the unsubscribeAssociationDiscovery operation of the VE Resolution using the subscription identifier as parameter. As a result, the VE Resolution will stop sending notifications pertaining to the identified subscription.

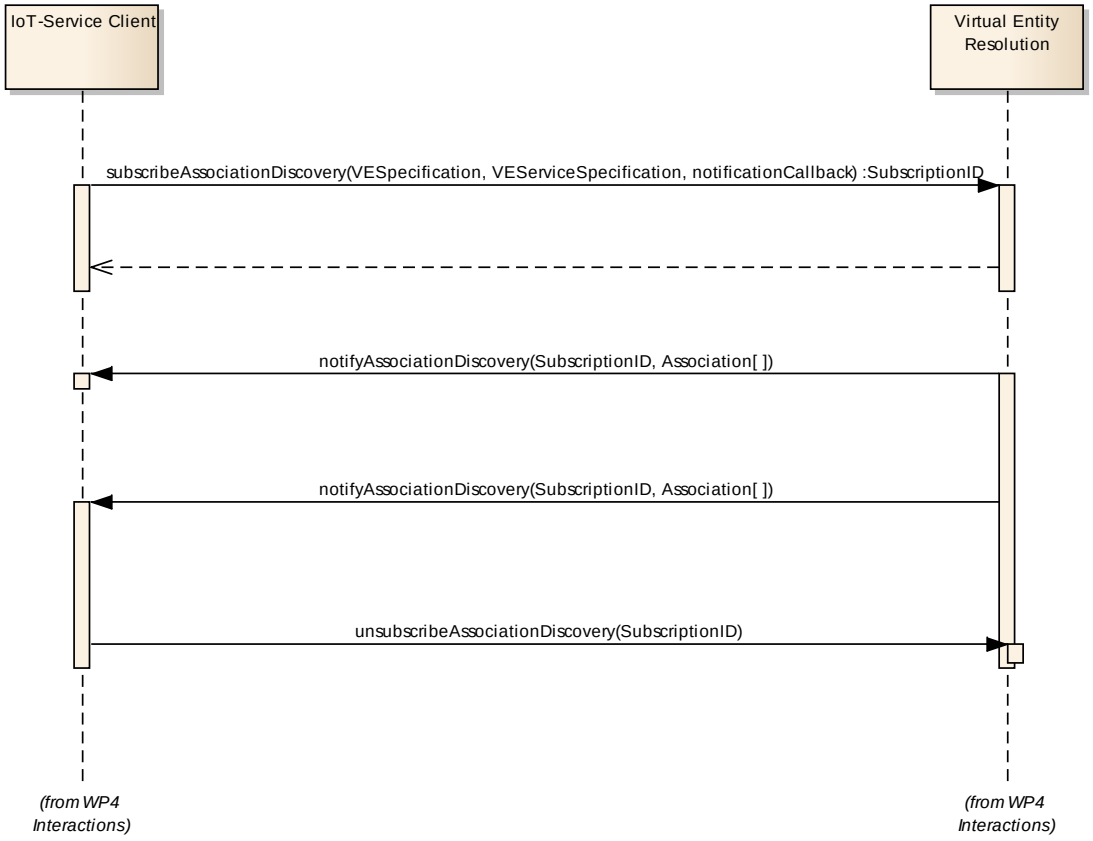

Figure 65 Subscribe Discovery of Associations based on VE Specification and VE Service Specification

Interaction Diagram: Insert Associations

An IoT Service, the VE & IoT Service Monitoring or even another component in the system inserts an Association into the Virtual Entity Resolution component. An association is the relation between a VE-ID and a Service Identifier and is described by the attribute name and additional information. The call to the Virtual Entity Resolution component is synchronous and uses the insertAssociation operation with the Association as parameter. The Virtual Entity Resolution component inserts the Association into its internal information base and returns the AssociationID that uniquely identifies the stored Association. As a result, the updated information required for lookup and discovery can efficiently be found.

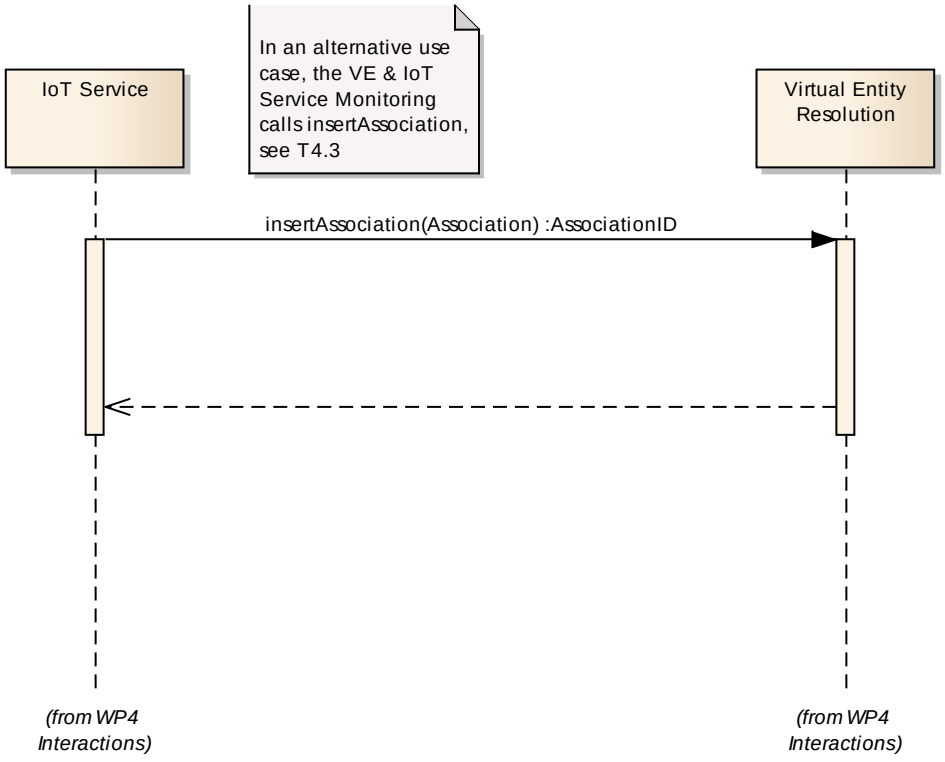

Figure 66: Insert Association.

