Internet-of-Things Architecture

IoT-A

Project Deliverable D1.2 – Initial Architectural Reference Model for IoT

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

Doc. Ref.:	D1.2
Responsible Beneficiary :	Siemens AG
Editor(s):	Joachim W. Walewski (Siemens)
List of contributors:	Martin Bauer (NEC), Nicola Bui (CFR), Pierpaolo Giacomin (HEU), Nils Gruschka (NEC), Stephan Haller (SAP), Edward Ho (HSG), Ralf Kernchen (UniS), Mario Lischka (NEC) ; Jourik De Loof (ALU BE), Carsten Magerkurth (SAP), Stefan Meissner (UniS), Sonja Meyer (SAP), Andreas Nettsträter (FHG IML), Francisco Oteiza Lacalle (TID), Alexander Salinas Segura (UniW), Alexandru Serbanati (CATTID), Martin Strohbach (NEC), Vincent Toubiana (ALBLF), Joachim W. Walewski (Siemens)
Reviewers:	Stephan Haller (SAP)
Contractual Delivery Date:	2011-05-31
Actual Delivery Date:	2011-06-16
Status:	Draft

Version and date	*Changes*	*Reviewers / Editors*
V0, 2011-04-04	Table of Contents (up to second layer)	Joachim W. Walewski
V1, 2011-05-27	All Sections populated. Proof-reading by peers.	All
V2, 2011-05-28	Editorial proof reading	Joachim W. Walewski
V3, 2011-06-01	Proof reading of entire document.	Stephan Haller
V4, 2011-06-10	Feedback to all proof-reading comments provided and all Sections improved as per the provided criticism.	All
V5, 2011-06-10	Second proof reading of entire document.	Stephan Haller
V6, 2011-06-16	Feedback to all proof-reading comments provided and all Sections improved as per the provided criticism.	All

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

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

Executive summary

One of the main outputs from the IoT-A project is an architectural reference model for the Internet of Things. This project deliverable provides an initial version of this model, viz. version 0.9.
In this report we discuss the motivation behind this effort, the aspirations of the wider community, and what procedure we followed when drafting the architectural reference model presented here. We also explain how our IoT Reference Model, the IoT Reference Architecture, and concrete IoT architectures relate to each other.
The overview Section is followed by a detailed discussion of the IoT Reference Model. This discussion covers the main building blocks of the IoT Reference Model, i.e., the domain model, the information model, and the communication model.
In the next Section we provide a first look at the IoT Reference Architecture, focusing on the functional view and also security and privacy aspects.
In the appendix, the reader can find the requirements that guided the inference of the reference architecture, and we also provide an overview of use cases that will be used in our future work for defining interfaces between functionality groups of the reference architecture and for validating the reference architecture itself.

Table of Content

List of abbreviations

API	Application-programming interface
AuthN	Authentication
AuthS	Authorisation
CA	Certification authority
CCTV	Closed-circuit television
CD	Constrained device
CDSecFeat	Security feature for constrained device
CIM	Common information model
CP	Control point
D*n.m*	IoT-A deliverable *n.m*
DNS	Domain-name system
DP	Data processor
DS	Data sink
EMR	Electronic medical record
EPC	Electronic Product Code
EPCIS	Electronic Product Code Information Services
ERP	Enterprise resource planning
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
OASIS	Organization for the Advancement of Structured Information Standards
QR	Quick response
M2M	Machine to machine
ONS	Object-naming service
OPEX	Operational expenditure
OS	Operation system
OSI	Open -ystem interconnection
QoS	Quality of service
PE	Physical entity
PN	Pseudonymisation
R&D	Research and development
RF	Radio frequency
RFID	Radio-frequency identification
RMD	Remote monitoring device
SID	Shared information & data model
SoTA	State of the art
SW*n*	Stakeholder workshop *n*
TRA	Trust and reputation
UCD	Unconstrained device
UDDI	Universal Description, Discovery and Integration
UNI.*n*	Unified requirement, number *n*
USDL	Unified service-description language
VE	Virtual entity

WP	Work package
WSN	Wireless sensor network
ZigBee	Specification for a suite of high-level communication protocols

1. Vision of the architectural reference model

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. While this situation persists in present times, it is commonly believed that the advent of power-efficient protocols for low-cost communication will even enhance this trend by enabling a plethora of cost-effective, rapidly evolving connected devices. But there is a high risk that application developments provide limited interoperability, if there is no common standardisation and understanding of the IoT domain. In some fields such as manufacturing and logistics, communication and tagging solutions are well established (they provide a clear business benefit in terms of asset tracking and supply-chain management). In other fields, such as home automation, only recently new families of products have shown that economic and social benefits can be achieved. Yet, these domain-specific solutions have mostly been based on protocols developed with a single application or scenario in mind. Many of these solutions are not interoperable at the communication layer, and often not even at the service layer. They thus represent closed vertical silos that co-exist in parallel, but do not promote integration. Although many technology and system providers label their solutions as **Inter**net-of-Things technologies, in reality they form disjoint **Intra**-nets of Things. Furthermore, 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, many of these solutions are based on inappropriate models of governance and fundamentally neglect privacy and security in their design.

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 and that defines security, privacy, performance, and similar needs. 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. This has the major advantage of ensuring backward-compatibility of the model and also the adoption of established, working solutions to 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 the architecture methodology defined within Section 1.

Figure 1 shows an overview of the process we used for defining the different parts that make the IoT-A architectural reference model. Notice that definitions of terms such as reference architecture, etc. can be found in an external glossary [IoT-A, 2011]. Starting with existing architectures and solutions, generic baseline requirements can be extracted and used as an input to the design. The IoT-A architectural reference model 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 architectural reference architecture can be validated against selected business scenarios. A stakeholder analysis contributes to the 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 1.4. According to common usage, this part constitutes a subset of the vision [Open Group, 2009], which is why it is folded into Section 1. This dependency is also reflected in Figure 1.

- 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 reference model includes a general discourse on the IoT domain, a domain model as a top-level description, an information model explaining how IoT knowledge is going to be modelled, and a communication model in order to understand interaction schemes for smart objects. The definition of the IoT reference model is conforming to the OASIS reference model definition [MacKenzie, 2006]. A more detailed description of the IoT reference model is provided in Section 2.

- The **IoT reference architecture** is the reference for building compliant IoT architectures (see Section 3). 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 [ANSI, 2000; Woods, 2005]. Definitions of these terms are also provided in Section 1.2.2. The creation of the IoT Reference Architecture focuses on abstract sets of mechanisms rather than concrete application architectures.

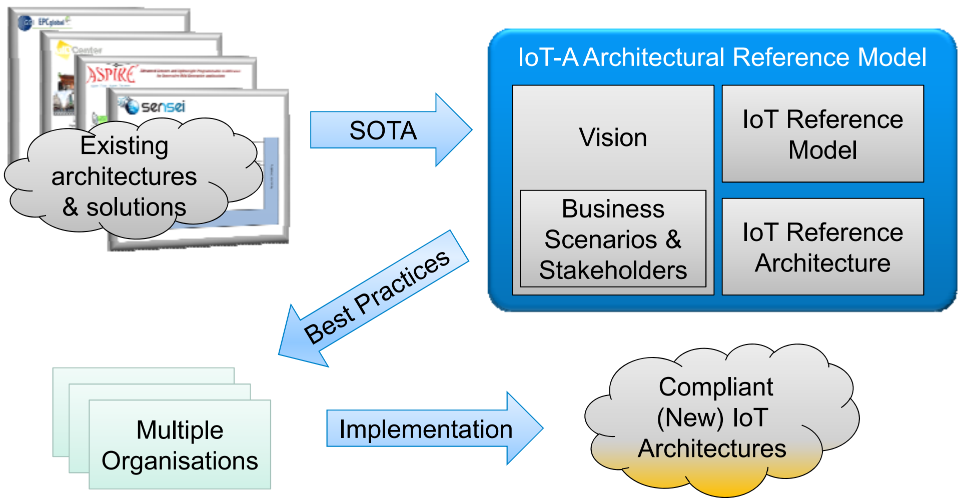
Existing architectures & solutions
SOTA
IoT-A Architectural Reference Model
Vision
IoT Reference Model
Business Scenarios & Stakeholders
IoT Reference Architecture
Best Practices
Multiple Organisations
Implementation
Compliant (New) IoT Architectures

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 architectural reference model provides best practices to the organisations so that they can create compliant IoT architectures in different application domains. Where application domains are overlapping, the compliance to the reference architecture ensures the interoperability of solutions and allows the formation of new synergies across those domains.

Section 1 started with a general overview of the motivation and structure of the IoT-A architectural reference model. In Section 1.1, the different uses for this model are introduced. Section 1.2 describes the dependencies on other IoT-A project deliverables. In Section 1.3, the methodology for building a reference architecture is explained. Section 1.4 introduces the relevant business scenarios and stakeholders. And finally in Section 1.5, the results reported in this deliverable are summarised.

1.1 Uses of an architectural reference model

This Section describes the beneficial uses of the IoT-A architectural reference model, with focus on the contained IoT Reference Architecture. These uses are described in the following list.

Generation of architectures
One benefit is the use of the reference architecture (together with best practices that are use-case specific) for the generations of compliant architectures for specific systems. This is done by enabling tool support. The benefit of such a generator 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 IoT systems that are built according to such derivative architectures [Shames, 2003; Usländer, 2007].

Identifying differences
When using the aforementioned system-generation tools based on the architectural reference model, any differences in the derived architectures can be attributed to the particularities of the pertinent use case [Shames, 2003]. When applying the architectural reference model predictions of system complexity, system cost, etc. are available for the general 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 uncertainty might become possible. The overall implementation effort is most certainly less than developing the use case without the help of an architectural reference model.

Benchmarking
Another important use is benchmarking. For example, NASA used a reference architecture of its new exploration vehicle to better benchmark tenders it was going to receive during a public bidding process [Tamblyn, 2007]. In other words, 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 so doing it also provides a high level of transparency to the benchmarking process.

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 intricacies of IoT and related technical aspects.
Third, the architectural reference model can assist project leaders when planning the work ahead 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 architectural reference model aids in identifying independent building blocks for IoT systems. This constitutes very valuable information when dealing with questions like system modularity, third-vendor options, re-use of already developed components, etc.

1.2 Project-internal input

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

- 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 [Pastor, 2010]. This requirements list was analysed and views and perspectives were assigned to all requirements. A list of these requirements and the assigned views/perspectives can be found in Annex A.1.
- D1.1, which contains a summary of the state of the art of IoT-related architectures, service interfaces, communication layers, resolution infrastructures, and hardware [Bui, 2011]. 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 (see Annex A.2)

Furthermore, as already mentioned, IoT-A provides a webpage on which all the IoT terminology that is used in this deliverable (and will be used in forthcoming IoT-A deliverables) is listed [IoT-A, 2011].

1.3 Architecture methodology

The IoT-A project follows a spiral design and development model. Describing architectures is a modelling exercise. Through this exercise, a better understanding of the IoT domain is reached. Also, this exercise allows for iterations in the architecture building process, so that the stability of models, as part of the architecture, is increased. Each iteration generates updates and additional content for the architecture description, as the understanding of the application IoT domain increases. An architecture methodology is defined to ensure consistency of the architecture description during each iteration. The architectural reference model mainly describes dependencies between models (i.e., the IoT Reference Model guides the definition of the IoT Reference Architecture). Once a change is proposed in one modelling aspect, a clear chain of dependencies can be followed. By so doing, an overall consistency of the IoT-A architectural reference model is achieved.
This Section outlines our architecture methodology.

1.3.1 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 both more abstract than system architectures that have been designed for a particular application. From the literature, we can extrapolate the dependencies of reference architecture, architectures, and actual systems (see depicted in Figure 2). Architectures do help in designing, engineering, building, and testing actual systems. At the same time, understanding systems 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. A reference architecture can provide guidance in form of best practices. 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.

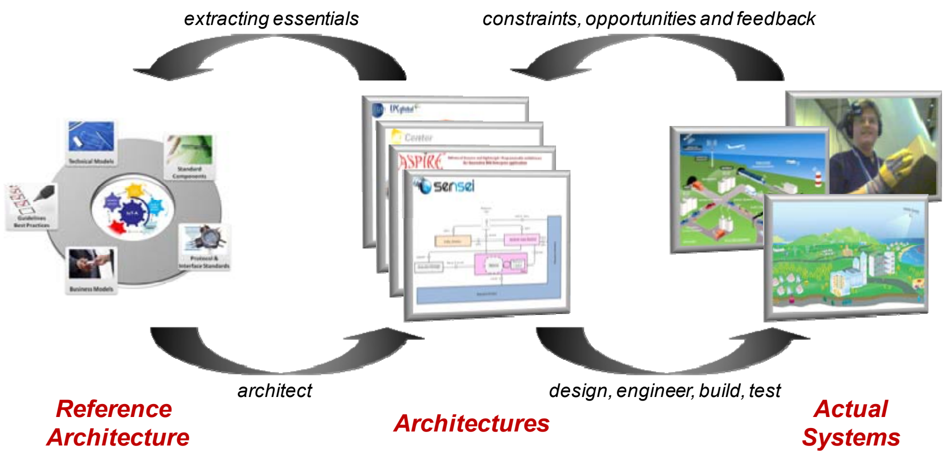

Figure 2: Relationship between a reference architecture, architectures, and actual systems (adapted from Mueller [Mueller, 2008])

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

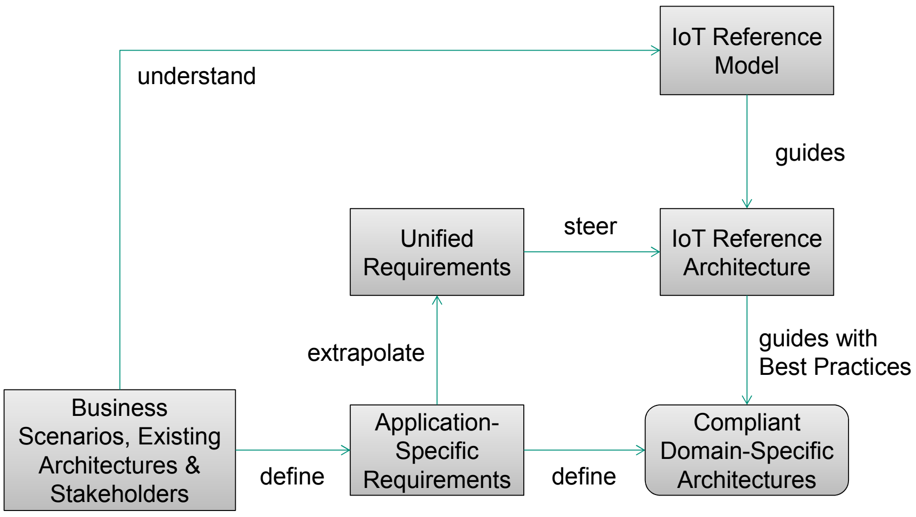
IoT Reference Model
understand
guides
Unified Requirements
steer
IoT Reference Architecture
extrapolate
guides with Best Practices
Business Scenarios, Existing Architectures & Stakeholders
define
Application-Specific Requirements
define
Compliant Domain-Specific Architectures

Figure 3: High-level taxonomy of the IoT reference model and IoT reference architecture dependencies and model influences.

The IoT Reference Model provides guidance for the description of the IoT reference architecture. The IoT reference architecture in turn guides the definition of compliant domain-specific architectures. Essential inputs for the definition of the IoT reference model are stakeholder concerns, business scenarios, and existing architectures. Important here is 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. A more detailed explanation of what guides the IoT-A architecture-modelling process is provided in the next Section.

1.3.2 Structure of the IoT Reference Architecture description - views and perspectives

The IoT reference model by itself (see Section 2) 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 reference architecture (see Section 3). In order to build such a reference architecture, we not only need the domain model, as provided in the reference model, but also technical requirements that can be used for inferring particularities of the architecture. For this step a comprehensive list of (technical) requirements is needed.
First, it is explained how the requirements for the IoT-A architectural reference model are inferred. Second, it is described how the IoT Reference Architecture is derived from such a list of requirements.
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 when drafting the reference model.
2. The state of the art concerning thing-centric communication and Internet technologies was considered, and a list of internal requirements is inferred. The state of the art was collected in Deliverable D1.1 [Bui, 2011]. The unabridged requirements derived from this (plus some additional requirements based on the experience of the project partners) will be published in a future version of the architectural reference model. In the current version of this deliverable, an initial list of such requirements is listed in Annex A.2.
3. For the third activity, a group of IoT stakeholders was established and queried for their visionary IoT 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 additional requirements (see Annex A.1).

Stakeholder aspirations can be diverse because of different application domains and differences in related business models. Nevertheless, there are some common themes when it comes to stakeholder aspirations.

- Many stakeholders see IoT as a means of improving their current business, for instance logistics. 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 were, as expected, identified as a core topic. Privacy also equates to the overall acceptance of IoT. If individuals and other users cannot experience a sufficient level of privacy when utilising IoT technology, this might 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.

Requirements steer the design of the IoT reference architecture and support its respective description (see Section 3). For the definition of the IoT reference architecture, we are following standardised architecture structuring mechanisms as defined in the literature [Open Group, 2009; Rozanski, 2005]. The three main terms in this respect are viewpoints, views and perspectives, which are introduced in the following

> "A *view* is 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". [Open Group, 2009]
>
> "A *viewpoint* is 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". [Open Group, 2009]
>
> "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 *views*." [Rozanski, 2005]

Following these definition the requirements analysis identified several views and perspectives that serve as the basis for the IoT Reference Architecture structuring approach. The views and their definition are provided in the following list.

- **Functional** - Describes the system's runtime functional elements and their responsibilities, interfaces, and primary interactions
- **Information** - Describes the way that the architecture stores, manipulates, manages, and distributes data and information.
- **Deployment** - Describes the environment into which the system will be deployed, including the dependencies the system has on its runtime environment
- **Operational** - Describes how the system will be operated, administered, and supported when it is running in its production environment

Quality aspects of the IoT reference architecture are addressed through definitions of perspectives. The perspectives and their definition are provided in the following list:

- **Security and privacy** – Provides and analysis of the security threads in the functionality groups of the architecture and gives an explanation how security and privacy concerns should be addressed.
- **Performance and scalability** – Provides the ability of the system to handle a large number of devices, services and processes in an efficient way. Further more, the fluctuation of requests towards those has to be handled in a scalable way.
- **Availability and resilience** – 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
- **Evolution and interoperability** – The ability of the system to be flexible in the face of the inevitable change that all systems experience after deployment; the ability of two or more systems or components to exchange and use information.

This deliverable provides the initial architectural reference model for the IoT. For the current version not all views and perspectives are addressed, but restricted to the functional view and the security and privacy perspective. These will be introduced in Section 3 as part of the IoT reference architecture.

Usually, stakeholder aspirations are not made in the language of perspectives and views. Therefore, each stakeholder aspiration was thoroughly analysed, and suitable views and/or perspectives were identified. 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.” [Pastor, 2010] 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 [Pastor, 2010]), we broke this concern down into two distinct functional requirements.

1. “The system shall provide functionality that allows the specification of business processes that autonomously monitor information related to Physical entities and controls the respective aspects of the Physical entity.” (UNI.31)
2. “The system shall provide means for IoT-entities to react autonomously on context data (e.g. by using a rule language).” (UNI.32)

The above example was provided in order to illustrate our requirement process, and also how we enable the traceability of requirements back to stakeholder aspirations. The unabridged list of requirements derived from stakeholder aspirations are provided in Annex A.1. The functional view is a recurring item in the list of unified requirements. This view is represented as a block-diagram of the architecture, which in itself constitutes a central result of the architecture project and an indispensable input for the development of a compliant IoT system. The IoT-A functional view is addressed in detail in Section 3.1.

1.3.3 How we meet requirements

So far we have outlined how requirements are resolved. Next, the question arises how we will meet those requirements and thus the stakeholder aspirations. The answer to this question has several parts.

First, as already pointed out, we have a requirements-engineering process in place which provides a step-by-step traceability from aspirations to requirements. In other words, we ensure that all aspirations are accounted for.
Second, the grouping of requirements by views and perspectives ensures that requirements of the same type are put into the same group, so that all the architecture and reference-model constraints for meeting all stakeholder aspirations are put in place in a coherent manner.
Third, our project-internal process foresees two reviews of the architectural reference model. During these reviews the architectural reference model itself will be assessed, but also whether it meets the stakeholder aspirations and requirements derived from the state of the art.

1.4 Business scenarios

In the future, all facets of the quotidian world will be influenced by IoT systems, in fact the whole world will be a global IoT scenario in which we will live and interact. This global scenario will be loosely compartmentalised, and its boundaries will be crossed by interaction processes and information flows. An Example for such a cross over is that physical entities could exist in many different social environments (work, family, personal, leisure...). Thus, the main difficulties arise when it comes to determining clear boundaries among these possible scenarios.

In any case, 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 kindly been provided by IoT-I [IoT-I, 2011].

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 decreases thus 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 process 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.