Interaction Diagram: Update Associations

An IoT Service, the VE & IoT Service Monitoring or even another component in the system updates an Association into the Virtual Entity Resolution component. An association is the relation between a VE-ID and a Service Identifier and is described by the attribute name and additional information. The call to the Virtual Entity Resolution component is asynchronous and uses the updateAssociation operation with the Association as parameter. The Virtual Entity Resolution component updates the Association in its internal information base, so the information required for lookup and discovery can efficiently be found.

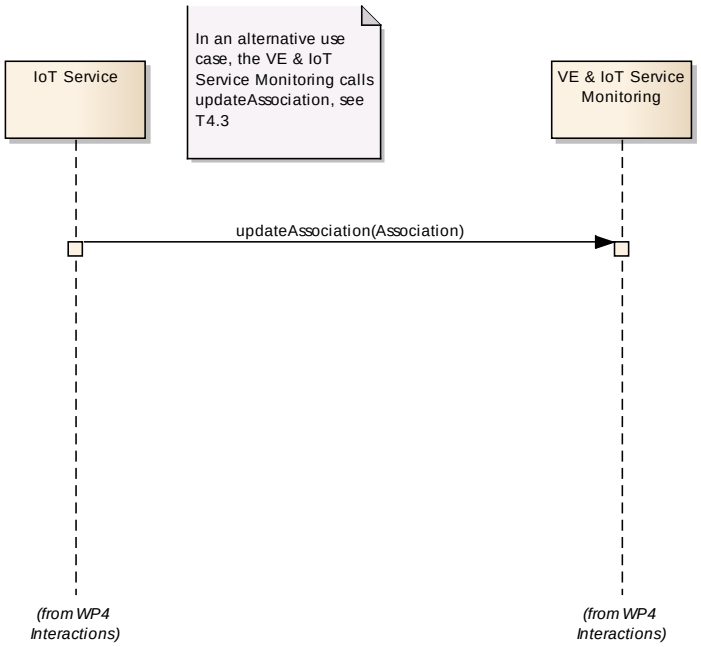

Figure 67: Update Associations.

Interaction Diagram: Delete Associations

An IoT Service, the VE & IoT Service Monitoring or even another component in the system deletes an Association from the Virtual Entity Resolution component. An association is the relation between a VE-ID and a Service Identifier and is described by the attribute name and additional information. The call to the Virtual Entity Resolution component is asynchronous and uses the deleteAssociation operation with the AssociationID as parameter. The Virtual Entity Resolution component deletes the Association identified by the AssociationID from its internal information base.

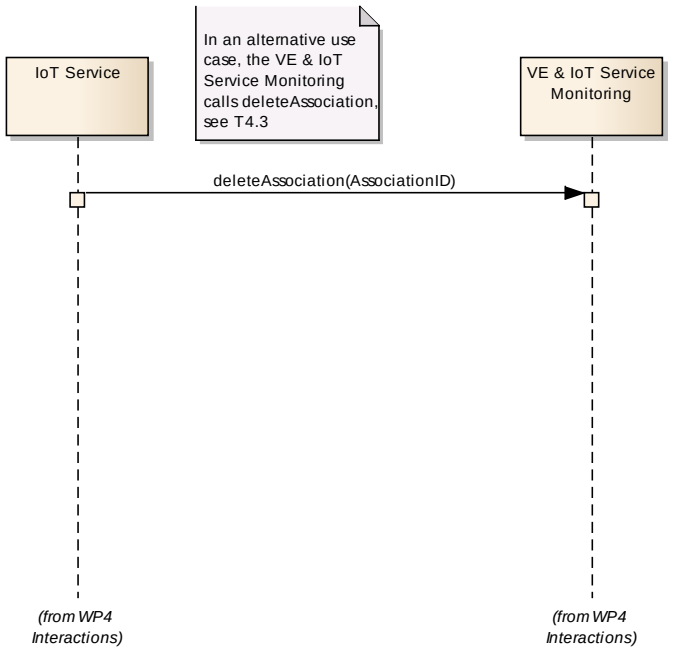

Figure 68: Delete Association.

C.3.1.3 Interface Definitions

In this subsection we present the operations of the Virtual Entity Resolution functional component, i.e., look up association, discover association, insert association, update association and delete association.

Interface Definition: Look Up Association

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Look Up Associations for Virtual Entity & VE Service Specification" Use Case	IoT Service Client	Virtual Entity Resolution	lookupAssociation: given the Virtual Entity ID and the VE Service Specification provide the fitting associations	VE-ID, VEService-Specification	VE-ID, VEService-Specification available

Output:

Functionality Output	Impacted Components	Return value	Exception
Provide the associations that fit the VE-ID	-	Array of Associations	- [no fitting services is a

and the VE-Service-Specification			normal case]

Interface Definition: Subscribe Association Look-Up

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Subscribe to look-up of Associations based on VE Identifier and VE Service Specification ” Use Case	IoT Service Client	Virtual Entity Resolution	subscribeAssociation Lookup: given the VE Entity ID and the VE Service Specification asynchronously notify the IoT Service Client about the fitting associations – as a result of the subscription and on any change of the service description	VE-ID VE Service Specification Notification-Callback	VE-ID available VE Service Specification available Notification-Callback and notify-Association-Lookup implemented on the IoT Service Client side

Output:

Functionality Output	Impacted Components	Return value	Exception
Subscription identifier	-	SubscriptionID	Subscription failed

Interface Definition: Notify Association Look-Up

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Subscribe to look-up of Associations based on VE Identifier and VE Service Specification	Virtual Entity Resolution	IoT Service Client	notifyAssociation-Lookup: an update with those Associations is provided, which, on the one hand, fit the subscription and, on the other hand, have	SubscriptionID, Association []	Notification-Callback available and IoT Service Client reachable using the

” Use Case			changed or have not previously been provided. The subscription to which the notification pertains is identified by the subscription identifier		Notification-Callback

Interface Definition: Unsubscribe Association Look-Up

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Unsubscribe to look-up of Associations ” Use Case	IoT Service Client	Virtual Entity Resolution	Unsubscribe-AssociationLookup: given the SubscriptionID cancel the respective subscription	SubscriptionID	SubscriptionID available VE Resolution reachable

Interface Definition: Discover association

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Discover Associations based on VE Spefication and VE Service Specification ” Use Case	IoT Service Client	Virtual Entity Resolution	discoverAssociation: given the VE Specification and the VE Service Specification provide the fitting associations	VE Specification, VE Service Specification	VE Specification, VE Service Specification available

Output:

Functionality Output	Impacted Components	Return value	Exception
Provide the associations that fit the VE Specification and the VE-Service-	-	Array of ServiceDescription	- [no fitting services is a normal case]

Specification			

Interface Definition: Subscribe Association Discovery

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Subscribe to discovery of Associations based on VE Specification and VE Service Specification ” Use Case	IoT Service Client	Virtual Entity Resolution	subscribeAssociationDiscovery: given the VE Specification and the VE Service Specification asynchronously notify the IoT Service Client about the fitting associations – as a result of the subscription and on any change of the fitting associations	VE Specification VE Service Specification Notification-Callback	VE Specification available VE Service Specification available Notification-Callback and notify-Association-Discovery implemented on the IoT Service Client side

Output:

Functionality Output	Impacted Components	Return value	Exception
Subscription identifier	-	SubscriptionID	Subscription failed

Interface Definition: Notify Association Discovery

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Subscribe to discovery of Associations based on VE Speciofication and VE Service Specification	Virtual Entity Resolution	IoT Service Client	notifyAssociation-Discovery: an update with those Associations is provided, which, on the one hand, fit the subscription and, on the other hand, have changed or have not	SubscriptionID, Association[]	Notification-Callback available and IoT Service Client reachable using the Notification-

” Use Case			previously been provided. The subscription to which the notification pertains is identified by the subscription identifier		Callback

Interface Definition: Unsubscribe Association Discovery

Input:

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Unsubscribe to discovery of Associations ” Use Case	IoT Service Client	Virtual Entity Resolution	Unsubscribe-AssociationLookup: given the SubscriptionID cancel the respective subscription	SubscriptionID	SubscriptionID available VE Resolution reachable

Interface Definition: Insert Association

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Insert Association” Use Case	IoT Service / VE & IoT Service Monitoring	Virtual Entity Resolution	insertAssociation: insert the given association into the information base of the Virtual Entity Resolution	Association	Association available

Output

Functionality Output	Impacted Components	Return value	Exception
Association identifier that uniquely identifies the stored Association	-	AssociationID	Association could not be inserted

Interface Definition: Update Association

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Update Association” Use Case	IoT Service / VE & Iot Service Monitoring	Virtual Entity Resolution	Update-Association: update the given association in the information base of the Virtual Entity Resolution	Association	Association available, Association to be updated stored in the Virtual Entity Resolution information base

Interface Definition: Delete Association

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Delete Association” Use Case	IoT Service / VE & Iot Service Monitoring	Virtual Entity Resolution	Delete-Association given the AssociationID delete the Association from the information base of the Virtual Entity Resolution	AssociationID	AssociationD available, Association in the information base of the Virtual Entity Resolution

C.3.2 Virtual Entity and IoT Service Monitoring functional component

C.3.2.1 Use Cases

This section covers the Virtual Entity and IoT Service Monitoring use cases. The Virtual Entity & IoT Service functional component is responsible for finding and monitoring dynamic associations between Virtual Entities and services. Static associations between Virtual Entities and services are valid all the time, e.g., in cases where the device providing the service is embedded in the Physical Entity which is the physical counterpart of the Virtual Entity. For dynamic entities this is not the case, i.e., they can become invalid. A dynamic association may for example be valid when the device providing the service and the Physical Entity are in close proximity and become invalid if one of them moves away.

Figure 692 covers the following use cases:

- *Assert static Virtual Entity to IoT service association*
  - This use case is internally triggered by the Virtual Entity & IoT Service Monitoring functional component.
  - The assumption is that the functional component was configured with respect to the aspects that need to be monitored in order to assert static associations.
  - The Virtual Entity & IoT Service Monitoring unit asserts a static association between a Virtual Entity and a service.
  - As the result of asserting a new static association, the Insert Association use case of the Virtual Entity Resolution is triggered (see B.3). Due to the static nature of the association, it does not have to be monitored.
- *Discover associations between Virtual Entities and services*
  - The use case is internally triggered by the Virtual Entity & IoT Service Monitoring functional component.
  - The assumption is that the component was configured with respect to aspects that need to be monitored in order to discover dynamic associations (see Annex B.3). Important aspects include the location, proximity, and other context information that is modelled for Physical Entities and devices hosting resources.
  - The Virtual Entity & IoT Service Monitoring discovers new dynamic associations by which Virtual Entities and services are related.
  - As the result of discovering a new dynamic association, the insert association use case of the Virtual Entity Resolution is triggered (see B.3). Also, as the association is dynamic, it needs to be monitored.
- *Monitor existing associations between Virtual Entities and services*
  - The use case is internally triggered by the Virtual Entity & IoT Service Monitoring functional component.
  - The assumption is that it the aspects that were relevant for the discovery of the dynamic association can change so the dynamic association becomes invalid.

- The Virtual Entity & IoT Service Monitoring function monitors the aspects that were relevant for the discovery of the dynamic association (see Annex B.3) to determine whether the association has changed or has become invalid.
- As the result of monitoring an existing dynamic association, the “update association” use case or the “delete association” use case of the virtual-entity resolution can be triggered.