Field of application	Impacts
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.
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, encompasing economy, people, governance, mobility, environment and living [Giffinger, 2007]. 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 [Nicholson, 2010]): • 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 reduction in the number of employees needed. Workers will be replaced by bar-code scanners, readers, sensors and actuators, and in the end by complex robots, as much 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 [Field, 2008]. 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
eHealth	Controlling and preventing is one of the main goals of future health care. Already today, people can have the possibility of being tracked and monitored by specialists even if both are not at the same place. 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. Providing information about the state of a patient makes the whole process more efficient, and also makes people much more satisfied. 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).
Environment	From the aforementioned application we infer that environment has many overlaps with other scenarios, such as smart home and smart city. The 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 turn off devices temporarily which are not in use. Finally, combining the analysis of supply and demand, energy enterprises will able to supply a more efficient demand shaping. They will not just give incentives to consumers, but actually turning 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.

Table 1: IoT business scenarios [IoT-I, 2011].

Societal Impact

Obviously, the introduction of IoT systems to our quotidian lives implies new revolutionary social benefits, but also it brings challenging handicaps. A positive impacts we have identified are

- A substantial improvement in health systems - we will live longer and healthy at home;
- daily activity (work and leisure) will be more rich, safety and comfortable and energy savings that result in a better environment.

These are not all, but the most important benefits.

On the other hand, there are also challengers that come with IoT. One of the most important challenges for society is to ensure that all citizens enjoy the benefits of IoT. Studies have predicted the loss of jobs for blue-collar workers due to self-organising smart objects and the problems of interaction and adoption of technology into society. An example for that former is that companies will be able to track all their products by means of RFID tags in a global supply chain. As a consequence, companies will be able to reduce costs and improve their productivity. But this will also come with a potential reduction in the number of employees needed. Therefore, private companies and administrations must be aware about the risk of a more unequal society and social exclusion; current tendencies lead to a divided society, white collar employees on one side and on the other side long-term unemployed.

As usual in ICT, the privacy provided by their platforms, or the lack thereof, is a big concern. IoT will potentially allow capturing, storing, and analysing an incredible amount of personal information, since we are speaking about mapping the physical world. Therefore, IoT platforms need to provide reliable systems that guarantee the privacy rights and also adequate and flexible access policies.

In the IoT-I project, a specific working group that analyses the societal impacts of IoT has been created. They are planning to release an extended report about this crucial topic in the near future.

1.5 Results

As already mentioned in the introduction, the architecture work follows a spiral design and development model. This document includes results of the first of three iteration cycles. This Section summarises the results presented in this document.

- The **vision** (Section 1) provides motivation, overview of the IoT-A architectural reference model, architecture methodology, links to other input documents, and business scenarios with identified stakeholders.
- The **IoT reference model** (Section 2) includes
  - **General discourse** about abstract quality concepts in IoT;
  - **Domain model** for the IoT, describing the main IoT concepts and their relationships;
  - **Information model** for information-storage and exchange;
  - **Communication model** for communication behaviour.
- The **IoT reference architecture** (Section 3), includes
  - **Functional view,** describing the grouping of IoT functions and their components;
  - **Security & privacy perspective,** describing architecture-quality aspects and guidelines regarding security and privacy.
- Support material, includes
  - **Requirements** (Annex A) as the basis for the functional view construction;

- **System use cases** (Annex B) for describing the behaviour of selected components in the functional view.

As the architecture work progresses, updates on the results are expected from both external inputs in form of feedbacks and requirements, as well as from research findings made in the IoT-A project. Each consecutive IoT-A architectural reference document will contain new additional materials. A roadmap for the next iteration of the architecture document is provided in Section 4.1.

2. IoT Reference model

As discussed in Section 1, the IoT reference model is an integral part of the IoT-A architectural reference model. The reference model provides the highest level of abstraction for the description of the IoT domain. A reference model captures and defines invariant aspects of a domain, viz. a set of unifying concepts, axioms and their relationships. It is important to stress that a reference model needs to be independent of specific standards, technologies, implementations, or other concrete details [MacKenzie, 2006].
The reference model also serves as a basis for the definition of the reference architecture (see Figure 3).
Alongside the definition of the reference model, common definitions for domain model and information model exist that substantially overlap with each other [IoT-A, 2011]. All models are defining entities representing concepts of the domain, their relationships, potential rules, and constraints. Yet, when looking carefully at the definitions, they do differ in scope. In our IoT reference model, the information model is domain-model specific and contributes thus to the domain model. In
Figure 4 the contributions of the above mentioned models to the reference model are depicted.

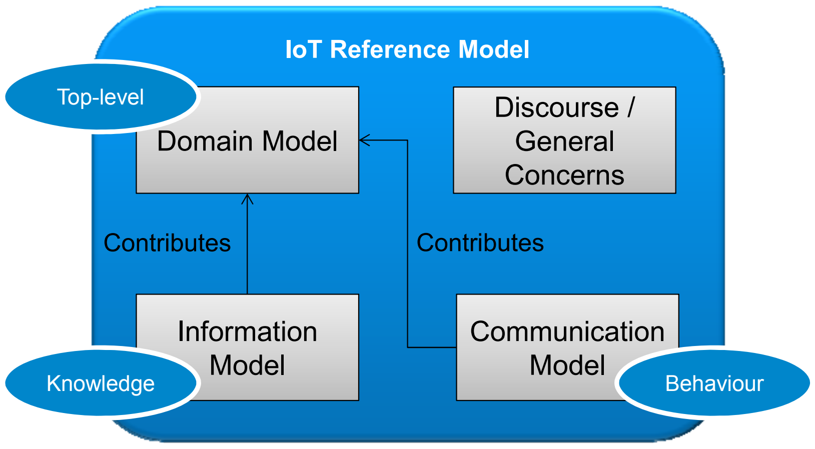

Figure 4: Contributions to the IoT reference model.

The IoT reference model, as presented in this Section, starts with a discourse about IoT. In this discourse we identify abstract quality concepts that have to be taken into account for the realisation of IoT systems (for instance, heterogeneity and interoperability). This discourse is followed by the definition of the domain model. In the domain model, concepts and entities that represent particular aspects of the IoT domain are summarised in a top-level domain description and their relations are defined. The domain model also serves as a common lexicon for and taxonomy of the IoT. Generally, entities in a domain model are either responsible for keeping track of certain information and for doing certain things. This refers to knowledge and behaviour, respectively [Oldfield, 2002]. In the IoT-A architectural reference model this knowledge is represented through an information model, which follows the definition of AutoI project [AutoI, 2011]. Its purpose is to specify the data semantics of the domain model.
Finally, the communication model addresses high-level communication paradigms pertinent to the IoT domain. The communication model presented describes how communication has to be managed in order to achieve the features required in the IoT.

A clear advantage for distinguishing between information, communication and domain model stems from the fact that models develop and stabilise over time. Improving the understanding of one model in turn allows validation of the dependent models. Where one model breaks the other model it needs to be re-modelled in order to address those inconsistencies. In the project this is enabled by different working groups contributing to the models, which establishes a conflict-rich and therefore dynamic thinking environment.

This Section first presents the discourse about the IoT domain in Section 2.1. The domain model is introduced in Section 2.2. Section 2.3 describes the related information model and Section 2.4 the related communication model.

2.1 Discourse about the IoT domain

The novelty of IoT is that it provides an extension of the Internet into the physical world. In other words, it makes the Internet aware about the physical world around it. Accepting this challenge translates into the need of explicitly modelling the physical world. Other challenges have been identified through earlier work in IoT related projects and others arise when thinking to extend the reach of the IoT according to the vision of the architectural reference model. The IoT reference architecture itself (see Section 3) can only partially address these challenges. However, based on an evaluation of different ways of addressing these challenges, it will be possible to define some best practices for developing specific IoT architectures and resulting systems. The best practices guide this development by telling an enterprise how to make best use of existing technologies and what additional software, hardware, and functionalities is needed for implementation.

As part of the IoT Reference Model we aspire to capture salient high-level IoT challenges. The list of challenges presented is non-exhaustive. The list will gradually be completed during the course of the project. The basis for the list is the project description, the state-of-the-art analysis, and the requirements analysis. Enhancements are expected in the course of the architecture work, once better understanding is reached. This understanding will stem from expected advances of the state-of-the-art. Gradually, understanding of challenges will mature and be translated into best practices with regards to the IoT reference model, viewpoints, and perspectives.

The **heterogeneity** of technologies in the IoT is significantly higher than in traditional computing systems. They include RFID, sensor networks, embedded systems and mobile technologies, and also a large variety of existing as well as emerging communication technologies. Therefore, **interoperability** has to be supported in all functionality groups (and communication layers). For communication, the co-existence of technologies, as well as bridges between different technologies, needs to be supported. For services, their integration across different technologies has to be achieved. This requires some form of common basis that has yet to be defined. At a minimum, a common way of describing the services and their interfaces has to be achieved. Interoperability not only has to be achieved across a single domain, but across different administrative and application domains.

Scalability is an important challenge when the IoT leaves the confinement of small, isolated vertical islands and becomes part of everyday life. The number of devices that need to be managed and that communicate with each other will be at least an order of magnitudes larger than the devices connected to the current Internet [Sundmaeker, 2010]. The ratio of communication triggered by machines as compared to communication triggered by humans will noticeably shift towards machine-triggered communication. Even more critical is the management of the data generated and their interpretation for application purposes. This relates to semantics of data, as well as the efficient handling of the resulting data streams.

The **manageability** of such large numbers of devices, especially in environments that cannot be centrally controlled, can only be addressed through autonomous behaviour including self-management, self-configuration, self-healing, self-optimisation and self-protection.

Besides static scenarios, such as IoT-enabled sensors being mounted to fixed infrastructure, physical entities, as well as devices, are assumed to be mobile. Therefore, they will encounter different **mobility** situations. As a result, associations between device information and services affecting physical entities need to be constantly monitored and updated in order to ensure the usability of the overall system.

Addressing **security** and **privacy** concerns becomes paramount due to the increasing pervasiveness and complexity of mostly wireless IoT system setups. Different IoT technologies have to be interoperable, and depending on computational power and energy constraints, tailor-made security mechanisms for communication have to be defined. Once data is captured and stored in an IoT system and maybe made available on the Internet, appropriate privacy mechanisms are needed. Information processing and reasoning may produce new information from “raw” data, so the system must also be able to resolve privacy settings for evolving data.

Reliability is a major factor for the acceptance of any system. For example, when looking at the nature of wireless sensor networks, it becomes apparent that the availability of information might vary over time, yet an end-user service that depends on this information still needs to respond in an appropriate way according to its initial purpose. A system has to provide functionalities that handle connectivity losses in various ad-hoc-like ways, such as caching information or finding other reliable sources of information. Due to the inherent unreliability of the IoT, which seems contradictory to the goal of reliability, new ways of making reliability issues transparent to the user have to be looked at.

In addition, the definition of the architectural reference model for the IoT as provided in this document has to consider future technology shifts, such as the appearance of new communication protocols. In order to facilitate these and related **changes**, an overall **evolution** of IoT systems future has to be addressed by an appropriate definition of the IoT reference model and the IoT reference architecture.

2.2 Domain model

2.2.1 Purpose

The IoT-A project defines a domain model as a description of objects belonging to a particular area of interest. The domain model also defines attributes of these objects, such as name and identifier. 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 [CCSDS, 2006]. Besides this official definition, and looking at our interpretation of it, our domain model also provides a common lexicon and taxonomy [Mueller, 2008]. The domain model is therefore an important part of any reference model. 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 doesn’t vary much from what does [Oldfield, 2002]. 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 issues, 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 problem domain in question. In our case the question is what defines the IoT.
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 the literature, the IoT domain suffers already from an inconsistent usage and understanding of the meaning of many central terms [Haller, 2010].

2.2.2 Main abstractions an relationships

This Section describes the IoT domain model used in the IoT-A project. It was developed by refining and extending two models found in the literature [Haller, 2010; Serbanati, 2011]. It is meant to capture the main concepts and the relationships that are relevant for stakeholders concerned with the IoT. In this and the next Section we use *italics* to refer to abstractions in the domain model. Concepts depicting hardware are shown in blue, software in green, animated objects in yellow, and concepts that fit into either multiple or no categories in brown.
The generic IoT scenario can be identified with that of a generic *user* that needs to interact with a (possibly remote) *physical entity* of the physical world (see Figure 5). 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 *active digital entity* (e.g., a *Service*, an application, or a software agent) that has a goal. The completion of goal is achieved via interaction with the physical environment. This interaction is mediated by the IoT. 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.

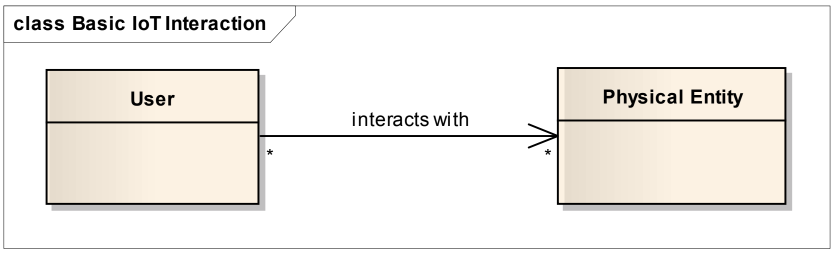

Figure 5: Basic abstraction of an IoT interaction

Physical entities are represented in the digital world via a *virtual entity*. This term is also referred to as „virtual counterpart“ in the literature [Römer, 2002], 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 entities* that are associated to a single *physical entity* that they represent. While ideally 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 the represented object. *Digital entities* can be either active elements (e.g., software code) or passive elements (e.g., a data-base entry).

- Ideally, *digital entities* are a 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.*

We define an *augmented entity* as the composition of a *physical entity* and its associated *virtual entity*. Any changes in the properties of an *augmented entity* have to be represented in both the physical and digital world. This is what actually enables everyday objects to become part of digital processes.
The relationship of augmented, physical and virtual entities is shown in Figure 6, together with all other terms and concepts that will be introduced in the remainder of this Section.
This association between *virtual* and *physical* entity usually is 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. These *devices* can be of the same technology, as in the case of body-area network nodes, or they can be of different technology, 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 devices themselves and not the entities or environments that these devices monitor.

From a functional point of view, *devices* can belong to either of three types.

- *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 camera that recognises people's faces. Information from *sensors* can be recorded in a *storage* for later retrieval.
- *Tags* are used by specialised *sensor* devices, which are usually called readers. Their sole purpose is to facilitate an 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.
- *Actuators* can modify the physical state of a *physical entity*. Actuators can move (translate, rotate, ...) simple *physical entities* or activate/deactivate functionalities of more complex ones.

Notice though, that actual *devices* can be an aggregation of several of these types. For instance, a sensor node often contains both *sensors* (e.g., movement sensing) as well as *actuators* (e.g., room-light switch). 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 *physical entities* or enable the controlling of *devices*. *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 a device. 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 associated with one or more *resources* that enable interaction with the *physical entity* that the *virtual entity* represents. This association is important in look-up and discovery processes.

Storage is a special type of *resource* that stores information coming from *resources* and that thus provides information about *physical entities*. This may include location and state-tracking information (history), static data like product type information, and many other properties. Since *storages* are *resources*, they can be deployed either on-device or in the network. On-device storages typically store information about one or only a few *physical entities*, e.g., the *physical entity* they observe. Network-based storages, such as an EPCIS repository (Electronic Product Code Information Services [EPCGlobal, 2007]), aggregate information about a large number of *physical entities*. Note that also *human users* can update the information in a *storage,* since not all known information about an entity is, or even can be, provided by devices.
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. All this is done via the network. On the lowest level –closest to the actual device hardware– , *services* expose the functionality of a *device* by accessing its hosted *resources (access)*. Other *services* may invoke such low-level services for providing higher-level functionalities, for instance executing an activity of a specified business process.

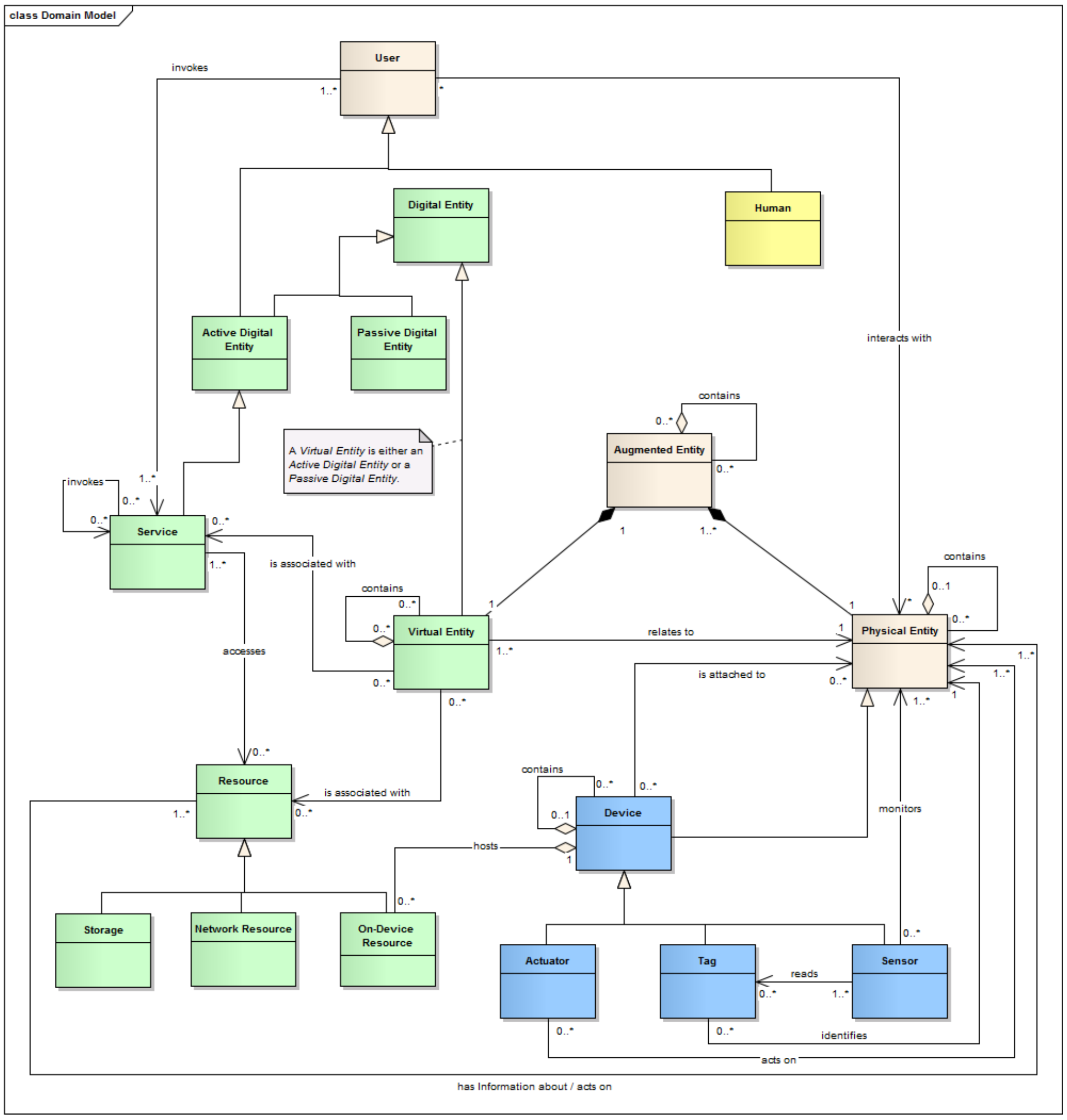

Figure 6: IoT Domain Model. All object names are explained in the main text. Hardware concepts are shown in blue, software in green, animated objects in yellow, and concepts that fit into either multiple or neither categories in brown.

Figure 7 depicts the relationship between *services*, *resources,* and *devices* and shows several deployment options. *Network-based resources* are not shown, as they can be regarded as being hidden behind cloud-based *services*.

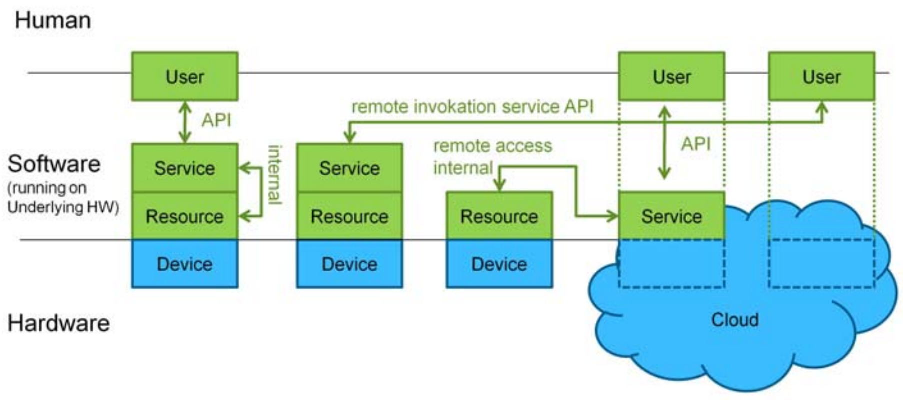

Figure 7: *Devices*, *resources* and *services*.

2.2.3 Definitions

The following table summarises the terms used in, or related to the domain model.

Abstraction	Definition	Examples
Active digital entity	Any type of active code or software program, usually acting according to a *business logic*.	• *Application* • *Service* • Software agent
Actuator	Mechanical device for moving or controlling a mechanism or system. It takes energy, usually transported by air, electric current, or liquid, and converts them into a state change, thus affecting one or more *physical entities*. (Definition based on the literature [Sclater, 2007].)	• Light switch • Robot
Address	An *address* is used for locating and accessing –"talking to"– a *device*, a *resource*, or a *service*. In some cases, *ID* and *address* can be the same, but conceptually they are different.	• IPv6 address • URL
Application	Software that implements *business logic*. *Applications* access *resources* that are needed to achieve the goal of the *business logic* through *services*. *Applications* can also provide *services*. Applications, for instance, can be implemented on a device, in an enterprise systems, or in the cloud. On-device *applications* are hardware-dependent. In some cases, their implementation can be minimal, i.e. only an extension of the OS/firmware of the *device*.	• Home-automation *application* • The firmware of an RFID tag • Software that aggregates the information collected from active GPS navigators in order to provide traffic information
Augmented entity	The composition of a *physical entity* and its associated *virtual entity*.	See • *Physical entity* • *Virtual entity*
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.	• Regulate the temperature of an environment • Check that only authorised personnel can access a building
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.	• Mobile Phone • Embedded system or sensor node with multiple *sensors* and/or *actuators* • Any *sensor, actuator,* or *gateway*

Abstraction	Definition	Examples
Digital entity	Any computational or data element of an IT-based system.	See • *Active/passive digital entity* • *Virtual entity*
Discovery	Discovery is a service for finding unknown resources/services, based on a (rough) specification of the desired result. It may be utilised by a human or another service. Credentials for authorisation are considered when executing the discovery.	• Bluetooth, *device* and *service* discovery • UDDI • Google
Gateway	*Device* that provides protocol translation between peripheral trunks of the IoT that are provided with lower parts of the communication stacks (see Section 2.4). 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.	WSN *gateway*, connecting local sensor-node *devices* to the Internet
Human	A *human* that either physically interacts with *physical entities* or records information about them, or both.	Any person
Identity	Properties of an entity that makes it definable and recognizable.	Who am I?
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*.	• EPC • URN • Biometric feature set
Infrastructure *services*	Specific *services* that are essential for any IoT implementation in order to work properly. Such *services* provide support for essential features of the IoT.	• *Resolution services* • *Look-up services* • *Discovery services*
Interface	Named set of operations that characterise the behaviour of an entity. [OGS, 2002]	Any kind of API
Look-up	In contrast to discovery, *look-up* is a service that finds existing known resources by using a key or identifier.	Data-base look-up
Network resource	*Resource* hosted somewhere in the *network*, e.g., in the cloud.	• Data repositories • See also *storage*
On-Device *resource*	*Resource* hosted inside a *device* and enabling access to the *device,* and thus to the related *physical entity*.	• *Device* driver • Programming API • See also *storage*

Abstraction	Definition	Examples
Passive digital entity	A digital representation of something stored in an IT-based system.	• Data-base entry • File
Physical entity	Any physical object that is relevant from a user or application perspective.	Pallets, boxes containing consumer goods, cars, machines, fridges etc., as well as animate objects like animals and humans.
Resolution	Query response process 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.	• EPC ONS • DNS
Resource	Computational element that gives access to information about, or actuation capabilities on a *physical entity*.	See • *On-device resource* • *Network resource*
Sensor	A *device* identifying or recording features of a given *physical entity*.	• Temperature sensor • RFID reader • Camera
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).	Web service
Storage	Special type of *resource* that stores information coming from *resources* and provides information about *physical entities*. They may also include *services* to process the information stored by the *resource*. As *storages* are *resources*, they can be deployed either on a *device* or in the *network*.	• On *device*: data cache on *gateway*, data on an RFID tag • Network-based: EPCIS repository, ERP data base
Tag	Label or other physical object used to identify the *physical entity* to which it is attached.	• RFID tag • QR code label
Thing	Generally speaking, any physical object. In the term 'Internet of Things' however, it denotes the same concept as a *physical entity*.	See *physical entity*

Abstraction	Definition	Examples
User	A *Human* or some *active digital entity* that is interested in interacting with a particular physical object.	See • *Human* • *Active digital entity*
Virtual entity	Computational or data element representing a *physical entity*. *Virtual entities* can be either *active* or *passive digital entities*.	See *active/passive digital entity*

Table 2: Terminology pertaining to the domain model.

Table 3 summarises the differences between discovery, look-up, and resolution.

	Input	Output	Output cardinality	Central Directory	Process
Discovery	Set of properties to be matched	Set of *devices* / *resources* / *services* matching the properties	Multiple	Usually not	Typically broadcast/multicast within the network within a predefined range
Look-up	ID (or set of attributes)	Information about one *physical entity* / *service* / *resource* / *device*	Single	Yes	Simple query-response to a directory
Resolution	ID	Address	Multiple	Usually multiple directories involved	Multi-step process with known starting point (i.e., initial resolution server)

Table 3: Summary of differences between discovery, look-up, and resolution.

Note that only *services*, *devices*, and possibly *resources*, are discoverable, not *physical entities.* However, services that have information about a *physical entity* can be discovered*.* Once such a *service* (or alternatively a central directory/repository) is known, it can be used to look up information about the *physical entity*.

2.3 Information model

In this deliverable, the information model is used for demonstrating how to communicate (i.e., retrieve and store information) with a *VirtualEntity.* It is also demonstrated how the pertinent data and metadata is saved. The information model is powerful enough to express device-level information and entity-level information, and to link this information together. For instance, several temperature sensors in a room can be modelled as entities with an attribute *hasTemperatureMeasurement*. The corresponding virtual entity "room" could then be modelled with an attribute *hasTemperature* that contains the aggregate values of the temperature sensor entities. The (potentially dynamic) association between the sensor devices and the entities is solved by the resolution infrastructure. The information model could support capturing these associations in two ways:

1. By keeping a history of the associations, e.g. by adding metadata to the *hasTemperature* values that link to the device entity (data provenance);
2. By recording the current association, e.g. by introducing an attribute metadata object that links the entity to the resources that are currently able to contributed to the values of this attribute.

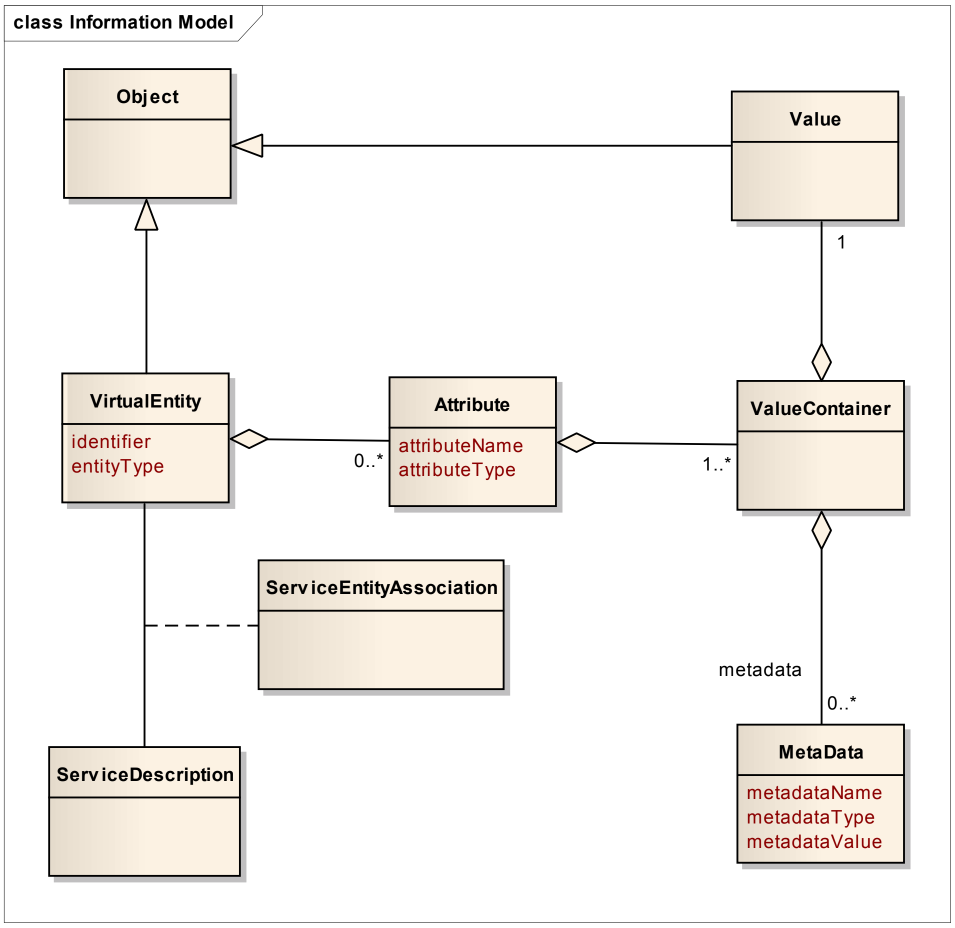

Figure 8: Information model

The information in Figure 8 shows seven components with their internal data, metadata, and their connections between each other. On the left is the *VirtualEntity* which represents the observed entity. Every *VirtualEntity* has several standard attributes. Examples for such attributes are a unique identifier or *entityType*, defining the type of the entity representation, e.g. a human, a car, or even a temperature sensor. The *entityType* may refer to concepts in an ontology that may further define additional attributes (see, for instance, [OWL2, 2009]). A *VirtualEntity* can have zero to *n* different attributes (*Attribute*). Each attribute has a name (*attributeName*), a type (attributeType) and one to *n* values (*Value*). This way one could, for instance, model an attribute *nearbyDevices,* which itself has several values. Each value is connected with zero to *n* metadata information (*MetaData*). 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* is connected to one *Value* and to zero to *n* metadata information units

(*MetaData*). The metadata can, for instance, be used to save the timestamp of the value, or other quality parameters, such as precission. The *VirtualEntity* is also connected to the *ServiceDescription* via the *ServiceEntityAssociation*. The modelling of the service description using the information model is currently in process.

In the future, the the information model will expanded to two additional usage areas:

1. **Service description:** Services provide access to resources and are used to access information or to control physical entities. A service description describes a service, using, for instance, a service description language such as USDL. The information a service provides is associated to a *VirtualEntity*. The association also captures whether the service is used for accessing information or controlling information, or both.
2. **Actuation Control:** The information model will be further reviewed to capture how entities can be controlled, for instance via actuators.

2.4 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 insight about the main interactions among the actors in the domain model. We developed a communication stack similar to the ISO OSI 7-layer model for networks, mapping the needed features of the domain model unto communication paradigms. We also describe how communication schemes can be applied to different types of networks in IoT.

2.4.1 Communication stack

This model aims at mimicking the ISO/OSI stack [ISO, 1994], but it puts the focus IoT systems requirements and characteristics.

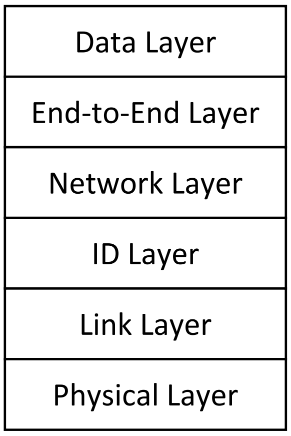

Figure 9 – IoT Communication stack

The model, as depicted in Figure 9, stresses 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 universe, 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 interfaces.

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 data layer interfaces with the data layer. A high-level description of the data pertinent to IoT is provided by the information model (see Section 2.3).

2.4.2 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.
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 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.

2.4.3 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 [Shannon, 1984].

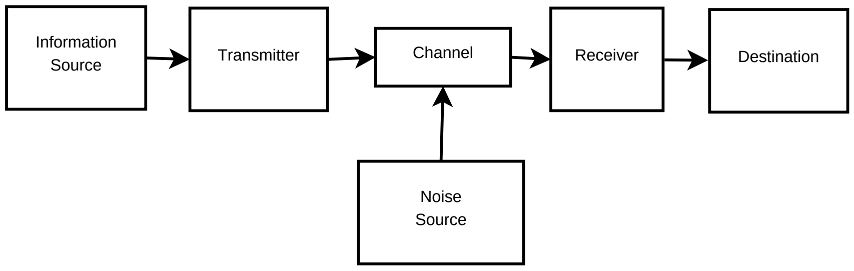

Figure 10: Schematic diagram of a general communication system.

Figure 10 depicts end-to-end communication between distant devices. The pair "information source" and "transmitter" is embodied by the digital entity (see Section 2.2), and the pair "receiver" and "destination" is embodied by a user, which could be a service, a human or, a distinct digital entity.

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 11, where the Internet provides an almost transparent "glue" between two gateways.

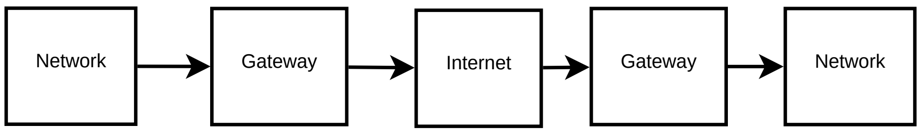

Figure 11: Channel model for the current Internet.

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 12.

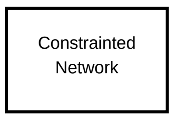

Figure 12: IoT channel for a single constrained network

In a slightly more complicated case, the IoT channel can consist of several constrained networks, which can rely on different network technologies (see Figure 13).

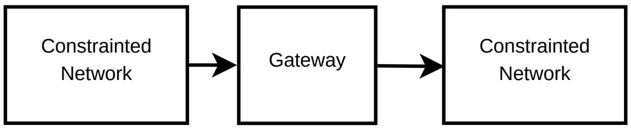

Figure 13: IoT channel for communication over two constrained networks.

A different case consists of a channel embodied by a constrained network and an unconstrained one (see Figure 14).

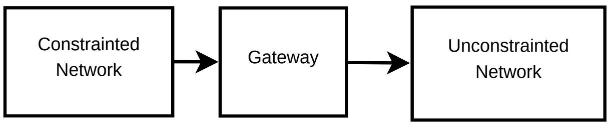

Figure 14: IoT channel for communication constrained to unconstrained networks.

An additional case consists in a channel formed by two constrained networks intermediated by an unconstrained one (see Figure 15).

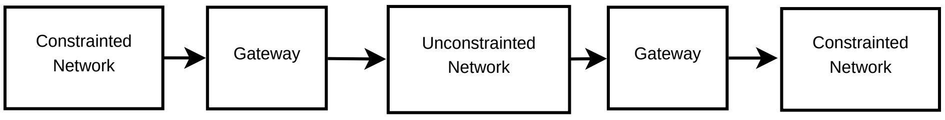

Figure 15: IoT channel for communication over two constrained networks intermediated by an unconstrained one.

The case we consider the most important in the IoT is the one involving two constrained networks linked by the Internet (see Figure 16).

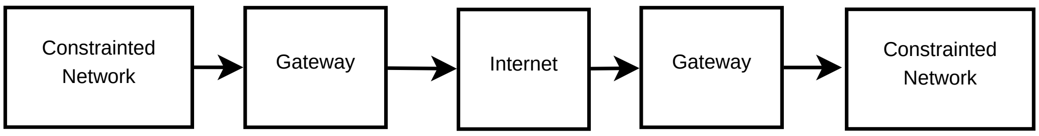

Figure 16: IoT channel for communication over two constrained networks intermediated by the Internet.

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.

2.4.4 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 cartoon of IoT communication can is provided in Figure 17, 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.

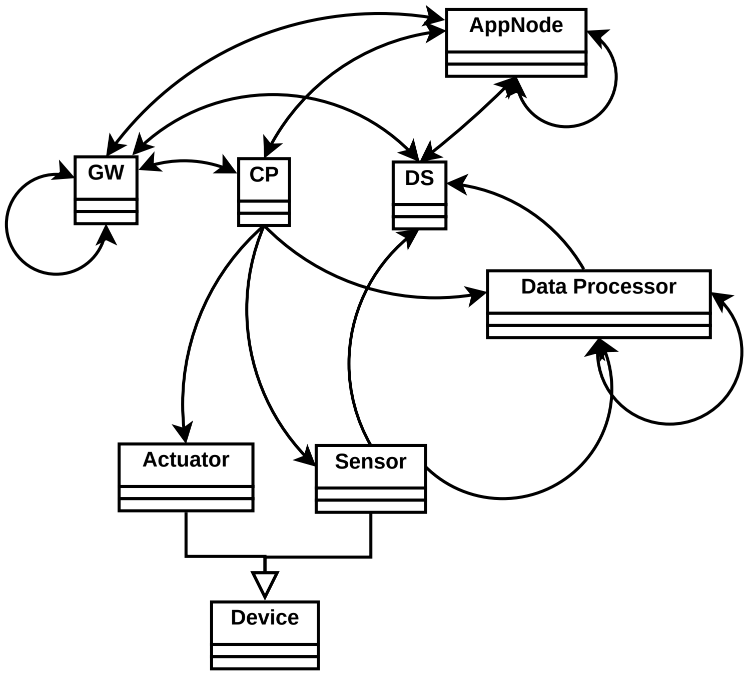

Figure 17: Communication layer of the IoT domain model from an application point of view. AppNode: application node; GW: gateway; CP: control point; DS: data sink.

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.

Data sink (DS): A data sink 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 sinks 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): 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 sink need to be hosted in the same network. A gateway can obviously communicate with other gateways and forward traffic from control points, data sinks, data processors, and AppNodes.

3. Reference architecture

As discussed in Section 1, the IoT Reference Architecture consists mainly of views and perspectives. What views and perspectives depends on the unified requirements that are inferred from application-specific requirements. By aid of our requirements-engineering process (see Section 1.3.2) we identified a reference architecture that contains the views and perspectives shown in Figure 18.

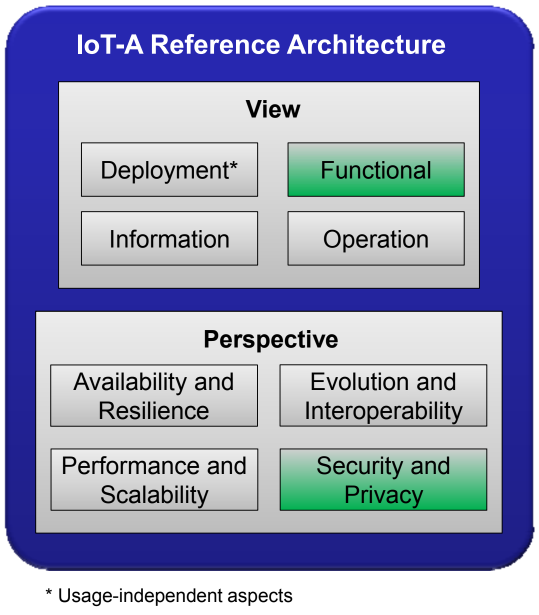
IoT-A Reference Architecture
View
Deployment*
Functional
Information
Operation
Perspective
Availability and Resilience
Evolution and Interoperability
Performance and Scalability
Security and Privacy
* Usage-independent aspects

Figure 18: Detailed view of the IoT reference architecture. Green boxes: addressed in this deliverable.

Of the views identified in our requirements process, only the functional view is addressed in this report (see Section 3.1). Of the perspectives identified only security and privacy are addressed (see Section 3.2). The other views and perspectives will be dealt with in future deliverable addressing the architectural reference model (see Section 4.1).

3.1 Functional view

To define a functional view for the IoT-A reference architecture, we first identified the key functional groups such an architecture needs to provide in order to meet the requirements identified during our requirements-engineering process. In total, seven functional groups were identified.

- From the four central abstractions identified in the domain model (physical entities, devices, resources, and services), we derived the **‘virtual entity (VE) and information’** and the **‘IoT service & resource’** functionality groups. The former provides functionalities for accessing VEs and devices, while the latter provides functionalities for accessing resources and services.

- With regards to the plethora of communication technologies that the IoT-A reference architecture needs to support, we identified the need for a '**device connectivity and communication**' functionality group.
- Requirements expressed by stakeholders regarding the possibility to build services and applications on top of the IoT are covered by the '**process execution and service orchestration'** and **'application'** functionality groups.
- To address consistently the concern expressed about IoT security and privacy, we grouped the required functionalities in a '**security'** functionality group.
- Finally, a **'management'** functionality group is required to manage the different functionality groups.

Furthermore, for diverse reasons, such as interoperability and system modelling, each functionality group was subdivided into functional components. Also, each link between functionality groups is equipped with two interfaces.

3.1.1 Functionality groups

The functional decomposition of the IoT-A reference architecture is depicted in Figure 19. As discussed before, functional components are grouped in seven functionality groups. These seven groups are briefly described bellow. More detailed descriptions of these groups components are provided in Sections 3.1.2 and 3.1.3.