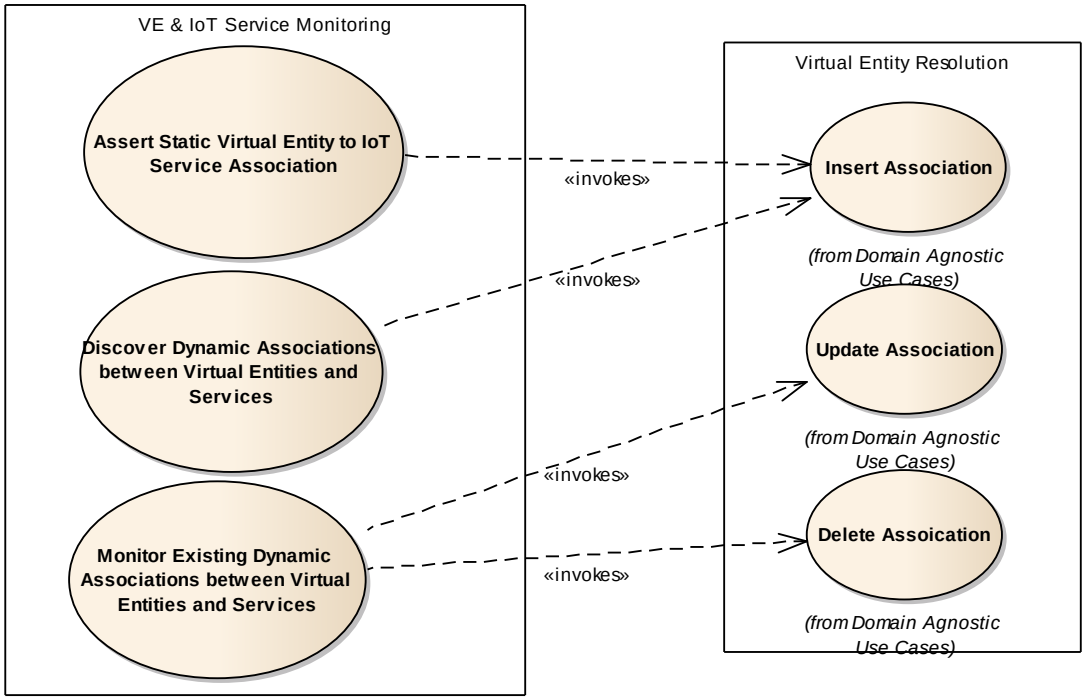

Figure 69: Virtual Entity & IoT Service Monitoring.

C.3.2.2 Interaction Diagrams

The Interaction diagram related to the use cases of the Virtual Entity and IoT Service Monitoring functional component are depicted below.

Interaction Diagram: Assert Static Association

The VE & IoT Service Monitoring component monitors possible static associations between Virtual Entities and IoT Services based on relevant information that may for example include location, ownership or other context parameters. Once a static association has been found, the VE & IoT Service Monitoring asynchronously calls the Virtual Entity Resolution using the insertAssociation operation with the newfound Association as parameter.

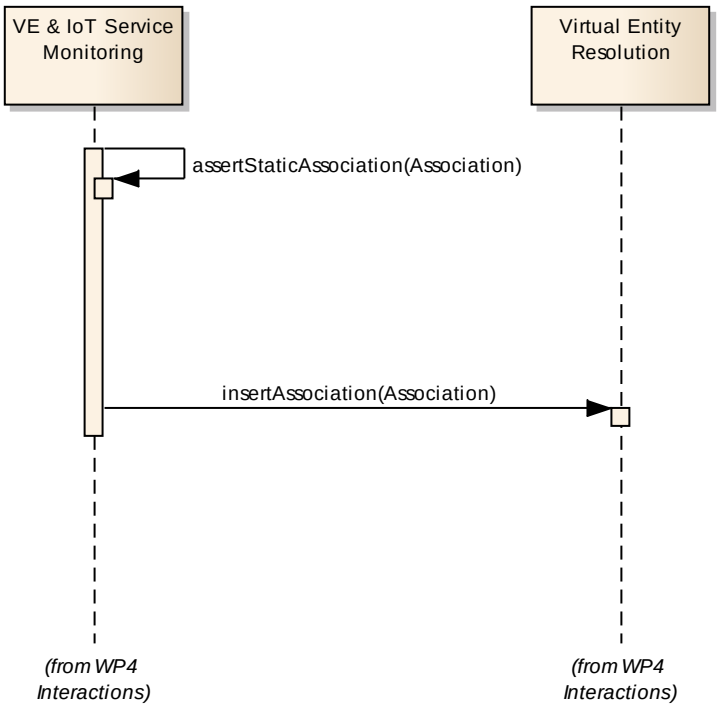

Figure 70: Assert Static VE-IoT Service Association.

Interaction Diagram: Discover Dynamic Associations

The VE & IoT Service Monitoring component monitors possible dynamic associations between Virtual Entities and IoT Services based on relevant information that may for example include location, ownership or other context parameters. Once a dynamic association has been found, the VE & IoT Service Monitoring asynchronously calls the Virtual Entity Resolution using the insertAssociation operation with the newfound Association as parameter. The difference to the case of the Static Association is that the validity of the dynamic associations has to be constantly monitored.

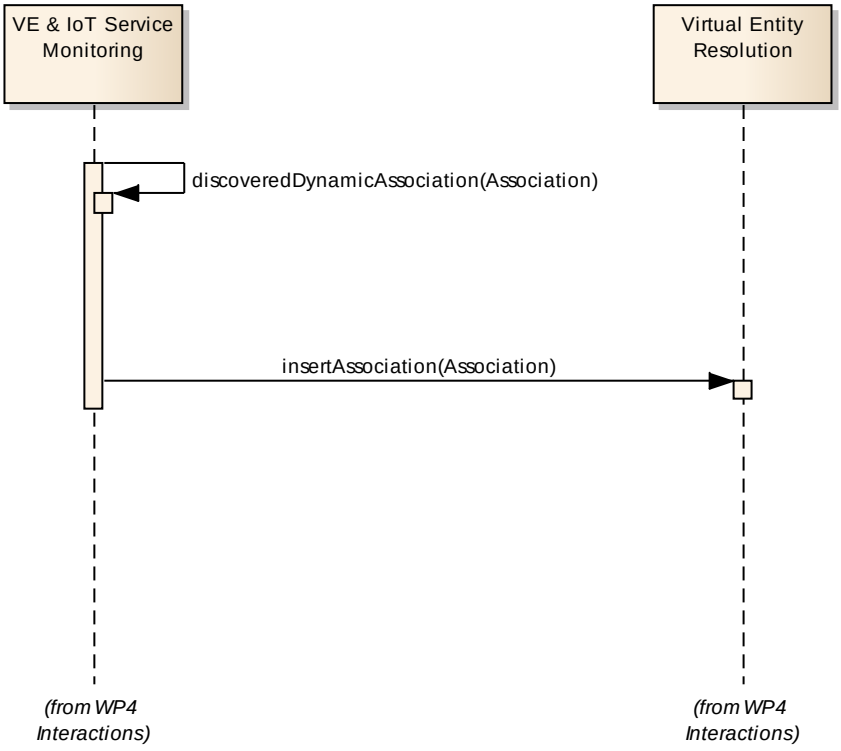

Figure 71: Discover Dynamic Associations between VEs and Services.

Interaction Diagram: Monitor Existing Dynamic Associations - Update

The VE & IoT Service Monitoring component monitors existing dynamic associations between Virtual Entities and IoT Services based on the information that lead to establishing the association. If a change in an existing association has been found, the VE & IoT Service Monitoring asynchronously calls the Virtual Entity Resolution using the updateAssociation operation with the updated Association as parameter.