1. **Applications:** This group describes the functionalities provided by applications that are built on top of an implementation of the IoT-A architecture.
2. **Process execution and service orchestration:** This functionality group organises and exposes IoT resources so that they become available to external entities and services. Through this set of functionalities, and the APIs that expose them, IoT Services become available to external entities and can be composed by them.
3. **Virtual entity (VE) and information:** This group maintains and organises information related to physical entities, enabling search for services exposing resources associated to physical entities. It also enables the search for services based on the physical entity they are associated to. When queried about a particular physical entity, this functionality group will return addresses of the service related to this particular physical entity.
4. **IoT service & resource:** When queried about a specific service, this group will return its description, providing links to the exposed resources. This group also provides the functionalities required by services for processing information and for notifying application software and services about events related to resources and corresponding physical entities.
5. **Device connectivity and communication:** This functional block provides the set of methods and primitives for device connectivity and communication (the first referring to the possibility for a device to be part of a network, the second to the possibility for this device to be source or destination of messages). Also, this group contains methods for content-based routing.

In addition to these "longitudinal" functionality groups, two sets of "transversal" groups were identified. 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. Indeed, 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. The same applies to quality-of-service expectations that should not be too high on certain components and too lax on others.

6. **Management:** In order to manage computational resources efficiently, management has to be handled by a single group of functionalities.
7. **Security:** Security functions have to be consistently applied by the different groups of functionalities. Specifically, access-control policies shall consistently be applied in order to prevent unauthorised applications from obtaining access to sensitive resources. Privacy will also be enforced through pseudonymity, i.e. different (generic) identities can be used by a user when accessing IoT services.

The rest of this Section details the above functionality groups and describes their components, as well as the interfaces that link them. First we start by depicting the global functional decomposition (see Figure 19), and then describe the functionality-group components, starting with the longitudinal functionality groups (Section 3.1.2). Finally, we detail the transversal functionality groups (Section 3.1.3).

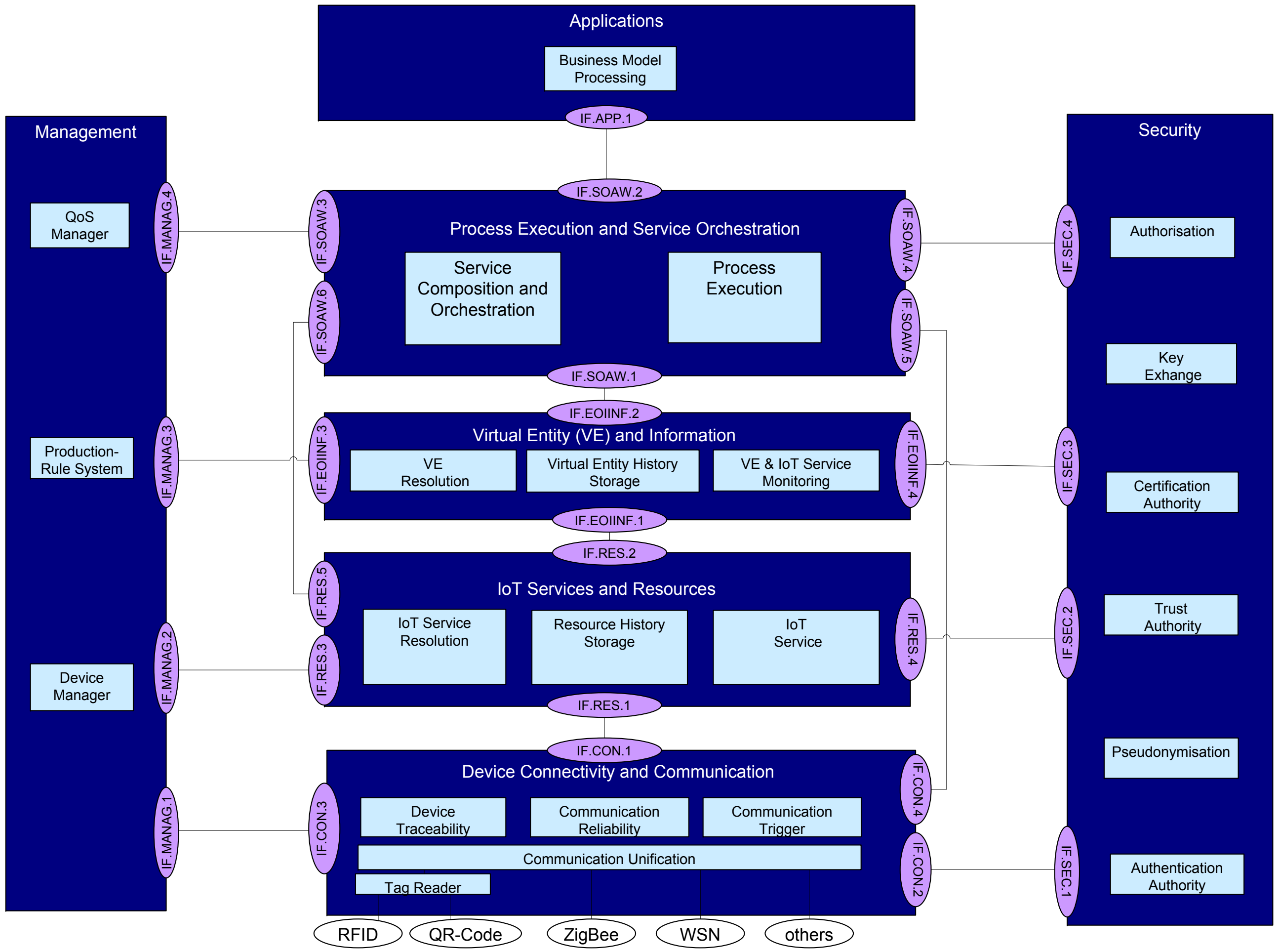

Figure 19: Diagram depicting the functional view of the IoT reference architecture. Each major box represents a functionality group, while the smaller boxes represent functional components. The lines between functionality groups –terminating in ellipses- represent interfaces.

Process execution & service orchestration	**Virtual entity & information**	**IoT service & resource**	**Device connectivity & communication**
Service composition and orchestration	***Virtual-entity resolution***	***IoT service resolution***	***Communication unification***
Execute IoT-aware process models defined by process-modelling applications in the application layer. This is achieved by utilising IoT services orchestrated according to the service-composition and orchestration component. General tasks are: • Deploy process models for planning service executions, • Instantiate these services.	This functional component provides functionalities for retrieving lists of services that expose resources related to particular virtual entities. If the VE is not clearly identified by the user, this block will also provide the functionality to retrieve a VE based on a description.	Maintain and provide information regarding an identified service: • This component can be used to update the description of a service, • Retrieve this description, • Provide the address of the identified service.	This functional component provides an access to IoT devices that is agnostic of the devices technology. It also ensures that all devices can interoperate. The main solution for reaching this will be to provide bridges between different protocol stacks and to identify convergence points.
Process execution	***Virtual-entity & IoT-service monitoring***	***Resource history storage***	***Communication reliability***
Make use of the functionalities provided by the IoT-services-and-resources functionality group to: • Increase quality of information, • Support flexible services, • Orchestrate IoT services.	This functional component maintains associations between VEs, resources, and exposed services.	Provides storage capabilities for measurements generated by resources (resource history). It also provides additional services associated to the processing of stored information. For deployment consideration, it should be noted that this component and the virtual-entity history storage can be hosted by the same hosted in the same storage.	Given the heterogeneity of information flowing through the IoT, this functional block will provide uniform interfaces for retrieving data from different sources. It will use the most efficient communication protocol according to delay-sensitivity communication.
	Virtual-entity history storage	***IoT service***	***Device traceability***
	Publish integrated context information (PE context information - dynamic and static), PE state information, PE capabilities. For deployment consideration, it should be noted that this component and the Resource History storage could be hosted by the same entity.	Interpret and process information based on rules or processes defined by a user/application. This might even include data-mining processes that periodically analyse information and send notifications to consumer of the service.	Most IoT devices are subject to different availability (duty-cycling, passive RFID, etc.). This functional component provides methods for enhancing device traceability. Examples for such methods are hand-over, access logs, etc.
			Communication trigger
			Triggers the establishment of communications based on policies, events, or schedules.
			Tag reader
			Read tag values and act as a communication interface to the tags.

Table 4: Description of longitudinal functionality groups.

3.1.2 Components and functionalities of longitudinal functionality groups

Once we identified the functionality groups shaping the high level functional decomposition of the IoT reference architecture, we subdivided them into functional components and depicted a more accurate picture of the architecture (see Figure 19). In a next step, the functionalities that shall be provided by the group, i.e. its functional components, were defined in more detail. The result of our analysis is provided below.

3.1.2.1 Application

Component: business-process modelling	
Purpose: 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 application layer, as it is an external, but necessary, tool used to build applications based on the IoT-A architecture. **Fulfils requirements:** UNI.6, UNI.31	
Process-models designer	*The application provides the capabilities of creating executable model representations, e.g. in a BPMN-2.0-derived format, that can be executed in the process-execution component.*
IoT business-processes modeller	*Provides the tools necessary for modelling business processes using the standardised notation,[2] i.e. using novel modelling concepts specifically addressing the idiosyncrasies of the IoT ecosystem.*

3.1.2.2 Process execution and service orchestration

Component: service composition and orchestration	
Purpose: Make use of service functionalities provided by the IoT-services & resources functionality group to: • *Increase quality of information,* • *Support flexible services,* • *Orchestrate IoT services.* **Fulfils requirements:** UNI.8, UNI.10, UNI.43, UNI.64, UNI.87	
Increase quality of information	*This functionality group 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.*
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.*

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

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 composition or complex event processing.*
Set service priority	*Supports prioritisation of services.*
Component: process execution	
Purpose: Executes IoT-aware process models, which are defined by process-modelling in the application layer. This execution is achieved by utilising IoT services that are orchestrated by the service-composition-and-orchestration component. **Fulfils requirements:** UNI.8, UNI.64	
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.*
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 process-execution component utilises the service-compositio-and-orchestration component.*
Run application	*After resolving IoT services, the respective services are invoked using the service-composition-and-orchestration component. 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.*

3.1.2.3 Virtual entity and information

Component: virtual-entity (VE) resolution	
Purpose: This functional component maintains the link between a virtual entity and the resources that are associated to it. Through this component, it is possible to retrieve a list of services exposing resources related to a virtual entity, which is either already known by the requestor or might be discovered by providing specifications of the virtual entity. **Fulfils requirements:** UNI.16	
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.*
Lookup VE-related services	*Searches for services exposing resources related to a virtual entity.*
Update VE-associations	*Updates associations between a physical entity (and the related virtual entities) and the IoT resources that are associated to this entity.*
Component: virtual-entity & IoT-service monitoring	
Purpose: Maintains associations between virtual entity, resources, and exposed services related to this physical entity. **Fulfils requirements:** UNI.38, UNI.43	
Monitor VE-resource association	*Monitors associations between VEs and the IoT resources hosted by devices attached to this VE.*
Monitor VE-Service association	*Monitor existing associations between VEs and services.*
Assert VE-Service association	*Asserts a static association between a VE and a service. Due to the static nature of the association, it does not have to be*

	monitored.
Component: virtual-entity history storage	
Purpose: Publishes integrated context information (PE context information - dynamic and static) PE state information, PE capabilities. For deployment consideration, it should be noted that this component and the Resource History storage could be hosted by the same entity. **Fulfils requirements:** UNI.41, UNI.46	
Get VE history	*Stores and retrieves information recorded about a virtual entity.*

3.1.2.4 IoT service & resource

Component: IoT-service resolution	
Purpose: Maintains and provides information regarding an identified service. ○ *This component can be used to update the description of a service,* ○ *Retrieve this description,* ○ *Provide the address of the identified service.* **Fulfils requirements:** UNI.4, UNI.30, UNI.74, UNI.75	
Update service Description	*Modifies the description of the resource exposed by an IoT service.*
Resolve service	*Resolves the address of an IoT service.*
Get service description	*Retrieves the description of an IoT service.*
Get Service exposing a resource	*Retrieves a list of services exposing the searched resource.*
Annotates resource from device description	*Semantically annotates information based on the device description. This functionality provides the metadata required to interpret the information that the device provides about the physical entity.*
Component: resource history storage	
Purpose: Provides storage capabilities for the measurements generated by resources (resource history). It also provides additional services associated to the processing of the stored information. For deployment consideration, it should be noted that this component and the virtual-entity history storage could be hosted in the same storage. **Fulfils requirements:** UNI.41, UNI.46	
Get resource history	*Retrieves the list of information that has been recorded by a resource (resource history).*
Component: IoT service	
Purpose: Interprets and processes information based on rules or processes defined by a user/application. This might even include data-mining processes that periodically analyse information and send notifications to consumer of the service. **Fulfilled Requirement:** UNI.18, UNI.27, UNI.59, UNI.74	
Process Information	*Interprets and processes information based on the device description.*

3.1.2.5 Device connectivity and communication

Component: communication unification	
Purpose: This functional component provides access to IoT devices. The component is agnostic in respect to the devices technology. The main solution for reaching this will be to provide bridges between different protocol stacks and to identify convergence points. **Fulfils requirements:** UNI.3, UNI.16, UNI.47, UNI.48, UNI.49, UNI.71	
Find a common gateway	*Finds for two devices the lowest layer (in the IoT stack) that implements interoperable technologies and allows these two devices to communicate.*
Tag publishing	*Unlike active devices that are autonomous, tags need a reader in order to send/provide information. This functionality exposes tags as virtual entities.*
Publish device associations	*Informs the resource-and-information functionality group of the current aggregation status at the device level.*
Monitor device associations	*Monitors association and grouping of devices behind a gateway.*
Assess device description	*Enforces the compliancy between semantic device descriptions, so that information exchange between these devices is possible. Indeed, for devices to be able to exchange information, a common language needs to be used. This language is used for describing the information and making sure that information is consistently exchanged. This function ensures that the device description is consistent with this language.*
Component: communication reliability	
Purpose: Given the heterogeneity of information flowing through the IoT, this functional block provides uniform interfaces for retrieving data from different sources. It uses the most efficient methods according to data sensitivity and delay tolerance of the requesting application. **Fulfils requirements:** UNI.26, UNI.28, UNI.29, UNI.50, UNI.58	
Get route for a specific content	*Routes the messages according to their content.*
Transmit delay-sensitive information	*Transmits delay-sensitive information.*
Setup time-sensitive communication	*Supports reliable communication between devices hosting time-sensitive resources.*
Component: device traceability	
Purpose: Most IoT devices are subject to different availability (duty-cycling, passive RFID, etc.). This functional block will provide methods for enhancing device traceability, such as hand-over, access logs, etc. **Fulfils requirements:** UNI.12, UNI.20, UNI.21, UNI.45, UNI.50, UNI.51	
Check device authorisation	*Verifies that the device is registered and authorised to communicate on the network.*
Check transmission activity	*Provides real-time status information of transmission activity.*
Initialise device roaming	*Updates locator when a device changes network location.*
Component: communication trigger	
Purpose: Triggers the establishment of communications based on policies, events, or schedules.	

Fulfils requirements: UNI.17, UNI.49

3.1.3 Components and functionalities of transversal functionality groups

3.1.3.1 Management

Component: QoS manager	
Purpose: Manages the QoS when using functionalities provided by the different components of the architecture. This information is then provided to services and applications that make use of this resource. **Fulfils requirements:** UNI.59, UNI.60, UNI.61	
Assess policy	*Manages consistency of the QoS requirements expressed and supported by the different functionality components*
Get QoS policy	*Informs the 'process execution and service orchestration' and 'device connectivity and communication' of the QoS required/supported by the requesting application.*
Component: device manager	
Purpose: Manages device at the hardware/firmware level **Fulfils requirements:** UNI.14, UNI.15, UNI.19, UNI.55	
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.*
Component: production-rule System	
Purpose: This component will be used to express and enforce a set of conditions that, once fulfilled, will automatically trigger some pre-defined action. Such rule could be used to verify the integrity of a virtual entity, services, and the platform. Initialise signal failure by triggering an alarm signal **Fulfils requirements:** UNI.32,UNI.66	

3.1.3.2 Security

Component: authorisation	
Purpose: Based on the policies set by the owner/administrator of a resource/service, the functional component authorisation decides about whether a requested access should be granted or denied. **Fulfils requirements:** UNI.1, UNI.2, UNI.22, UNI.24, UNI.40, UNI.62, UNI.67	
Check access authorisation	*Controls access to functionalities and information in the different functionality groups based on the requestor ID (e.g., directly/in SOA services/applications).*
Select secured communication protocol	*Selects a secured communication protocol supported by the device and adapted to the resource sensitivity (device resources/supported protocol/sensitive data).*

Tag resource as sensitive environment	*Stores sensitive resources in a safe environment. The database managing these sensitive resources duplicates the resources to enforce reliability.*
Component: key exchange	
Purpose: This functional component is provided by a trusted entity. It distributes symmetric keys for M2M communication. These keys can be of temporary nature for pseudonymisation and concealed aggregation. **Fulfils requirements:** UNI.22, UNI.24	
Component: certification authority	
Purpose: Provides certificates binding an Virtual Entity to defined attributes like : • *IP addresses* • *Public keys* *In addition, it can assert certificates provided by another certification authority.* **Fulfils requirements:** UNI.22, UNI.24	
Component: authentication authority	
Purpose: Authenticates the user and provides assertion of its identity or chosen pseudonym. Additional attributes, e.g. roles, can be added to the assertion. It can use federation mechanisms for authentication between different domains. **Fulfils requirements:** UNI.22, UNI.24	
Component: trust and reputation	
Purpose: Maintains reputation of each device or service based on recommendations and feedbacks received from other devices and direct observations of device behaviours and measurement accuracy. This functionality might be centralised for a specific domain. **Fulfils requirement:** UNI.40, UNI.62	
Evaluate resource reliability	*Asserts that the device hosting the accessed resource is trustworthy enough for this resource to be used by a critical service or application.*
Component: pseudonymisation	
Purpose: Provides functions required to support user privacy (mostly through anonymity/pseudonymity); covers the creation and management of pseudonyms either for • *Users that activate a pseudonym during the authentication or at a later point in time;* • *Resource/devices that utilise pseudonyms to protect the privacy of the user.* **Fulfils requirements:** UNI.1, UNI.2, UNI.40, UNI.61	
Sanitise data set	*Removes traces of the user ID in a given data set or during an access to the service.*

3.2 Security perspective

This Section describes the security perspective of the reference architecture. The Section is divided in two parts dealing with security and privacy at communication level and within the infrastructure services, respectively. The general approaches to communication security and the functional components and interactions needed for achieving system-level privacy and security are described in the following.

3.2.1 Communication security

The communication security model is an architecture primer for enabling security features in IoT communication solutions. As stated elsewhere [Bui, 2011], securing the communication at protocol level is very difficult in the case of IoT, since resources are typically 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 devices and unconstrained devices. 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 elsewhere in the literature [Bui, 2011].
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 reference architecture. The same issue may arise again when implementing a concrete architecture. As in the communication model (see Section 2.4), we address the problem by separating domains of high heterogeneity and demanding constraints from the more homogeneous domain. We are also providing a standard interface between the two.

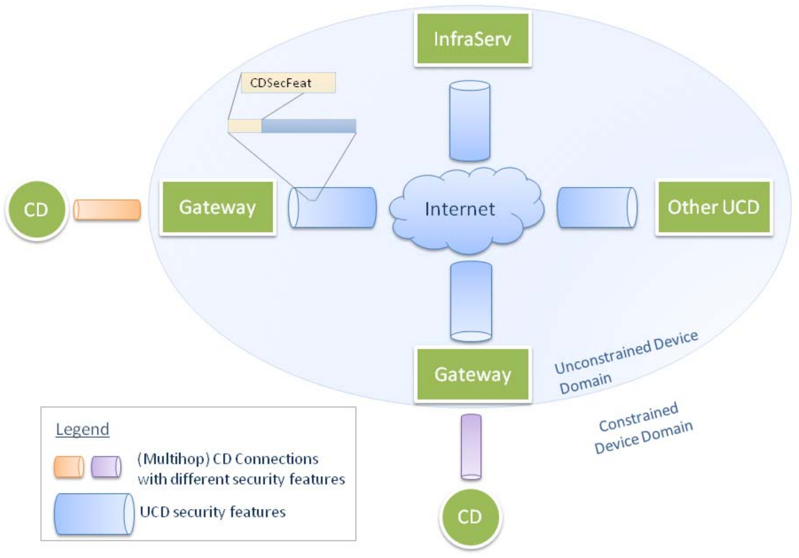
InfraServ
CDSecFeat
CD
Gateway
Internet
Other UCD
Gateway
Unconstrained Device Domain
Constrained Device Domain
CD
Legend
(Multihop) CD Connections with different security features
UCD security features

Figure 20: Providing the best security features for the lower layers in each IoT domain by introducing *Gateways* with active functions. CD: constrained device; UCD: unconstrained device. CDSecFeat: security feature for constrained device.

The solution adopted is based on 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. This usually involves the adaptation between different protocol-layer implementations up to the network layer (see Section 2.4). The fact that gateways are generally unconstrained devices means that they can also be used for scaling down functionalities (such as security) from the UCD domain to the CD 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 itself and other nodes of the UCD domain. (Optional; impacts on trust assessment.)
- Management of security features belonging to the peripheral network. (Optional)
- Description of security options related to outgoing traffic. (Authentication of source node, cryptographic strength, ...)
- Filtering of incoming traffic according to user-defined policies, which take into account security options of incoming traffic, destination-node preferences and so on. (Optional)

Gateways are not relevant and thus invisible at the end-to-end layer level. Despite end-to-end security, lower layers may use heterogeneous security features across network sub-domains or for point-to-point communication. 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 take 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.

3.2.2 Infrastructure services for enabling security and privacy

In order to achieve security and privacy at the application-layer level, a set of security features must be provided on top of the security features provided:

- Trust of IoT infrastructure components (e,g. resolution/lookup, authorisation, or certification authority)
- Trust of IoT actors (IoT-service invokers and providers)
- Accountability of actions performed through the IoT
- Privacy for data handled by the infrastructure
- Privacy related to sensitive resources when they are provided to users

Note that, while accountability and privacy features might be conflicting in some cases, the mechanisms for providing both are needed, and concrete architectures derived from the reference architecture should balance the specific tradeoffs.

The aforementioned privacy and security features require the provision of the following mechanisms, which can be viewed as technical requirements.

- Access control for resolution/lookup services.
- Access control for IoT services, providing access to resources.
- Pseudonymisation of humans as well as IoT services (client and provider side); this also applies to the related virtual entity.
- Authentication of users.
- Confidentiality, integrity, and freshness of exchanged messages. Freshness is a communication-security concept, implying that replay attacks cannot be performed. This has implications for the authentication of the endpoints of a communication path and the encryption used for communication.

In order to satisfy these requirements, the following components of the security domain have been identified:

- Authorisation (AuthS)
- Trust and reputation (TRA)
- Authentication (AuthN)
- Pseudonymisation (PN)
- Key exchange and key management (KEM)
- Certification authority (CA)

Such components may be co-located and aggregated, or could be distributed geographically and operate on their own. It is also likely that these components will be run by different operators according to specific policies reflecting relative trust.

3.2.2.1 Authentication (AuthN)

A block diagram of this functional component can be found in **Figure 21**. It authenticates users accessing resources in IoT-A based architectures through IoT service clients. Authentication means are generic. In case authentication is certification-based, the authentication component will leverage the certification authority.

In most cases, the request of information or resolution of a service is triggered on behalf of a particular user. This scenario is also valid for application acting autonomously. This user has to be authenticated and the related assertion that a service client is acting on behalf of a user has to be provided. This functionality is provided by the authentication component and can be invoked by:

- A user who needs an ID assertion for interacting with other components (e.g., authorisation, trust & reputation, ...) or for accessing resources through IoT-Services. This can occur offline. In the later case, the user first requests an assertion of his identity that will be used in a second step[3] for requesting the needed credential from the authorisation components (in conjunction with the key exchange). These credentials can then be used offline.
- An IoT Service (provider side), which received a request from a user. The user provided an ID assertion in the request. The IoT service is tasked to verify the ID assertion, and for this purpose it contacts the authentication component. This component verifies the assertion of the identity or of the chosen pseudonym.
- An IoT-service provider who becomes active and want to join a system secured through a Kerberos-like protocol.
- The authorisation component that needs to verify the ID assertion that was provided by a user when requesting offline authorisation credentials.

The authentication component might trigger an action on the key-exchange-management component as a consequence of successful or unsuccessful authentication. This component might use federation mechanisms for authentication between different domains according to regional or enterprise-defined policies.

[3] Assuming that the authentication component and authorisation component are not co-located and cannot interact directly.

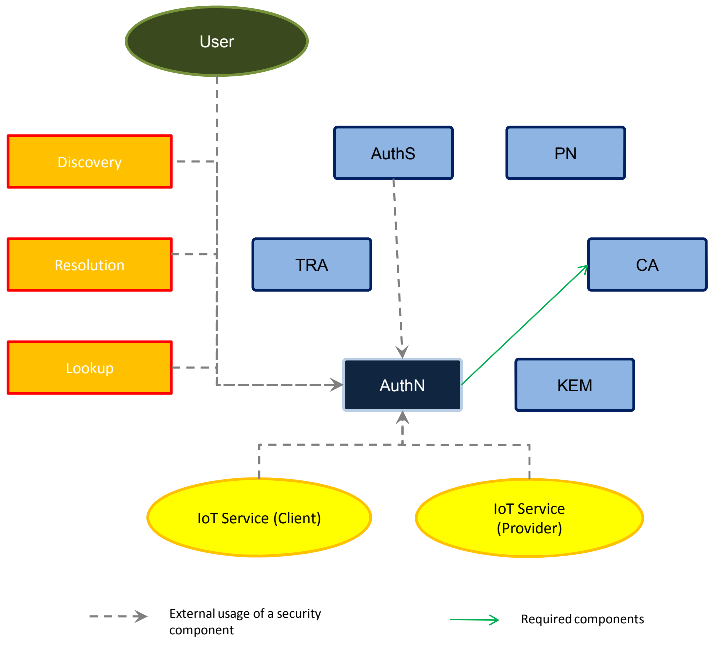

Figure 21: Usage and dependencies of the authentication component.

3.2.2.2 Authorisation (AuthS)

A block diagram of this functional component can be found in **Figure 22**. The authorisation component controls the access to the information (including resolution and discovery information) based on the policies set by the owner/administrator of a resource/service. Authorisation decides on whether an access should be granted or denied.
Two general approaches have been identified

- *On-the-fly:* An enforcement point intercepts all access to the resources/services by an authenticated user and triggers the authorisation component to evaluate the access policies.
- *Credential-based:* The user presents credentials which legitimate an access. Additionally, attributes (e.g. roles) might be included in the request. The authorisation component is responsible for deciding what access privileges should be encoded in the credential.

The first approach is a conventional solution for all kind of resource access including Web services. The second approach is used in case the availability or communication with all entities cannot be guaranteed. This case is relevant for nomadic users. In both approaches, additional information or credentials (e.g. on trust) can also be taken into account when deciding about granting or denying access.
The component is also used by IoT services that want to provide offline access to their resources. This is done in order to obtain the (cryptographic) means for verifying the offline credentials of potential users.
Storing and managing the changes to the access privileges is also a task for the authorisation component, which is quite different for each approach. The authorisation component might be split into sub-components related to policy enforcement, decision, and administration (PEP, PDP, and PAP,[4] respectively).
This component might use federation mechanisms for authentication between different domains according to regional or enterprise-defined policies.

[4] Policy-enforcement point, policy-decision point, policy-administration point; respectively.

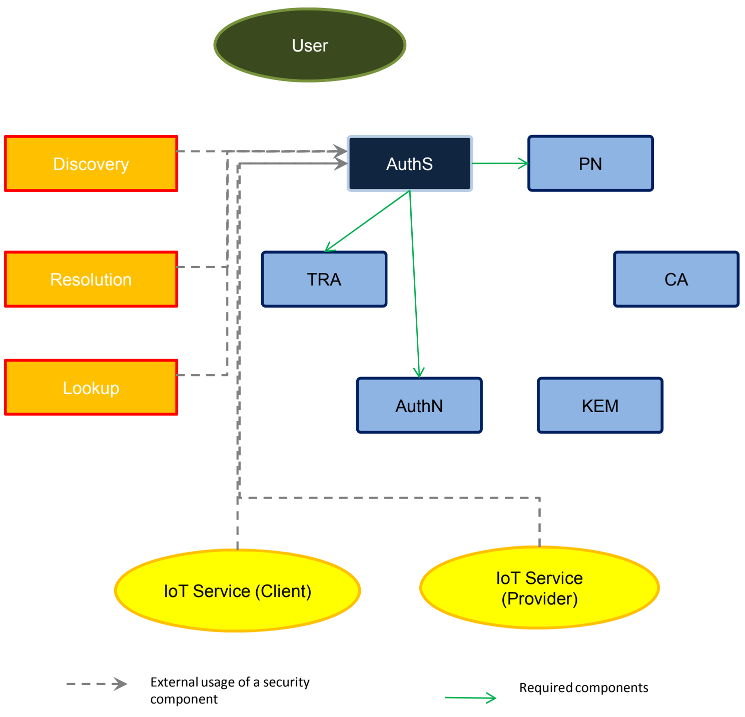

Figure 22: Requirements and usage of the authorisation component.

3.2.2.3 Trust and Reputation (TRA)

A block diagram of this functional component can be found in **Figure 23**. As different players are interacting with each other, a method to establish trust is needed. This trust encompasses legitimate behaviours of a player, as well as its reputation regarding its ability to judge the trust of other players. Since a globally centralised system is not a viable option for the IoT, a functional component that records maintains the reputation of each device or service is needed. This reputation can be based on the recommendations and the feedback received from other devices, infrastructure services (such as authentication), and direct observations of IoT services. It can also be based on measurements of the precision of the data provided. The presence of security features at lower protocol layers,[5] as well as the strength of the security features, are also taken into account.
This functionality can be embodied by a federated distributed architecture operating in smaller domains, but it has to be fault tolerant within these domains.

[5] Pertinent topics are: authentication features of RFID tags; confidentiality of the peripheral network.

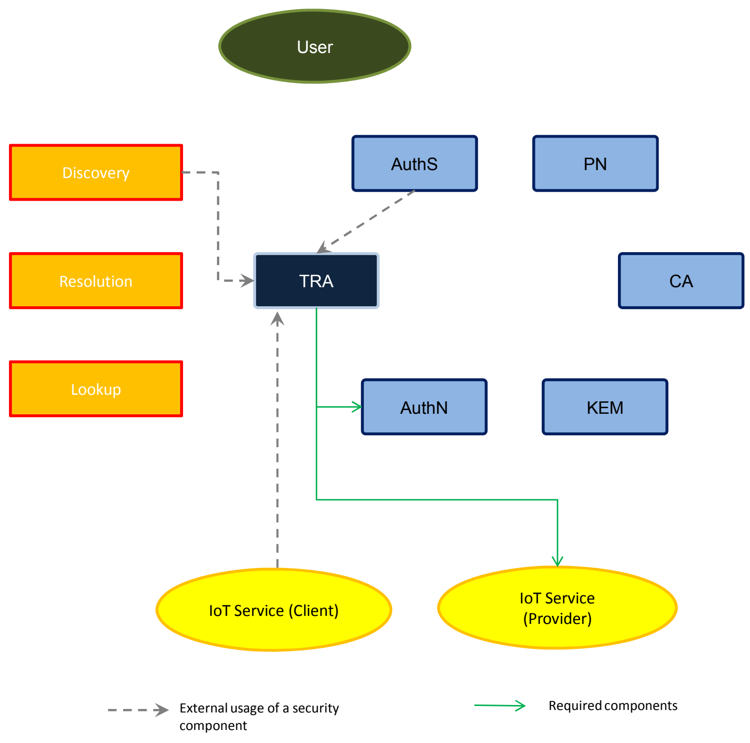

Figure 23: Dependencies and usage of the trust-and-reputation component.

3.2.2.4 Pseudonymisation (PN)

A block diagram of this functional component can be found in Figure 24. This component covers the creation and management of pseudonyms either for

- Users, who use a pseudonym for authentication purposes or at a later point in time to interact with an IoT service; or
- IoT services providing access to resources that need to use pseudonyms to protect the privacy of the owner or user of the augmented object;

The pseudonymisation component is also used by the authorisation component in order to determine whether the user presenting a pseudonym assertion is entitled to access a given resource/service.

During the creation of a certificated pseudonym, the key pair used for registering the related certificate with the CA is created by the KEM.

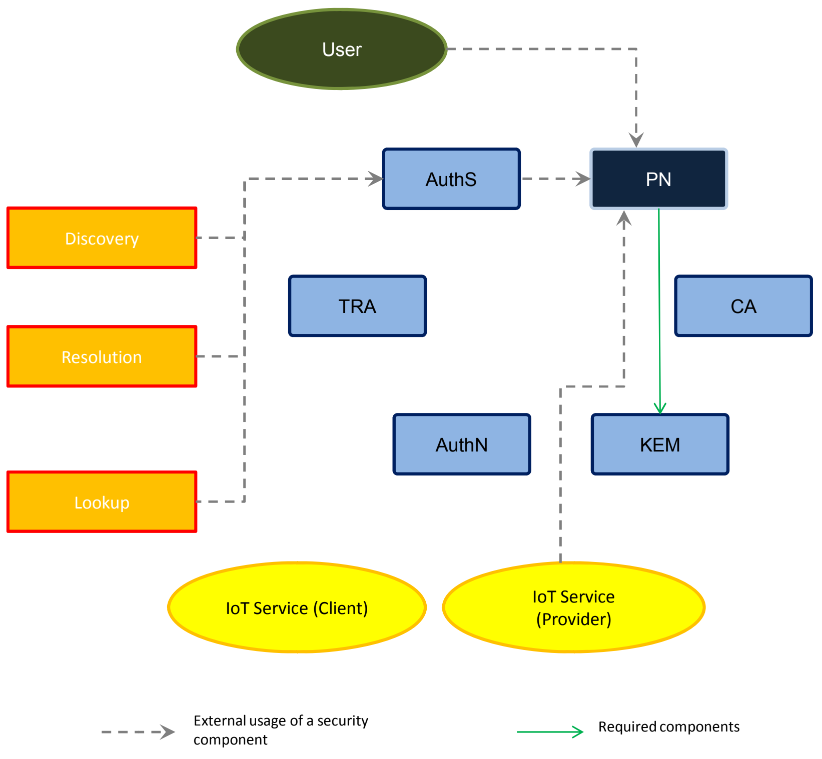

Figure 24: Dependencies and usage of the pseudonymisation component.

3.2.2.5 Key Exchange and Management (KEM)

A block diagram of this functional component can be found in Figure 25. This component provides the functionality for creating, distributing, and managing keys. Such keys can either be symmetric (M2M communication) or asymmetric (for pseudonymisation and concealed aggregation). Generated keys might be temporary and thus used for single, limited-time-span tasks. Keys with a greater longevity need to be registered at the certification authority.
Keys used for communication encryption are stored on the key-exchange-management component, while those used for authentication are stored in the certification authority.

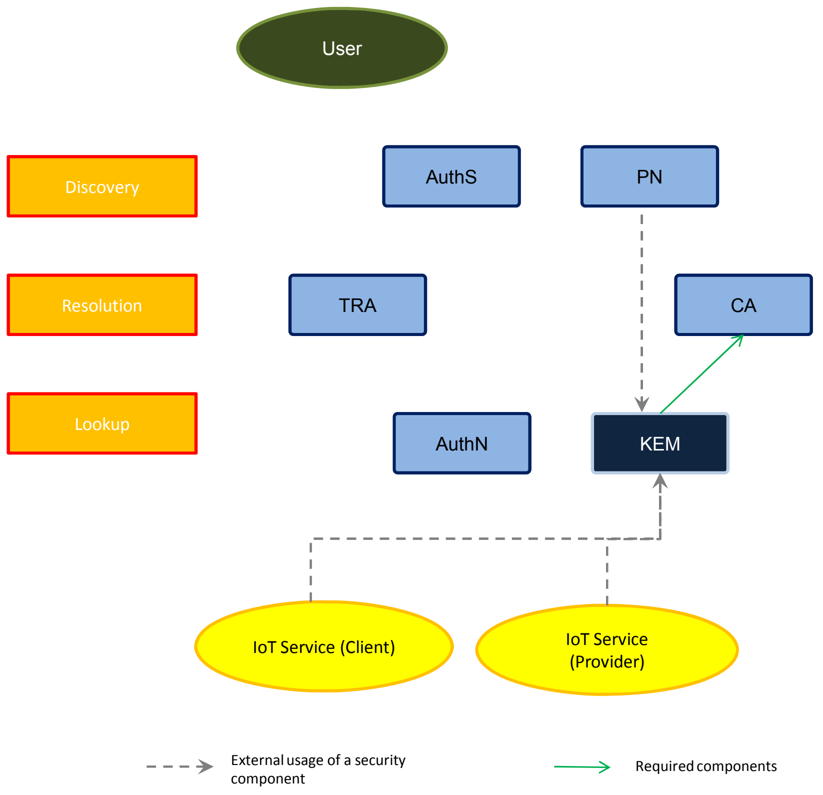

Figure 25: Dependencies and usage of the key-exchange and key-management component.

3.2.2.6 Certification Authority (CA)

A block diagram of this functional component can be found in Figure 26. This component is a legacy component, which provides almost the same features as certification authorities. Specifically, it provides certificates that are binding a service (provider- or client-side) to defined attributes like

- IP addresses;
- Public keys.

Based on such certificates, secure service-based communication can be established. Other components, like trust and reputation, as well as authorisation, rely on this component to link their activities to the correct subjects.

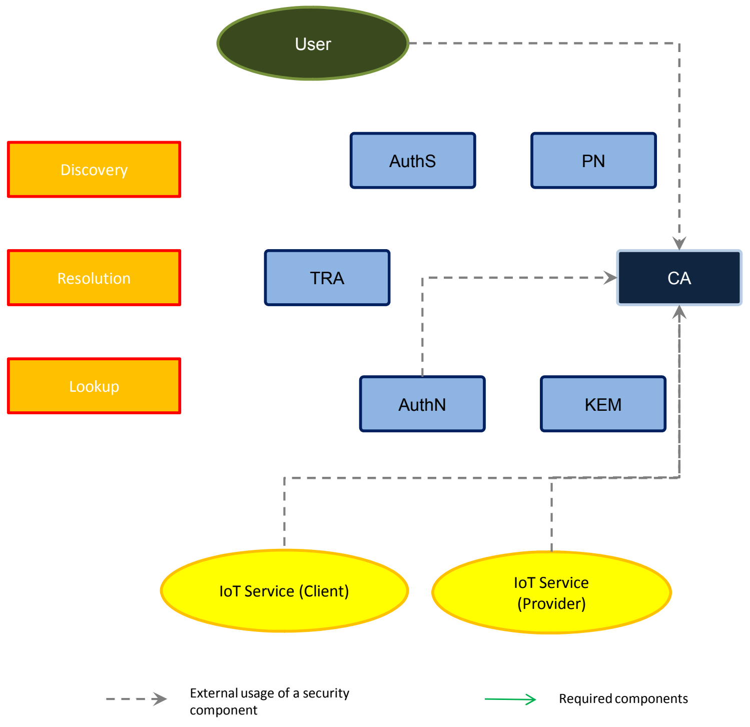

Figure 26: Usage of the certification authority. The certification authority does not depend on other components.

4. Summary and outlook

In this public deliverable we presented an initial architectural reference model for the IoT. It is apparent that this work is a comprehensive task that comes with many potential pitfalls. This risk mandates a structured approach to this architecture exercise. With the vision in Section 1, our structured approach is formulated and motivated. This defined architecture process makes the modelling steps is traceable and in turn can be used in future iterations on the architectural reference model.

The two other main contributions in this document are the IoT reference model in Section 2 and the IoT reference architecture in Section 3.

With the IoT reference model, an abstract understanding of the IoT domain is achieved, providing a discourse into challenges, domain model, information model, communication model.

The IoT reference architecture provides a summary of the derived views and perspective and provides details on the functional view and the security and privacy perspective.

The depth of description of models included in this document varies due to different starting points of the modelling tasks. For instance, the domain model has been worked on and refined since the start of the project and it is thus already reaching a certain maturity level. In contrast, the information and communication model is only in an initial state and needs to be extended further. The aim for future releases of this architectural reference model is to refine models to the extend that they reach the same maturity level. Furthermore, validation of the models has to be achieved. Section 4.1 provides a roadmap for this effort, with detailed targets for all model updates.

It should also be mentioned that a preview of this deliverable was provided during the IoT Week event Barcelona in June 2011.[6] During this event, many interested parties from industry, standardisation organisations, and academia showed their interest in our work.

Publications on our overall approach and the models described here are planned. A major goal of such publication is to generate the impact necessary for our work becoming be a major contribution towards a prosperous Internet of Things.

[6] See http://www.iot-week.eu/

4.1 Roadmap toward D1.3

This deliverable is only a first step forward towards a stable and reviewed architectural reference model. In this Section we outline what parts will be added to this deliverable during the time ahead.

What?	Description
Unified requirements	New input from stakeholders will be provided in the future, and this input will again be translated into unified requirements according to the same process as was used for in D6.1 [Pastor, 2010].
Internal requirements	The list of internal requirements inferred from D1.1 [Bui, 2011] will be finalised and the same implies for internal requirements that are provided by project partners (see Annex A.2). In the meanwhile, we will also continually monitor the pertinent literature and ad new requirements to this list whenever needed.
Update of the functional view	The above new requirements will be used to update the functional view (see Section 3.1), and the consistency of the current functional view will be checked against these new requirements. In case new functional components will be identified, they will of course be added to the functional view. Also, in light of these new requirements, it will be checked, whether the location of a functional component in a specific functionality group has to be reconsidered.
Views not covered yet	In this document only the functional view has been covered. The current information model (as part of the IoT reference model) only provides a first basis for the information view. According to the requirement process carried out by the pertinent IoT-A work package, two more views are of importance for IoT: deployment and operation. These three views will be addressed in the next version of this deliverable.
Perspectives	Currently, this document does not cover any architectural principle or quality aspects besides security and privacy. In the next version of the deliverable, we will also address already identified perspectives, viz. performance and scalability; availability and resilience; as well as evolution and interoperability.
Metainformation model	The information model presented in this deliverable has to be developed further. The steps to be undertaken are: • Collection of existing information models dealing with software architecture, networks, and communication. Examples are the Common Information Model (CIM) and the Shared Information & Data Model (SID). • Validation of collected information models for IoT. • Identification of gaps regarding the areas described in the IoT-A reference architecture and the IoT-A domain model. • Filling the gaps with new or adapted information models.