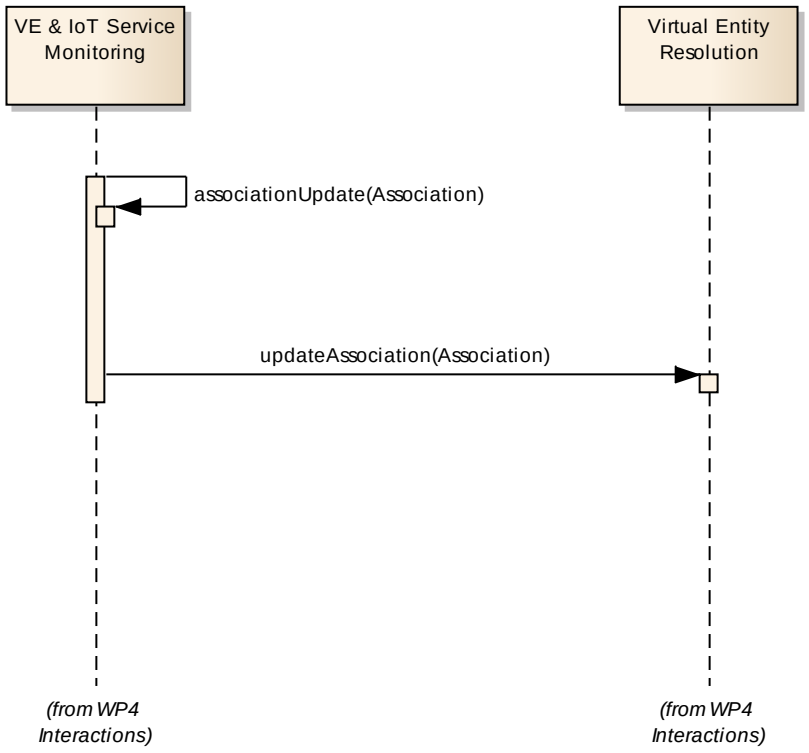

Figure 72: Monitor and Update Existing Dynamic Associations.

Interaction Diagram: Monitor Existing Dynamic Associations - Delete

The VE & IoT Service Monitoring component monitors existing dynamic associations between Virtual Entities and IoT Services based on the information that lead to establishing the association. If a change in the information is found that invalidates the association, the VE & IoT Service Monitoring asynchronously calls the Virtual Entity Resolution using the deleteAssociation operation with the AssociationID as parameter.

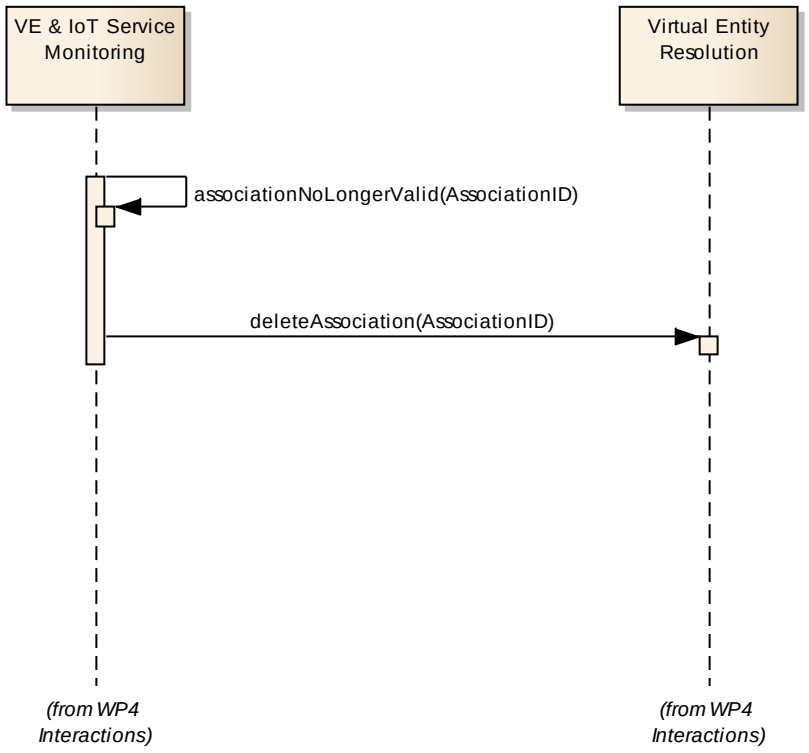

Figure 73: Monitor and Delete Existing Dynamic Associations.

C.3.2.3 Interface Definitions

In this subsection we present the operations of the VE & IoT Service Monitoring functional component, i.e., assert static association, discovered dynamic association, association no longer valid and association update.

Interface Definition: Assert Static Association

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Assert Static Virtual Entity to IoT Service Association " Use Case	VE & IoT Service Monitoring (Monitoring)	VE & IoT Service Monitoring (Adaptation)	assertStaticAssociation: a static association was discovered, update information accordingly	Association	Discovered static association

Interface Definition: Discovered Dynamic Association

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Discover Dynamic Associations between Virtual Entities and Services" Use Case	VE & IoT Service Monitoring (Monitoring)	VE & IoT Service Monitoring (Adaptation)	discoveredDynamic-Association: a dynamic association was discovered, update information accordingly and start monitoring the validity of the dynamic association	Association	Discovered dynamic association

Interface Definition: Association No Longer Valid

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Monitor Existing Dynamic Associations between Virtual Entities and Services” Use Case	VE & IoT Service Monitoring (Monitoring)	VE & IoT Service Monitoring (Adaptation)	associationNoLongerValid: it was discovered that the association with given AssociationID is no longer valid	AssociationID	AssociationID given

Interface Definition: Association Update

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Monitor Existing Dynamic Associations between Virtual Entities and Services” Use Case	VE & IoT Service Monitoring (Monitoring)	VE & IoT Service Monitoring (Adaptation)	associationUpdate: it was discovered that the given Association needs to be updated	Association	Updated Association

C.4 Security

C.4.1 IoT Service Resolution functional component

C.4.1.1 Use Cases

In this section we present two use cases that illustrate the utilisation of security-related functional components.

Both use cases extend the “Discovery of an IoT-Service based on Service Specification” by adding additional steps before and after ensuring security and privacy related aspects. It has to be emphasised that this use case “discovery of an IoT service based on service specification” is just a place holder for any of those use cases identified in the previous Appendix (B.2, B.3).

Use Case 1: Secure Discovery of an IoT Service

This use case illustrates how the discovery of services has to be restricted to those users or applications that are authorised to know about it, including the creation of a new pseudonym (to ensure the privacy of a user). In this use case, it is assumed that the communication between functional components is not limited.