What?	Description
	• Building a metainformation model that covers all the different information models and links them to the IoT-A reference architecture and the IoT-A domain model.
Interaction sequences and interface definition	What information is exchanged through the interfaces identified in the functional view and what information models are used to describe this information?
Use cases	We will derive technical implications from the use cases defined by the stakeholders (see D6.1 [Pastor, 2010]) and the usage scenarios covered in the SoTA (see D1.1 [Bui, 2011]). These use cases will be used for enriching the already presented business scenarios, use cases, and the interface analysis (see above).
Best practices	We will provide a selection of best practices of how to generate compliant architectures from the reference architecture.
Challenges	We will provide an update to the definition of challenges definition with a focus on impact factors for the different challenges. Questions such as what is relevant in order to achieve scalability have to be elaborated further.

References

[AuotI, 2011]	“Autonomic Internet (an FP7 project)”, http://ist-autoi.eu/autoi/index.php# (accessed 2011-06-10), 2011
[ANSI, 2000]	ANSI/IEEE, "ANSI/IEEE 1471-2000 Standard for Systems and Software Engineering - Recommended Practice for Architectural Description of Software-Intensive Systems," ANSI/IEEE, 2000
[Bui, 2011]	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
[Cantor, 2005]	S. Cantor, J. Kemp, R. Philpott, E. Maler, “Assertions and Protocols for the OASIS Security Assertion Markup Language (SAML) V2.0”, OASIS, http://docs.oasis-open.org/security/saml/v2.0/saml-core-2.0-os.pdf (accessed 2011-06-15), 15 March 2005
[CCSDS, 2006]	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
[EPCGlobal, 2007]	EPCGlobal, “EPC Information Services (EPCIS) Version 1.0.1 Specification”, EPCglobal standard specification, September 2007
[Field, 2008]	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
[Giffinger, 2007]	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
[Haller, 2010]	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
[IoT-A, 2011]	Internet-of-Things Architecture, “Terminology”, http://www.iot-a.eu/public/terminology (accessed 2011-06-14), 2011
[IoT-I, 2011]	IoT-I, ”Survey on IoT scenarios”, private communication, 2011.
[MacKenzie, 2006]	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

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

[Nicholson, 2010] 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

[OGS, 2002] 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

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

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

[OWL2, 2009] OWL 2 Web Ontology Language Definition, http://www.w3.org/TR/owl2-overview/ (accessed 2011-06-14), 2009

[Pastor, 2010] 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

[Römer, 2002] 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

[Rozanski, 2005] 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

[Salinas, 2010] A. Salinas Segura, R. Kernchen, S. Meyer, J. Riedl, F. Lopez Aguilar, A. Bassi, “Project Deliverable D6.6 - Report on Stakeholder opinions”, 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

[Serbanati, 2011] 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

[Sclater, 2007] Sclater, N., “Mechanisms and Mechanical Devices Sourcebook”, 4th Edition (2007), 25, McGraw-Hill

[Shames, 2003] 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

[Sundmaeker, 2010] H. Sundmaeker, P. Guillemin, P. Friess, and S. Woelfflé (Eds.), "Vision and Challenges for Realising the Internet of Things. ISBN 9789279150883, 2010

[Tamblyn, 2007] 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

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

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

Annex A – 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.

The two sets of requirements used in developing the reference architecture are presented here for the reader's reference.

1. An initial set of requirements coming from the stakeholders.
2. A set of requirements which come from internal partners in the IoT-A project

Schematically, can be shown as:

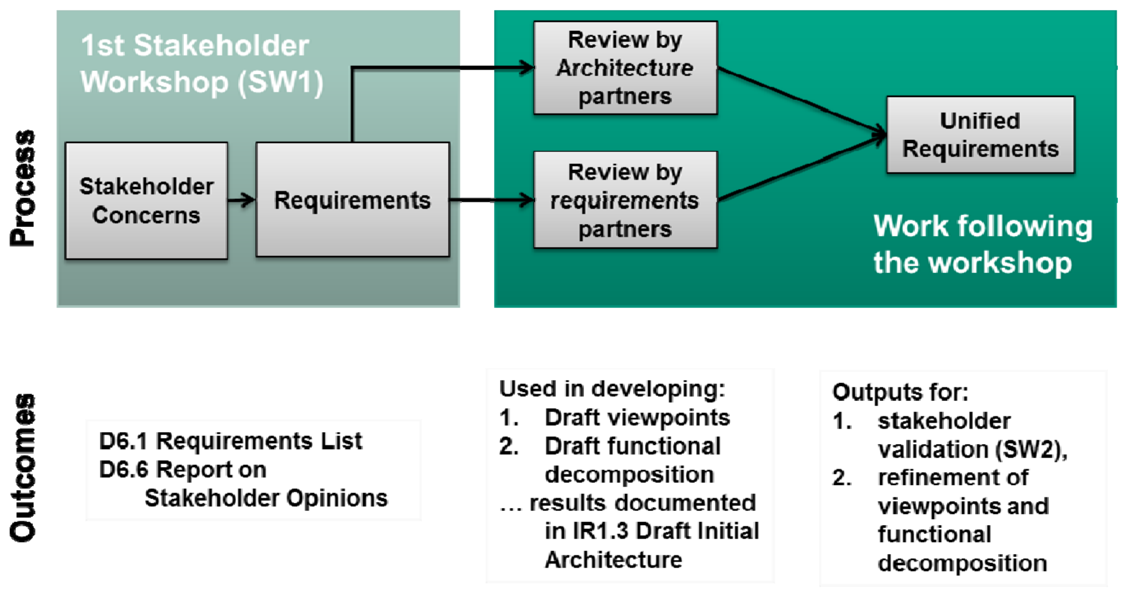
Process
1st Stakeholder Workshop (SW1)
Stakeholder Concerns
Requirements
Review by Architecture partners
Review by requirements partners
Unified Requirements
Work following the workshop
Outcomes
D6.1 Requirements List
D6.6 Report on Stakeholder Opinions
Used in developing:
1. Draft viewpoints
2. Draft functional decomposition
… results documented in IR1.3 Draft Initial Architecture
Outputs for:
1. stakeholder validation (SW2),
2. refinement of viewpoints and functional decomposition

Figure 27: Overall process by which requirements were developed, so that they could serve as inputs for the requirements to the Architecture Reference Model

A description of the requirements processes are as follows.

A.1 Requirements from stakeholders

The process began with collecting requirements from the 7 stakeholders during the first stakeholder workshop in Paris, October 2010. The members of the stakeholder group were representative of a wide range of business domains with an interest on Internet of Things: Logistics, Health Care, Technology Integration, Retail, Automotive, Service Integrators, Telecom Operators, Law, Standardization and Veterinary Medicine.

The requirements 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 in the Draft Initial Architecture, IR1.3.

The inputs of WP1 and WP6 were then combined, so that a unified set of requirements were obtained (as shown in Figure 27). These resulting sets of requirements were then used to refine the views and functional decomposition as found in this document, D1.2 Initial Architecture Reference Model.

For the reader's reference, the process diagram describing how the unified stakeholder requirements led to the architecture reference model is presented as follows:

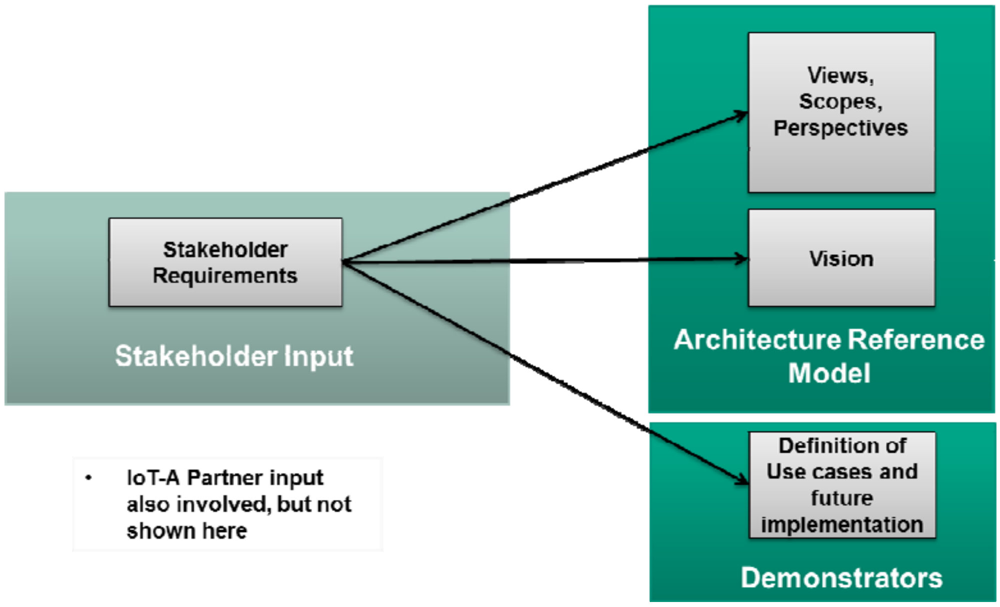
Views, Scopes, Perspectives
Vision
Architecture Reference Model
Stakeholder Requirements
Stakeholder Input
Definition of Use cases and future implementation
Demonstrators
• IoT-A Partner input also involved, but not shown here

Figure 28: The process by which stakeholder requirements were developed into inputs for developing the architecture reference model

For the reader reference, the unified requirement list and the corresponding view, perspective and relevant concepts from the Reference Model are presented here. Out of scope requirements have been excluded from the document.

ID	Unified Requirement	View	Perspective	Reference Model	Rationale (from stakeholder)	Requirement Type
UNI.1	The system shall provide a means to allow people to use Internet of Things Services anonymously		Security and Privacy	Human User, Service	Citizens want to protect their private data	Functional Requirement
UNI.2	Human users have control how their data is exposed to other users		Security and Privacy	Human User, Service, Resource	Citizens want to protect their private data	Functional Requirement
UNI.3	The system shall provide an device-interaction protocol	Functional		Device, Control Point, Gateway	I would like a way to create and exchange semantics between objects in order to design new applications	Functional Requirement
UNI.4	The system shall provide a model for describing Physical entities semantically	Information		Physical Entity, MetaData	I would like a way to create and exchange semantics between objects in order to design new applications	Functional Requirement
UNI.5	The system shall provide interfaces for accessing the semantical descriptions of entities	Functional		Active Digital Entity	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 Requirement

ID	Unified Requirement	View	Perspective	Reference Model	Rationale (from stakeholder)	Requirement Type
UNI.6	The system shall propose means to design applications taking into account the semantical decriptions of Devices/Physical entities	Information		MetaData, Service	I would like a way to create and exchange semantics between objects in order to design new applications	Functional Requirement
UNI.8	The system shall be able to run Applications and Services concurrently		Performance and Scalability	Active Digital Entity, Service	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	Functional Requirement
UNI.10	The system shall enable autonomous goal-driven (task-driven) collaboration between Devices or Services	Operational		Device, Service	I would expect that the traffic lights collaborate for a goal	Functional Requirement
UNI.12	The system shall be able to handle interference between IoT Devices (avoidance and detection)	Deployment		Device	In order to achieve a reliable eHealth service the system must be interference-free	Functional Requirement
UNI.14	The system shall support Devices to activate themselves into a collaboration	Operational		Device, Service	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	Functional Requirement
UNI.15	Devices shall have the possibility to be remotely controlled and configured	Operational		Device, User, Service	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	Functional Requirement
UNI.16	The system shall support Physical entity location tracking (geo spatial and/or logical location)	Information		Physical Entity, Service	High value assets need to be tracked in order to avoid theft and also to know where they are currently located	Functional Requirement
UNI.17	The system shall support event-based, periodic, and/or autonomous communication between devices	Functional		Data Sink, Control Point, Gateway	Citizens want to use features of smart products	Functional Requirement
UNI.18	The system shall support data processing (filtering, aggregation/fusion, ...) on different IoT-system levels (for instance device level)	Information		Active Digital Entity, Data Processor	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 Requirement

ID	Unified Requirement	View	Perspective	Reference Model	Rationale (from stakeholder)	Requirement Type
UNI.19	The system shall support provider-based Device management	Deployment		Device	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 Requirement
UNI.20	The system shall support the real-time monitoring of the radio usage of Devices and gateways	Operational		Device, Gateway	The application knows the current radio transmission activity of the M2M device	Functional Requirement
UNI.21	The system shall support the management of the radio transmitting Devices in real-time	Operational		Device	The application can control the radio transmission	Functional Requirement
UNI.22	The system shall support secure communications through secure messaging tool		Security and Privacy	Resource, Service	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 Requirement
UNI.23	The system shall provide access to external information sources, e.g. health databases		Evolution and Interoperability	Resource, Storage	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 Requirement
UNI.24	The system shall provide secure communication, e.g. for health information		Security and Privacy	Service, Resource, Device	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 Requirement
UNI.26	The system shall support time critical message handling and delivery		Performance and Scalability	Service, Resource, Device	In case of emergency the RMD has to send or receive time critical messages	Functional Requirement
UNI.27	The system shall support priorization of Services		Performance and Scalability	Service	In case of time-sensitive services the system needs to assure that important services are prioritized	Functional Requirement
UNI.28	The system shall support some mechanism of messages priorization		Performance and Scalability	Service, Resource, Device	Not every message has the same priority	Functional Requirement
UNI.29	The system shall provide a support for routing of data based on content	Functional		Service, Resource, Control Point, Gateway	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 Requirement

ID	Unified Requirement	View	Perspective	Reference Model	Rationale (from stakeholder)	Requirement Type
UNI.30	The system shall provide a resolution infrastructure for naming, addressing and assignment of Virtual entities and Services	Functional		Virtual Entity, Service	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 Requirement
UNI.31	The system shall provide functionality that allows the specification of business processes that autonomously monitor information related to Physical entities and controls the respective aspects of the Physical entity	Functional		Physical Entity, Service	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.	Design constraint
UNI.32	The system shall provide means for IoT-entities to react autonomously on context data (e.g. by using a rule language)	Functional		Sensor, Data Processor, Data Sink, Application Node	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.	Design constraint
UNI.36	The system shall provide means for linking entity specific user data of many users to one Physical entity	Information		Physical Entity, MetaData, User, Service	My wish is to retrieve the capacity of a thing. Thus, I can plan a change maintenance of all my bulbs if they can said when they should be changed	Functional Requirement
UNI.40	The system shall provide technical ways to ensure security and resilience		Availability and Resilience		Road users and energy providers want to avoid shortages/ blackouts	Non-functional Requirement
UNI.41	The system shall provide a historical information of the Physical entity monitoring	Information		Physical Entity, Storage, MetaData	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.	Functional Requirement
UNI.42	The system shall inform the User about its status and vice versa	Information		User, MetaData	Both the M2M server and the M2M device must be able to provide information about the current state	Functional Requirement

ID	Unified Requirement	View	Perspective	Reference Model	Rationale (from stakeholder)	Requirement Type
UNI.43	The system shall enable the composition of Augmented entity-related Services on devices and cloud services	Functional		Augmented Entity, Service, Device, Active Digital Entity	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 indgredient 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 Requirement
UNI.45	The system shall provide interfaces in order to allow the access using Mobile Devices	Functional		Service, Device	The mobile phone of the consumer can and should be used for interacting with product centric services	Functional Requirement
UNI.46	The system shall support user profiling	Information		Service, Storage, MetaData, User	The mobile phone of the consumer can and should be used for assisting the user in all purchase relevant aspects	Functional Requirement
UNI.47	The system must enable interoperability between Devices/Resources/Services and Applications		Evolution and Interoperability	Device, Resource, Service, Active Digital Entity	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.	Non-functional Requirement
UNI.48	The system shall provide an interoperable solution at the naming and addressing level		Evolution and Interoperability	Virtual Entity	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 Requirement
UNI.49	The system shall provide interfaces with legacy systems		Evolution and Interoperability	Service	Citizens doesn't want to use several city systems	Functional Requirement
UNI.50	The system shall provide mobility at the networking level	Operational			The use of M2M Devices for monitoring health related information is not confined to the residence of the patient.	Functional Requirement
UNI.51	The system shall support mobility of Devices/Services/Physical entities	Functional			Citizens want to access all areas of a city	Functional Requirement
UNI.56	The system shall support an energy aware architecture	Functional			Road users and energy providers want to avoid shortages/ blackouts	Functional Requirement
UNI.58	The system shall provide high reliability and low latency communications		Performance and Scalability		Communication blackouts are not accepted from client side and particularly if they are paying for premium services	Non-functional Requirement
UNI.59	The system shall provide different types of Services with different QoS associated to them		Availability and Resilience	Service	Communication blackouts are not accepted from client side and particularly if they are paying for premium services	Non-functional Requirement
UNI.60	The system shall provide different SLA		Performance and Scalability	Service	Communication blackouts are not accepted from client side and particularly if they are paying for premium services	Non-functional Requirement

ID	Unified Requirement	View	Perspective	Reference Model	Rationale (from stakeholder)	Requirement Type
UNI.62	The system shall provide highly trusted and secure communications and information management		Security and Privacy		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.	Design constraint
UNI.64	The system shall provide replanning of service execution		Availability and Resilience	Service	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.	Non-functional Requirement
UNI.65	The system shall be fault-tolerant and support always-on Services		Availability and Resilience	Service	Citizens want to use a reliable service	Functional Requirement
UNI.66	The system shall provide integrity validation of Virtual entities, Services and Platforms		Security and Privacy	Virtual Entity, Service	In certain life-critical applications the device may be required to perform a secure start-up procedure that includes integrity checking.	Functional Requirement
UNI.67	The system shall provide different access permissions to the information		Security and Privacy	MetaData	Sensitive data of patients must be kept secure in order to assure trust between the patients and to allow access to certain people	Functional Requirement
UNI.70	The system shall handle semantic interoperability between different semantical levels		Evolution and Interoperability	Service, MetaData	I would like a way to create and exchange semantics between objects in order to design new applications	Functional Requirement
UNI.71	The system shall provide standard communication between Augmented entity-related Services		Evolution and Interoperability	Augmented Entity, Service	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 standardisation of MIB (Management Information Base) of telecommunication equipments).	Design constraint
UNI.73	The system shall allow the semantic description of Physical entitys and Services by a user	Information		Physical Entity, Service, MetaData, User	I would like a way to create and exchange semantics between objects in order to design new applications	Functional Requirement
UNI.74	The system shall make comprehensive semantic information about Physical entities and services accessable to Human users and Active digital entities	Information		Physical Entity, Service, MetaData, User, Active Digital Entity	I would like to understand the semantics brought by the objects	Functional Requirement
UNI.87	The system shall support Service lifecycle management	Operational		Service, Resource, Storage	Road users want to use one service over a service life cycle	Functional Requirement

ID	Unified Requirement	View	Perspective	Reference Model	Rationale (from stakeholder)	Requirement Type
UNI.88	The system shall provide alarm signalling to indicate initialization failure on Services and Platforms		Availability and Resilience	Service, Resource	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 standardisation of MIB (Management Information Base) of telecommunication equipments).	Design constraint
UNI.89	The system shall support secure time synchronization		Availability and Resilience		Services which depend on a precise time need a guarantee that the devices they are communicating to have the right time.	Functional Requirement

A.2 Requirements from Internal Partners

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.

The approach taken was to ask each work package (which corresponded to the areas of orchestration, communication, discover and devices) 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 reference model.

Additionally upon completion of the system use cases (see Annex B), each work package was requested to extract the requirements for certain functionalities which an IoT system should have.

The reader should be aware that for this deliverable, the requirements from the internal partners are work in progress, and a complete list can be found in the subsequent D6.2 Updated Requirements List.

ID	Type	Priority	Description	Rationale	Fit Criterion
IR2.1	Functional Requirement	High	The process editor must be able to create BPMN 2.0.D25	BPMN 2.0 was evaluated to be the most IoT-aware process notation.	A BPMN 2.0 file is created by the editor.
IR2.2	Functional Requirement	High	The process editor must be extendable.	The reuse of a comprehensive tool allows to focus the effort.	New capabilities can be added to the process editor.
IR2.3	Functional Requirement	Medium	The process editor must provide facilities to model on business level.	A business user is not able to specify an executable process model.	The editor provides a special business view on the process, which excludes some execution details.

ID	Type	Priority	Description	Rationale	Fit Criterion
IR2.4	Functional Requirement	Medium	The process editor must provide facilities to model on technical level.	A technical user is not able to specify the business frame of a processes.	The editor provides a special technical view on the process, which enables to specify all execution details.
IR2.5	Non-Functional Requirement	Low	The process editor has to be enduser-friendly.	A business user needs to be able to model a process.	A person with no process execution background is able to model a process.
IR2.6	Functional Requirement	Low	The process editor must be able to verify the syntax of the process model.	The technical user needs information about the correctness of the syntax before the execution.	The editor provides a syntax checking functionality.
IR2.7	Non-Functional Requirement	Medium	The process editor must be "easily and fastly" extendable.	First project results should be presentable in a small time frame.	Small effort needed for the implementation of a new stencil.
IR2.8	Non-Functional Requirement	Medium	The process editor has to provide an attractive graphical user interface.	The project results need to be representable in a research review.	A user looking at the editor for the first time must say: "Wow, that's cool!"
IR2.9	Functional Requirement	Medium	The process editor must be interoperable with developments of other WPs and Tasks.	The projects results should be combinable to reach the common project goals.	The process editor uses the interfaces, commonly defined with the other WPs, where necessary.
IR2.10	Design Constraint	Medium	The process editor must support BPMN 2.0 completely (in particular the IoT-aware parts)	The development effort should focus on the BPMN IoT extension.	An IoT-aware sample process is completely representable.
IR2.11	Functional Requirement	High	The process modeling notation has to be extensible in terms of the definition of new stencils, 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.	The notation allows extensions by default or the notation was already extended in the past.
IR2.12	Functional Requirement	Medium	The process modeling notation has to be executable.	The projects task 2.2 and 2.3 should closely work together and represent a hand in hand solution.	Execution Semantics for the artifacts of the modeling notation are defined.
IR2.13	Non-Functional Requirement	High	The process modeling notation has to be IoT-aware.	Due to the DOW the project focuses on IoT processes.	.
IR2.14	Functional Requirement	Medium	The process modeling notation has to offer a graphical representation.	A graphical process notation offers a symbolism to easily model and document business processes.	A symbolism is available for the notation.

ID	Type	Priority	Description	Rationale	Fit Criterion
IR2.15	Non-Functional Requirement	High	The process modeling notation has to be a standard.	A common standard maximizes the potential application of industrial stakeholders.	The standard implementation is published and administrated by the corresponding organisation.
IR2.16	Functional Requirement	High	The BPMN extension must support an entity based approach defined by the domain model of WP1.	The domain model is one key result by WP1 and should fit to the business modeling approach of WP2.	All relevant domain model concepts are reflected by the process modelling approach.
IR2.17	Functional Requirement	High	The BPMN extension must support the process execution distributed over several devices.	In the IoT the execution of process steps can be distributed over several devices.	An example process can be executed over more than one agent based system including several devices.
IR2.18	Functional Requirement	High	The BPMN extension must support the modelling of different IoT specific interaction types.	The interaction between different devices, the integration of information about physical entities, and the interaction between services characterizes the IoT.	IoT specific interaction types are definable.
IR2.19	Functional Requirement	Low	The BPMN extension must support to arrange data distribution over several data storages (resources) of devices.	Business Processes in the IoT distribute data objects in resources of many devices.	For each data object and data storage the resource is definable.
IR2.20	Functional Requirement	Low	The BPMN extension must provide means to scalably model and execute processes independently of the number of involved process components.	In IoT processes multiple physical entities, devices, resources and services can appear, which could negatively effect the performance of the execution.	For each process model indicators are available, that allow to predict the scalability of the process.
IR2.21	Functional Requirement	Low	The BPMN extension must support the abstraction of individual process components.	In the IoT multiple devices, resources and services can appear. The accuracy and availability of accumulated data can be of much higher importance for the process than the data of individual components. The extension shall provide abstractive individual process components.	Individual process components are abstractable.
IR2.22	Functional Requirement	Medium	The BPMN extension must support means to express the availability of a process component.	Due to the mobile nature that physical entities, devices and its services and data often have, a business process can have a different availability depending on its involved components.	An indicator of the availability of individual process components is available.

ID	Type	Priority	Description	Rationale	Fit Criterion
IR2.23	Functional Requirement	Medium	The BPMN extension must provide means to express the tolerable error rate of a process.	Depending on the process, a process result is still acceptable as far it stays under a tolerable error rate.	Defective business processes can be modeled and executed (not exceeding a certain error threshold)
IR2.24	Functional Requirement	Medium	The BPMN extension must provide means for designing context-aware business processes.	Depending on occurring events the IoT processes need to be highly flexible.	Several events types are representable using the BPMN extension.
IR2.25	Functional Requirement	Low	The BPMN extension must provide means for expressing the uncertainty of process components.	The uncertainty of individual process components can influence the process creation on model and execution time.	The uncertainty of different process components can be indicated.
IR2.26	Functional Requirement	High	The BPMN extension must provide means for expressing real-time constraints.	As the process interact with augmented entities real-time constraints apply to these processes	Different real-time constraints can be expressed.
IR2.27	Functional Requirement	High	The process execution engine must be able to execute processes described in BPMN 2.0 format.	The graphically defined BPMN 2.0 process model can be executed without mapping the process model to another notation.	The process engine executes a BPMN 2.0 process without pre-processing.
IR2.28	Functional Requirement	High	The process execution engine must be able to execute defined BPMN 2.0 extensions.	The execution demonstrates the benefit of the graphical extension.	The process engine executes a BPMN 2.0 process with extensions without errors.
IR2.29	Non-Functional Requirement	High	The process execution engine must be "easily and fastly" extendable.	The development should focus on the IoT related extension.	Small effort to implement an example extension to the process execution engine.
IR2.30	Functional Requirement	Medium	The process execution engine must be interoperable with the results and development of the other WP task.	The projects results should be combinable to reach the common project goals.	The process execution uses the interfaces, commonly defined with the other WPs, where necessary.
IR2.31	Functional Requirement	Medium	The process execution engine must support BPMN 2.0 completely.	The development effort should focus on the BPMN IoT extension.	A process using all BPMN 2.0 artefacts is executable.
IR2.32	Functional Requirement	Low	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.	One process execution engine is used in task 2.2 as well as in task 2.4

ID	Type	Priority	Description	Rationale	Fit Criterion
IR2.33	Functional Requirement	High	Mobile entities must 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	A mobile entity moves to a different network and/or administrative domain, the mobile entity is identified as the same, and events are still received
IR2.34	Non-Functional Requirement	Medium	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	Implemented distributed event processing component
IR2.35	Functional Requirement	Medium	Processing of events must take quality of information (QoI) into account	In the processing step quality changes	Implementation of a QoI aware event processing algorithm
IR2.36	Functional Requirement	Medium	Quality of information related to virtual entities can be retrieved from the system	Different devices provide information with varying quality. An application may have certain quality requirements.	Quality of information related to virtual entities can be retrieved from the system
IR2.37	Functional Requirement	High	The IoT-A reference architecture shall provide events that can be related to augmented entities	Augmented entities are the key concepts in IoT-A with which the applications will deal with.	Design of an event framework that satisfies this requirement.
IR2.38	Functional Requirement	Medium	The IoT-A reference architecture shall provide event templates that can be related to types of augmented entities	Events can be defined for a class of augmented entities at design time, but evaluated for every augmented entities of the same type at runtime. Otherwise Events must be defined for every particular augmented entity.	Events can be defined per Entity Type
IR2.39	Functional Requirement	High	The IoT-A architecture shall provide a shared memory of the observable phenomenon	Due to services could not be online all the time it could be necessary to incorporate a shared memory in order to store this information.	Implement a shared memory to store the measurement information
IR2.40	Functional Requirement	High	The IoT-A architecture 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	Definition and description of service interfaces
IR2.41	Functional Requirement	High	The IoT-A architecture 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; this will be also related to requirement IR2.39	Definition and description of service interfaces based on existing standards (e.g. OGC SWE) and SENSEI information models and interfaces
IR2.42	Functional Requirement	Medium	The IoT-A architecture shall provide standard query end-points and generic reasoning mechanisms to infer the emerging data and to process the stored meta-data related to resources/entities	This will provide generic interface to query the stored meta-data and to enable high-level applications/services to perform query and reasoning upon the existing/emerging data	Existing technologies provided by the Semantic Web community to provide query and reasoning mechanisms are employed by the meta data models designed in WP2

ID	Type	Priority	Description	Rationale	Fit Criterion
IR2.43	Functional Requirement	Medium	The IoT-A architecture shall provide mechanisms to publish and present the resource/entity/service description meta data as linked-data	This will enable linking the published description to other domain knowledge and also location models described by third party ontologies or open linked data concepts and will also support reasoning the data based on high-level concepts and entities defined in domain ontologies	Meta data models as well as the semantic data designed in WP2 are provided as linked-data and define association attributes for the designed models to relate them to domain and location data that can be provided by existing ontologies and/or open linked data resources.
IR2.44	Functional Requirement	High	The orchestration engine shall interpret service descriptions	service orchestration is done based on service descriptions	Correct service compositions can be created
IR2.45	Functional Requirement	High	The orchestration engine shall support creation of new applications	Higher level services should create new functionality	Higher level service is created based on lower level services
IR2.46	Functional Requirement	High	The orchestration engine shall create new service descriptions	The newly created service must be registered with service discovery	Valid service descriptions registered to be discovered correctly
IR2.47	Functional Requirement	High	The orchestration engine shall support flexible composition	Services involved in compositions can fail and need to be replaced by some serving equal needs	Services are replaced by similar ones in case of error
IR2.48	Functional Requirement	High	The orchestration engine shall handle scopes for selecting services for composition	Scopes selected for composed service must be applied to the atomic services as well	Scopes are applied correctly to all services a composition contains
IR2.49	Functional Requirement	Low	The orchestration engine shall increase quality of information by service composition	QoI can be increased by using additional information as reference	QoI is increased
IR2.50	Functional Requirement	High	The orchestration shall access service resolution	Orchestration depends on service descriptions provided by discovery	Service resolution can be accessed
IR2.51	Functional Requirement	High	The orchestration shall provide a feedback to the user who sent a composition request	The feedback should contain a message about the success of the requested composition	Feedback is send in any case (success/failed)
IR2.52	Non-Functional Requirement	Medium	The orchestration engine shall provide feedback within a reasonable amount of time (<5sec)	A time out must be set for request/response loops	Every request/response loop finishes within the limit

ID	Type	Priority	Description	Rationale	Fit Criterion
IR2.53	Functional Requirement	Low	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	Preferred services are selected for composition
IR4.1	Functional Requirement	Medium	Discovery and lookup service of IoT systems shall allow the locating physical entities based on geographical parameters	Confirms our present plan of having some geographical representation. 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"	The architecture reference model incorporates geographical parameters in resolution service
IR4.2	Functional Requirement	Medium	A geographical location attribute shall exist for virtual entities	Confirms our present plan of having some geographical representation. Derived from SP requirement "A SmartProduct should be able to access the location information of other SmartProducts"	Reference model defines location as entity parameter
IR4.3	Functional Requirement	Medium	IoT-A 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."	Function is specified in reference model
IR4.4	Functional Requirement	Medium	IoT-A 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"	Function is specified in reference model
IR4.5	Functional Requirement	Low	The location model 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	Feature is specified in the reference model
IR4.6	Functional Requirement	Medium	The location model shall enable the implementation of the following queries: position 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 queries"	Location model is specified in reference model and the parameters are specified as necessary
IR4.7	Functional Requirement	High	The look-up service of IoT-A shall withhold or grant information depending on context such as application involved, requesting entity, and security permissions	Needed for fulfilling security requests of stakeholders. Derived from BRIDGE requirement: "A broad set of data from enterprise applications MAY be requested depending on context, industry, application, etc"	Feature/Best Practice is specified in the reference model

ID	Type	Priority	Description	Rationale	Fit Criterion
IR4.8	Functional Requirement	High	Services (and information providing services) connected with the IoT system can 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 and also in the IoT-A stakeholders. The BRIDGE requirement was "Data that companies are willing to provide to the Discovery Services are mainly URL addresses of databases / EPCIS repositories"	Feature/Best Practice is specified in the reference model
IR4.9	Functional Requirement	Medium	The Digital Entity History Storage should 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	Feature/Best Practice is specified in the reference model
IR4.10	Functional Requirement	High	The Digital Entity History Storage shall be restricted in who can call delete and update functions	The integrity and trust in the history storage block depends on how "unaltered" it is. The BRIDGE SoTA 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 to provide a journal functionality"	Feature/Best Practice is specified in the reference model
IR4.11	Functional Requirement	High	Clients requesting data via the Discovery/Lookup services shall be uniquely identifiable	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).	Prototypes of the function are specified with these parameters in the reference model
IR4.12	Functional Requirement	High	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)	Feature/Best Practice is specified in the reference model
IR4.13	Design Constraint	High	Access-control rights/ policies (set up by data owners) shall not be published publicly.	Access control policies themselves, if known, can give away information.	Feature/Best Practice is specified in the reference model
IR4.14	Functional Requirement	High	The IoT system must enable the dynamic discovery of relevant virtual entities and their related services based on respective specifications.	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.	A discovery function with the specification of the virtual entity and the specification of the required service as parameters
IR4.15	Functional Requirement	High	The IoT system must enable the dynamic discovery of relevant physical entities and their related services based on a geographical location scope.	Geographic location is one of the most important aspects for finding relevant physical entities. Spatial relations are of prime importance in the physical world.	A discovery function with the specification of the physical entity and the specification of the required service as parameters and a geographic location scope as parameters.

ID	Type	Priority	Description	Rationale	Fit Criterion
IR4.16	Functional Requirement	High	The IoT system must enable the lookup of service descriptions of specified services for an augmented entity with the augmented entity identifier as key for the lookup.	It is important to find the services related to an augmented entity that may provide information about it, allow to actuate the augmented entity or enable interaction with the augmented entity.	A lookup function providing service descriptions with the augmented entity identifier and a service specification as parameters.
IR4.17	Functional Requirement	High	The IoT system must 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.	A resolution function providing service locators with the service identifier as parameter.
IR4.18	Functional Requirement	High	The IoT system must be able to discover dynamic associations between an virtual entities and services related to the 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 associations must be discovered	Associations are dynamically discovered and added to the Virtual Entity Resolution
IR4.19	Functional Requirement	High	The IoT system must be able to track dynamic associations between an augmented entity and services related to the augmented entity to determine whether they are still valid.	Due to the mobility of augmented entities as well as devices whose resources are accessible through services, changing services may provide information, allow actuation or enable interaction with augmented entities. In order to provide the currently relevant services for an augmented entity, the dynamic associations must be tracked to determine whether they are still valid.	Associations are tracked and automatically removed if it is determined that the association is no longer valid
IR4.20	Functional Requirement	High	The IoT system must be able to discover dynamic associations based on geographic location and other context information.	Mobility is one of the key aspects for changing associations. By monitoring the location of physical entities and area for which resources can provide information, possibly in combination with other context information, dynamic associations between physical entities and services providing access to resources can be discovered.	By using location services new dynamic associations can be found
IR4.21	Functional Requirement	High	The IoT system must be able to track dynamic associations between an virtual entity and services based on geographic location 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 possibly other aspects.	By using location services, it can be determined when dynamic associations become invalid.

ID	Type	Priority	Description	Rationale	Fit Criterion
IR4.22	Design Constraint	Medium	The IoT 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. To develop its full potential interactions, including lookup and discovery, across domain boundaries must be possible.	Associations from a different administrative domain can be looked up or discovered.
IR4.23	Design Constraint	High	The IoT system must respect the privacy aspects when performing discovery, resolution and lookup	Privacy is a key aspect for the IoT.	Pseudomized identifiers are unlinkable to other identifiers or a specific user
IR4.24	Design Constraint	Medium	The IoT 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	Users can access services in a pseudomized manner
IR4.25	Functional Requirement	High	The IoT Service Identifier shall use the service/resource description for retrieval	The IoT System must consider the description of a service/resource for the semantic indexing on which the search will be performed	A semantic description of the Resources and the ID of the associated virtual entity is recorded in what we define the Discovery Server
IR4.26	Functional Requirement	High	The IoT System shall be able to accept and manage semantic queries from the user and return Resources/Services	Necessary for the match in the VE Semantic Retrieval	Rough specifications are available in the definition of the Discovery Service
IR4.27	Functional Requirement	High	The Discovery Service in local search, is required to find service/resource based on (rough) semantic description	Because the discovery service in local search combine the peer to peer discovery with the white search (no semantic filter) in the geo-localization context.	Feature is not currently described in Reference Model; This can be done in the Terminology Section or functional Decomposition when talking about the Discovery Service
IR4.28	Functional Requirement	High	The IoT system shall have a service to obtain a new identifier to the new VE registered resource/service and to save the description of its services	VE Service Identifier manages the ID (VID) and the semantic description, for the Global Discovery Search.	Feature is specified in the reference model

ID	Type	Priority	Description	Rationale	Fit Criterion
IR4.29	Functional Requirement	High	The IoT system shall have a service to insert the operational specifications of the new registered resource/service	VE Service Specification manage the association ID(VID) to the operational specification for the LookUp Service	Feature is not specified explicitly in the Reference Model
IR4.30	Functional Requirement	High	The IoT system shall have a service to register the proper URI and the locator of the new registered resource/service	To managed by dynamic linker, uses for the Resolution Service by return the last address/locator	Feature is not specified explicitly in the Reference Model
IR4.31	Functional Requirement	High	A VE that is associated with a PE that changes geolocation shall update coordinates/address/locator through IoT system service	IoT Service Monitoring is a service that manages the coordinates/address/locator and uses for the Resolution Service by return the last address/locator	Feature is not specified explicitly in the Reference Model
IR4.32	Functional Requirement	Medium	IoT system should define a common virtual identification system (virtual-ID)	An 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, ...)	Feature/Best Practice is specified in the reference model
IR5.1	Non-Functional Requirement	High	The communicated messages must not be spied by an unauthorized person or device	Confidentiality must be ensured	
IR5.2	Non-Functional Requirement	High	The device (contactless card for example) must not be activated without the consent of the owner	To avoid unsolicited scanning of people	
IR5.3	Non-Functional Requirement	High	The identifier of the device (ID of an RFID tag for example) must not be tracked by unauthorized entities	The tracking of items and then people raise the problem of privacy	
IR5.4	Functional Requirement		Connected objects shall be able to do energy harvesting	Maintain operation in harsh environments	Ensure IOT-A exploitation potential in an as wide as possible spectrum of application domains
IR5.5	Functional Requirement		Connected objects shall be able to communicate with each other through the network via standard communication interfaces	Enhance wide use potential	It is part of the overall architecture
IR5.6	Non-Functional Requirement		Data security&privacy should be enabled at atomic level		Part of the wider security & privacy framework

ID	Type	Priority	Description	Rationale	Fit Criterion
IR5.7	Non-Functional Requirement		Communication with the objects must be intermittent and command-based	Avoid traffic overhead	Part of the WP3 requirements for M2M
IR5.8	Non-Functional Requirement		Each object should have a universal ID, part of it read-only and part of it read/write	Enable object recognition and setup/configuration in the context of particular applications development	Enable faster and easier setup of use-cases
IR5.9	Non-Functional Requirement		Object capabilities may be universally defined at HW-level	Enable plug n'play operations at user services level	Enable rapid object functions integration in user services
IR5.10	Non-Functional Requirement		Atomic-level protocols must implement only functions related to data acquisition (e.g. DSP-level), crypto and security	Avoid overlap with user-level communication protocols (WP3)	High level communication protocols are studied in WP3

Annex B – System use cases

In this Annex, the system use cases for the different functionality groups or functional components are described according to the functional view of Section 3.1.

At this stage of the project, it is difficult to include detailed system use cases of all functionality groups or functional components.
The use cases that are already available are presented in this Section because they allow a preliminary evaluation of the different architectural components and their interactions. Furthermore, they help the reader in getting a better understanding of the function and interaction of these components. They also help in identifying additional internal requirements as mentioned in Annex A.

The current plan is to expand further on this modelling in IR 1.4 and D1.3 and complement this annex with:

- System use cases covering the components not covered in this document.
- Diagrams showing the interaction between components such as sequence or interaction diagrams.
- Diagrams modelling the use cases of WP7.

The remainder of this Annex is organized as follows.
First, the system use cases of the process-execution and service-orchestration functionality group are described.
Next, IoT-services and resources and virtual-entity resolution use cases are described followed by virtual-entity and IoT service-monitoring use cases.
Finally, the system use cases of the security functionality group are provided.

B.1 Process execution and service orchestration

The use cases presented in this Section demonstrate two primary functional components, namely process execution and service composition & orchestration. The former functional component processes more highly refined data and pertains to business-process management. Business processes are modelled at the business level and executed in an environment in which services are resolved at design or runtime. These services fulfil the process steps or activities outlined in the process model. This is where the second functional component, namely service composition & orchestration comes into play, when services need to be found and orchestrated in order to execute business steps.

The use cases depicted below illustrate the mechanics of these two functions.

Use case 1: Process Execution.

The process-execution diagram in Figure 29 illustrates how a process model is created by a modelling application and then serialised and deployed to different execution platforms. The context management framework (CMF) and real-world integration platform (RWIP) are outlined as examples, as these execution platforms are used as background Intellectual Property in the IoT-A project.

The use cases addressing the 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 reference architecture, one of the technical work packages in IoT-A will develop such a tool. 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.

2. The graphical model is then serialised in an executable form. The preliminary analysis of process execution languages and notations (which will be discussed in depth in the forthcoming deliverable D2.2 due at month 18 of the project) indicates that BPMN2.0 will most probably be the preferred output format for IoT-aware processes. A technical expert will use this serialisation to deploy the process to an execution environment, in which the process is to be run.