The actor in the use case shown in Figure 747 is a user who utilises a service client to discover an IoT-Service or a high-level service composition or orchestration. An example for such a service is discovery. The following use cases are all depicted in Figure 74.

- Authenticate the user: The user is authenticated and an assertion of his identity is provided[4].
- Discover person-related IoT services for authorised personnel: This use case extends the original discovery IoT service by adding security and privacy protection functionality. The use case includes:
  - Authorise general access to discovery: Apply access restriction to the authenticated user. Such restriction may include further obligations like pseudomisation of the result.
  - Discover service based on service specification.
  - As mentioned above this use case is just a place holder.
  - Filter discovery results: The original result list of the previous use case is limited to those results the authenticated user is allowed to see.
  - Create and deploy new pseudonym: An optional use case, in which the identifier which is discovered will be replaced by a pseudonym and provided to the user.

It is assumed as a pre-condition that the user is known and can be authenticated (e.g. through a password or asymmetric key). The authentication use case only has to be executed once for the validation period of the given assertion. In addition, the policies regarding the discovery of services with respect to privacy are deployed at the respective component. As a post-condition

[4] As an example, an OASIS SAML Authentication Assertion could be provided.

of the secure discovery of an IoT service, the user only receives those services that he is entitled to see due to privacy restrictions.

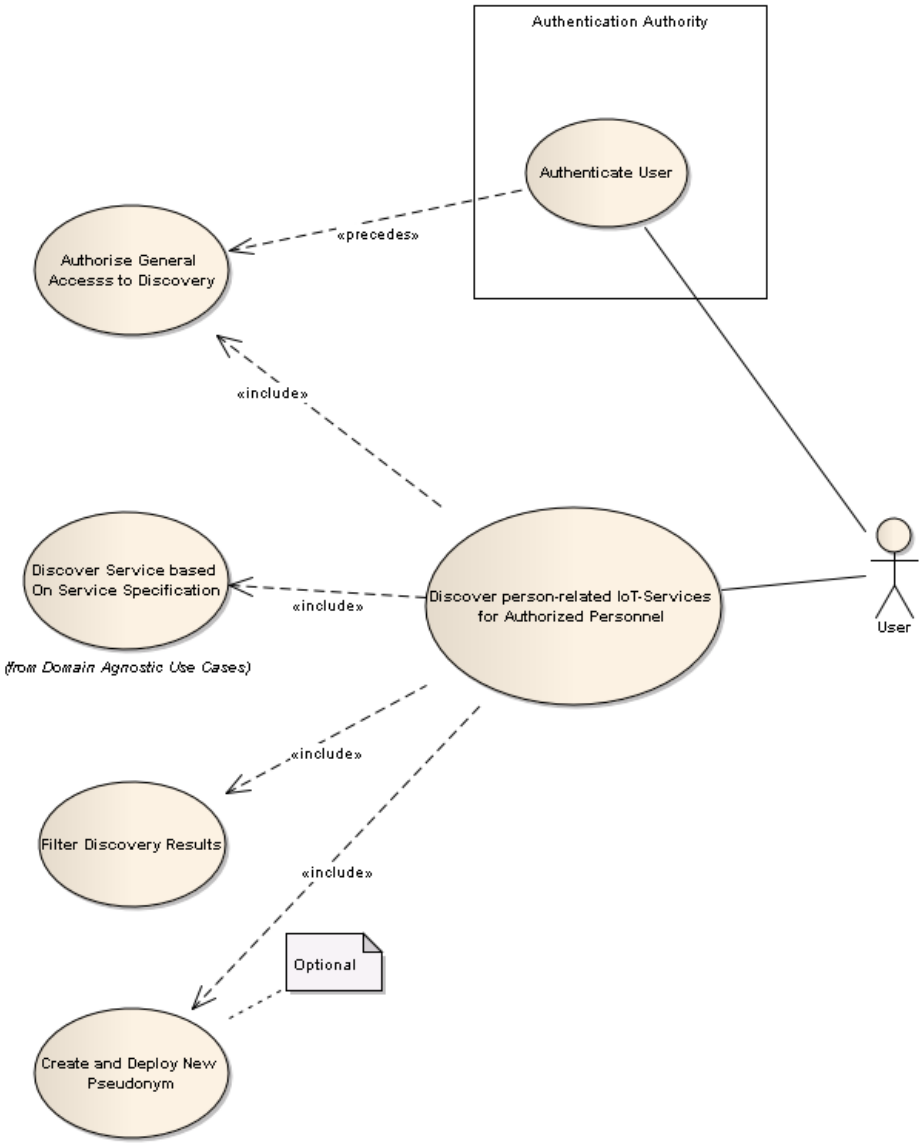

Figure 74: Secure discovery of IoT services.

Use Case 2: Secure Direct Discovery of IoT-Services

The discovery of IoT-Services that may reveal personal information, e.g. those used for health monitoring, needs to be secured also in those cases, in which the discovery is not able to access additional security information on the fly. Thus the related credentials have to be processed prior to the discovery.

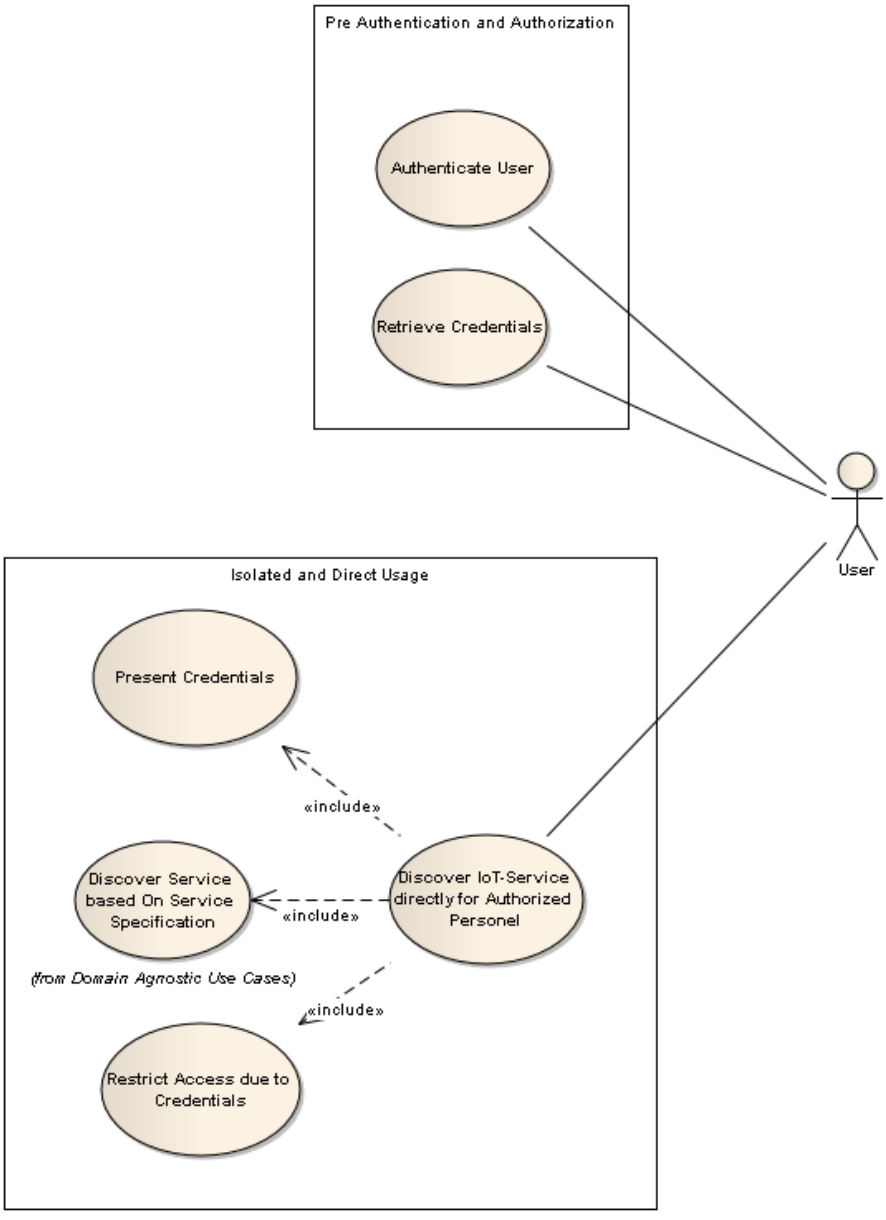

Figure 75: Secure Direct Discovery of IoT Services.

The actor in the uses case shown in Figure 758 is again a user who utilises a service client. In a first phase, during which the related components are available, the following actions take place:

- Authenticate the user: The user is authenticated and an assertion of this identity is provided.
- Retrieve credentials: Based on the identity of the user, a list of credentials is provided, which prove the privileges of the user in a self-contained manner. This proof can also be based on simultaneously deployed information.

During a second phase, the service client may only communicate directly with an isolated discovery component. This includes the actions:

- Discover an IoT service directly for authorised personnel: This use case extend the original Discover IoT-service, by applying access restrictions. It includes:
  - Present credentials: The credentials are verified and the related privileges will be retrieved.
  - Discover service based on service specification
  - As mentioned above, this use case is just a place holder.
  - Restrict access based on credentials: Applies the privileges of the user to the result of the previous use case, especially removes those services that the user is not allowed to see.

It is assumed as a pre-condition that the user is known and that the user can be authenticated (e.g., through password or asymmetric key). Authentication only has to be executed once for the validation period of the given assertion. These assertions allow the user to retrieve the access credentials for further processing during the second phase. In addition, the policies regarding the discovery of services (with respect to privacy) are deployed at the respective component realizing the "*retrieve credential*" use case.

It is assumed that during the second phase, the service client as well as the component realising the *discovery service* is unable to communicate with any of the components realising the use case of the first phase.

As a post-condition of the secure discovery of an IoT Service, the user only receives those services that he is entitled to see according to privacy restrictions.

C.4.1.2 Interaction Diagrams

The Interaction diagram related to the use cases above are depicted below.

Interaction Diagram: Restricted Discovery

Before interacting with the IoT System, the User has to authenticate with the Authentication component of the IoT System. The User synchronously calls the authenticate operation of the Authentication component, providing his/her credentials. The Authentication component verifies the credentials and provides an Assertion that provides the basis for the interaction between the User and the IoT System.

The User utilizes an IoT Service client for interacting with the system. As part of that a discoverService operation may be called by the IoT Service client as described in B.2.1.2.In addition to what is described there, the Assertion is passed to the IoT Service Resolution component as a new parameter. As the first step, the IoT Service Resolution verifies the Assertion calling the verify operation of the Authentication component, providing the Assertion as its parameters. If the Assertion can successfully be verified the operation returns true and the IoT Service Resolution can proceed with the discovery as described in Section B.2.1.2.

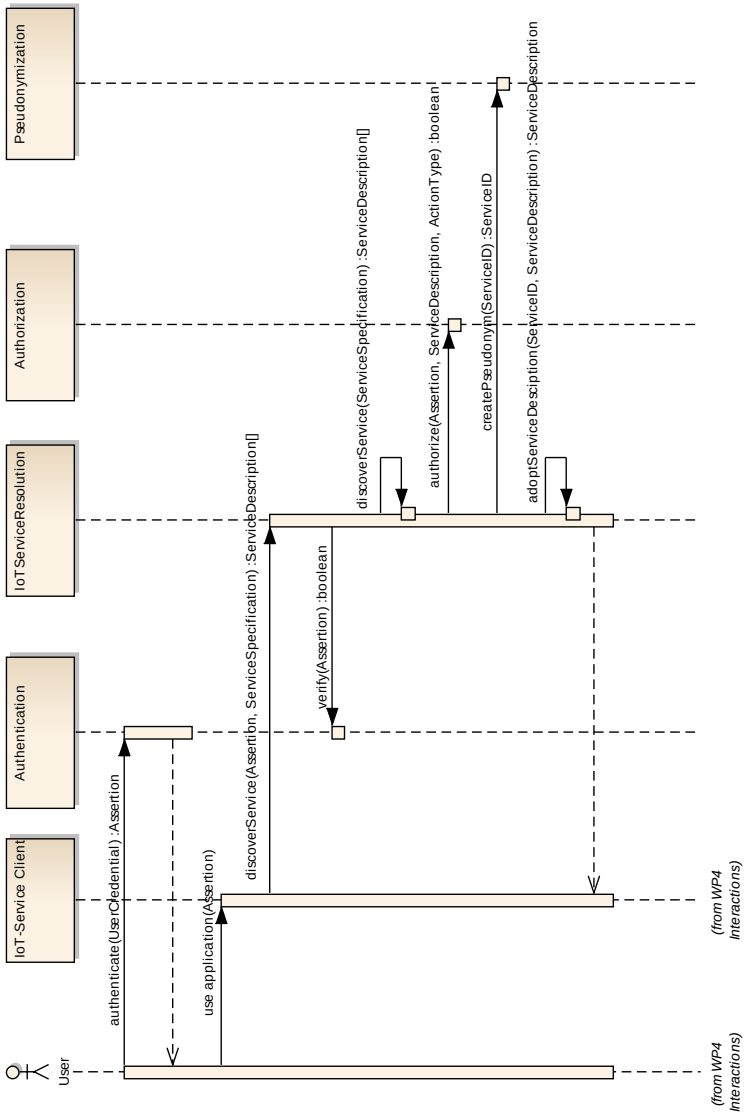

Figure 76: Restricted discovery.

The IoT Service Resolution component then has to check whether the requesting User is allowed to see each Service Description returned by the discovery operation. For this purpose, it calls the authorize operation of the Authorization component, providing the assertion, the Service Description, and the Action Type "discovery". The results can further be pseudonymized by calling the createPseudonym operation of the Pseudonymization component. The result is a new ServiceID. In the next step the IoT Service Resolution can replace the original ServiceID with the pseudonym ServiceID. Finally the array of discovered Service Descriptions is returned to the IoT Service Client as described in the original process in Section B.2.1.2.

Interaction Diagram: Restricted Lookup

In a similar way as the discovery, the service lookup must be controlled in order to protect the privacy of the targeted system (see Figure 770).

Again, the user authenticates to the Authentication components and receives and authentication assertion. In addition to the ServiceSpecification, this assertion is passed to the lookup IoT Service Resolution. The Resolution component verifies the Assertion at the Authentication component and – in case of positive result – the actual look-up process can start. If the ServiceID is a pseudonym (see Section B.2.1.2), then the pseudonym must be resolved first using the Pseudonymization component. Then, it is checked if the user is allowed to lookup that resulting ServiceID. Finally, the ServiceID is used by the actual look-up and the ServiceDescription is returned.

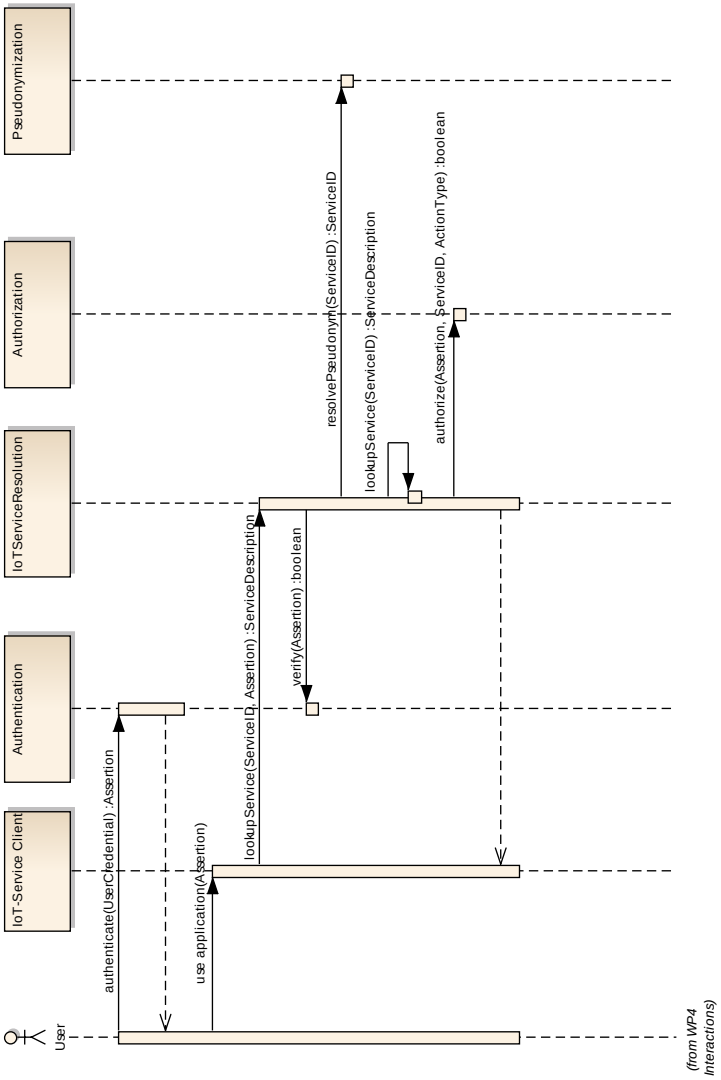

Figure 77: Restricted Lookup.

C.4.1.3 Interface Definitions

In this subsection we present the operations of the security-related functional components that are relevant for IoT Service Resolution, Virtual Entity Resolution and VE & IoT Service Monitoring. The functional components are Authentication, Authorization and Pseudonymization.

Interface Definition: Authentication

authenticate

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
"Restricted Discovery/ Lookup" Use Cases	User	Authentication	authenticate: check the users "identity" by validating his credentials	UserCredential	-

Output

Functionality Output	Impacted Components	Return value	Exception
Authentication Assertion to be presented to services	-	Assertion	Authentication failed

verify

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Restricted Discovery/ Lookup” Use Cases	IoT Service Resolution	Authentication	verify: check the validity of the assertion	Assertion	-

Output

Functionality Output	Impacted Components	Return value	Exception
is the assertion valid?	-	Boolean	-

Interface Definition: Authorization

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Restricted Discovery/ Lookup” Use Cases	IoT Service Resolution	Authorization	authorize: check if the assertion allows to perform the operation on the resource	Assertion Resource ActionType	-

Output

Functionality Output	Impacted Components	Return value	Exception
is the access allowed?	-	Boolean	-

Interface Definition: Pseudonymization

createPseudonym

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite

“Restricted Discovery/ Lookup” Use Cases	IoT Service Resolution	Pseudonymization	createPseudonym: create a Pseudonym for a ServiceID	ServiceID	-

Output

Functionality Output	Impacted Components	Return value	Exception
Pseudonym for the ServiceID	-	ServiceID	-

resolvePseudonym

Input

Illustrative Action	Calling Component	Called Component	Name of the functionality	Parameters	Prerequisite
“Restricted Discovery/ Lookup” Use Cases	IoT Service Resolution	Pseudonymization	resolvePseudonym: get the “real” ServiceID for the pseudonymized ServiceID	ServiceID	Pseudonym was created before

Output

Functionality Output	Impacted Components	Return value	Exception
ServiceID for the pseudonymized ServiceID	-	ServiceID	ServiceID not found