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 CMF. In these platforms, service capabilities and activity requirements are aligned. The goal of this alignment is to allow the choice of services that are capable of providing the required IoT-specific service qualities. For instance, a process might require a certain confidence level provided by a sensor service of at least 80%, so that only a subset of the available sensor services might be suitable within the execution of the respective process parts. At this stage of execution, the service-composition & orchestration component becomes relevant, as certain quality and capability parameters might not be met by individual services, but only by an orchestration of such services. The lower part of the diagram is thus explained in more detail within Figure 30.

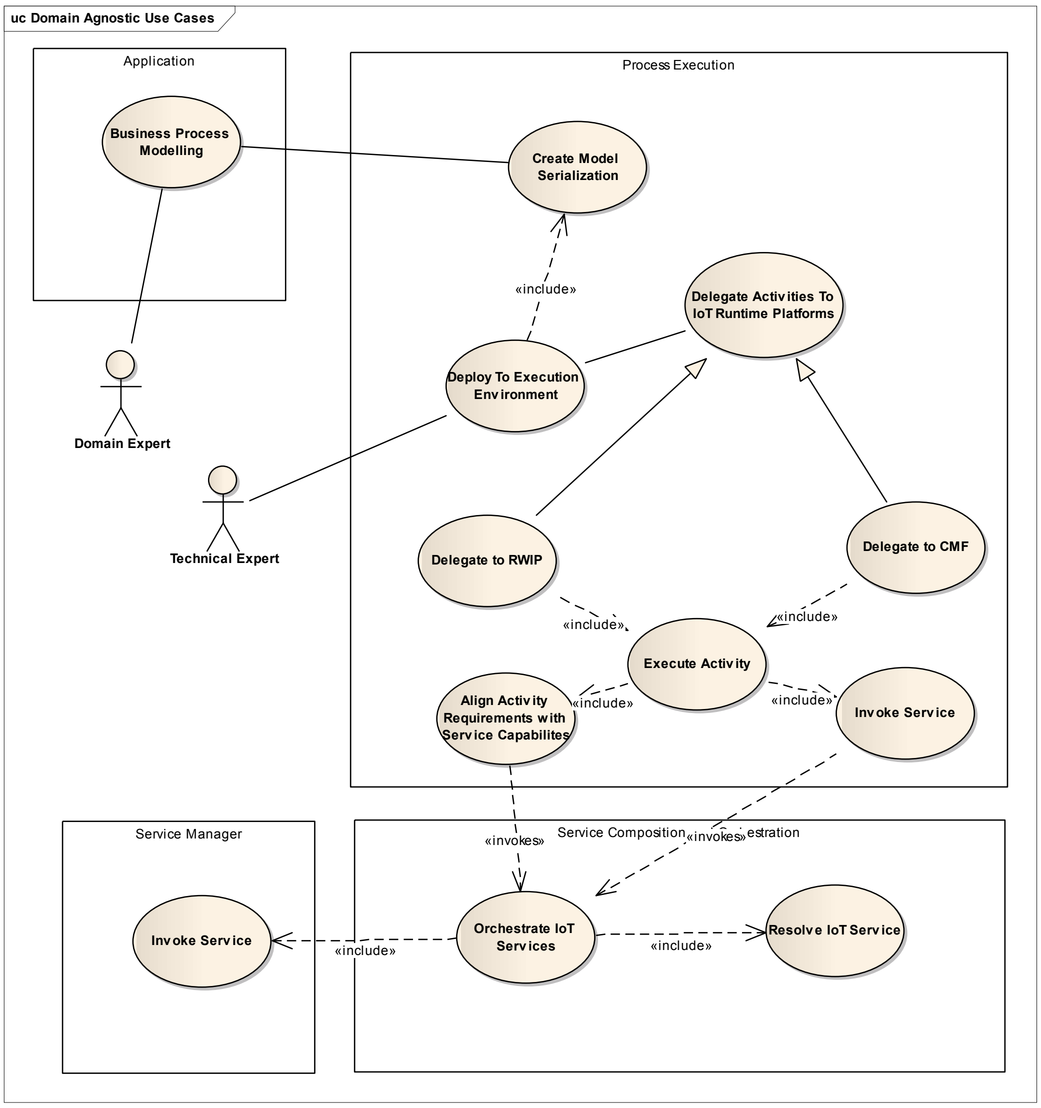

Figure 29: Use case process execution

Use case 2: Service composition and orchestration.

When individual process steps or activities need to be executed, the service-composition & orchestration diagram comes into focus. The following diagram shows the detailed and principal steps for service composition in a domain-agnostic way. The general principle is always that a mapping of services to VEs must be found by aligning information from the functionality group virtual entity resolution & IoT-service resolution.
As this chain of invocation typically progresses from the service-composition & orchestration component to the process-execution component (i.e. services are typically orchestrated in the execution of a process activity). Therefore, the process-execution component is shown as an actor starting the composition activities. While the diagram features the relationships to other functional components, we can disregard these for the time being and 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 precision for the information provided, fusing several similar services might increase information quality considerably, as errors are mitigated and faulty sensors excluded from the list of "providers".
- *Support flexible service composition*
  IoT services can be composed of other services, so that the composition of the individual services together might provide higher-level functionality. The composition is flexible because it is not made of pre-defined services but services able to meet the required functionality. If one service fails it can be replaced by a similar one. This behaviour is closely related to the previous aspect of an increase in information quality 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 process-execution component delegates service orchestration, viz. the execution of services appropriate for the specific process activity, to the service composition and orchestration function. It constitutes the interface between the process execution and the service-resolution infrastructure (the latter is discussed in the following Sections). In essence, the process-execution component conveys the service requirements needed for executing to the service-composition & orchestration component, which in turn utilises the IoT-service-resolution functionality in order to find and resolve appropriate services, and to create a suitable orchestration from them, if necessary.

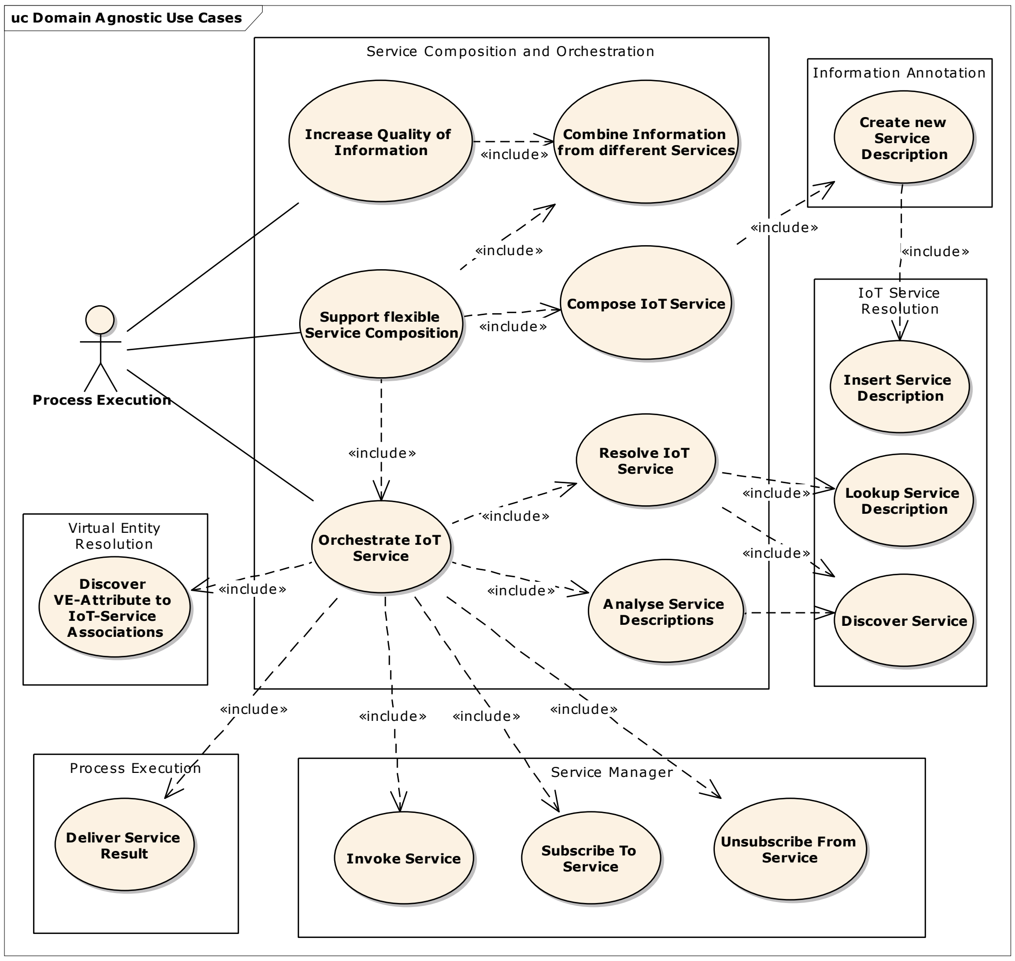
uc Domain Agnostic Use Cases
Service Composition and Orchestration
Increase Quality of Information
Combine Information from different Services
«include»
Support flexible Service Composition
Compose IoT Service
Process Execution
Resolve IoT Service
Orchestrate IoT Service
Analyse Service Descriptions
Information Annotation
Create new Service Description
IoT Service Resolution
Insert Service Description
Lookup Service Description
Discover Service
Virtual Entity Resolution
Discover VE-Attribute to IoT-Service Associations
Process Execution
Deliver Service Result
Service Manager
Invoke Service
Subscribe To Service
Unsubscribe From Service

Figure 30: Use case service composition and orchestration

B.2 Resolution of IoT service

The use cases of this Section cover the IoT Service Resolution functional component as identified in the functional view (see Section 3.1). 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.

The following use cases are depicted in Figure 31.

- *Resolve Service Identifier to URL/Address*
    - The use case is initiated by a user of the system, i.e., a human user or an active digital entity. 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.
- *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.
- *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 it 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.
- *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 that can be looked up or discovered, and/or reflect the status as reported by the services.

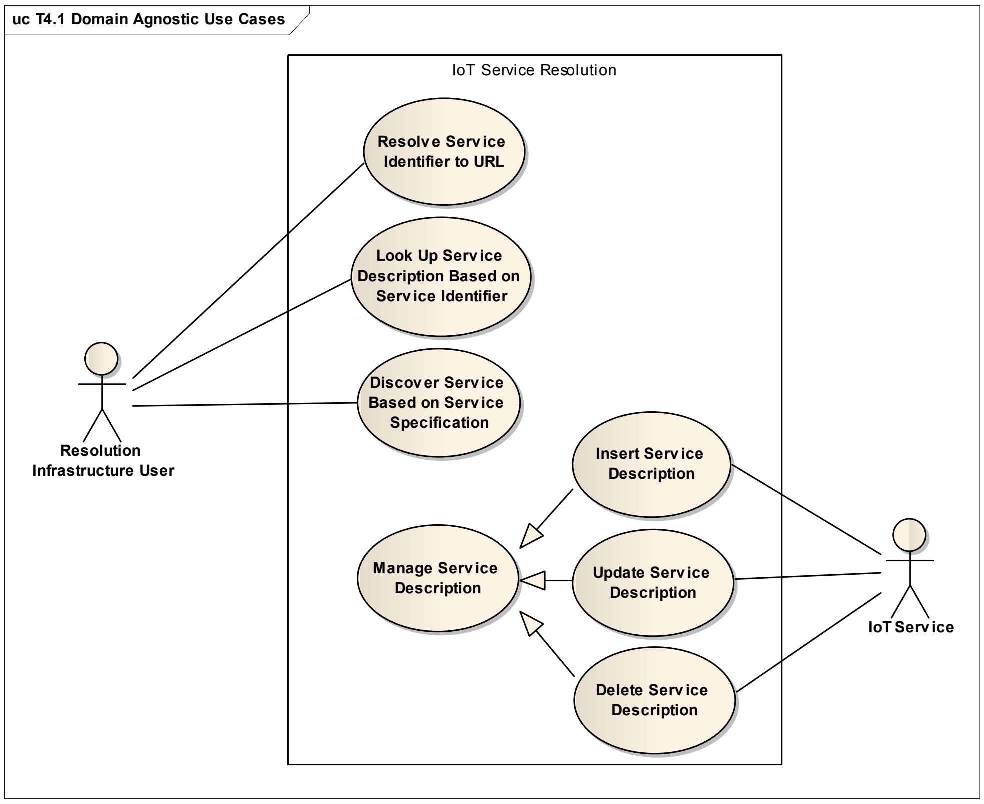
uc T4.1 Domain Agnostic Use Cases
IoT Service Resolution
Resolve Service Identifier to URL
Look Up Service Description Based on Service Identifier
Discover Service Based on Service Specification
Resolution Infrastructure User
Insert Service Description
Manage Service Description
Update Service Description
Delete Service Description
IoT Service

Figure 31:Use case IoT Service Resolution

B.3 Resolution of virtual entities

In this Section, the Virtual Entity Resolution functional component, as identified in the functional view (see Section 2.4), 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 32**.**

- 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 human user or an active digital entity like a software agent. The user wants to look up associations that associate the virtual identification 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 virtual identity of the virtual entity.
    - In this use case, the Virtual Entity Resolution looks up the associations corresponding to the virtual identification 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 identifiers of the required services whose description can then be looked up through the IoT Service Resolution.
- *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 now a 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 lookup 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.
- *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.

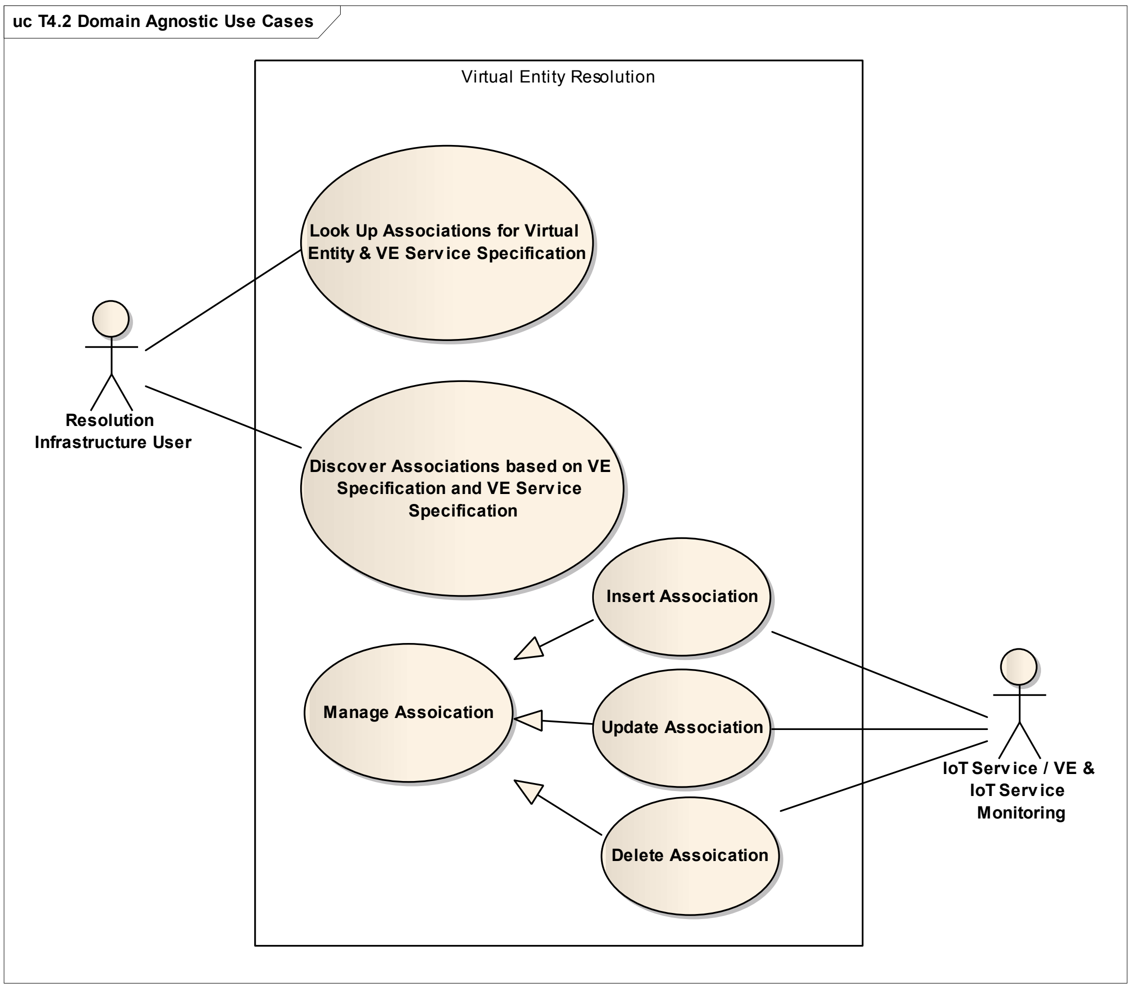
uc T4.2 Domain Agnostic Use Cases
Virtual Entity Resolution
Look Up Associations for Virtual Entity & VE Service Specification
Resolution Infrastructure User
Discover Associations based on VE Specification and VE Service Specification
Insert Association
Manage Assoication
Update Association
IoT Service / VE & IoT Service Monitoring
Delete Assoication

Figure 32: Use case Virtual Entity Resolution

B.4 Monitoring of virtual entities and IoT services

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.

Use cases depicted in Figure 33 cover the 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 changes 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.

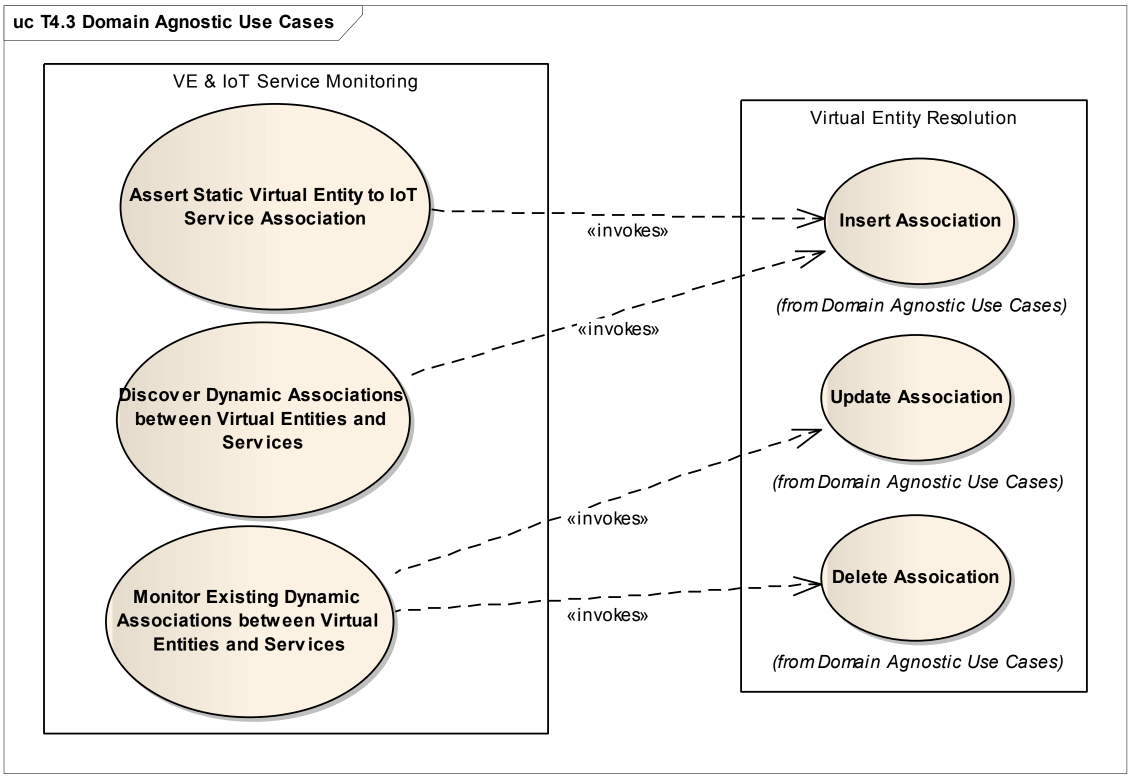
uc T4.3 Domain Agnostic Use Cases
VE & IoT Service Monitoring
Assert Static Virtual Entity to IoT Service Association
Discover Dynamic Associations between Virtual Entities and Services
Monitor Existing Dynamic Associations between Virtual Entities and Services
«invokes»
«invokes»
«invokes»
«invokes»
Virtual Entity Resolution
Insert Association
(from Domain Agnostic Use Cases)
Update Association
(from Domain Agnostic Use Cases)
Delete Assoication
(from Domain Agnostic Use Cases)

Figure 33: Use case Virtual Entity & IoT Service Monitoring.

B.5 Security

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 Annex (B.2, B.3, and B.4).

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 34 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
Figure 34

- *Authenticate the user*
  The user is authenticated and an assertion of his identity is provided[7].
- *Discover person-related IoT services for authorised personal*
  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* (see Section B.2 for details). 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 of the secure discovery of an IoT service, the user only receives those services that he is entitled to see due to privacy restrictions.

[7] As an example a SAML Authentication Assertion could be provided [Cantor, 2005]

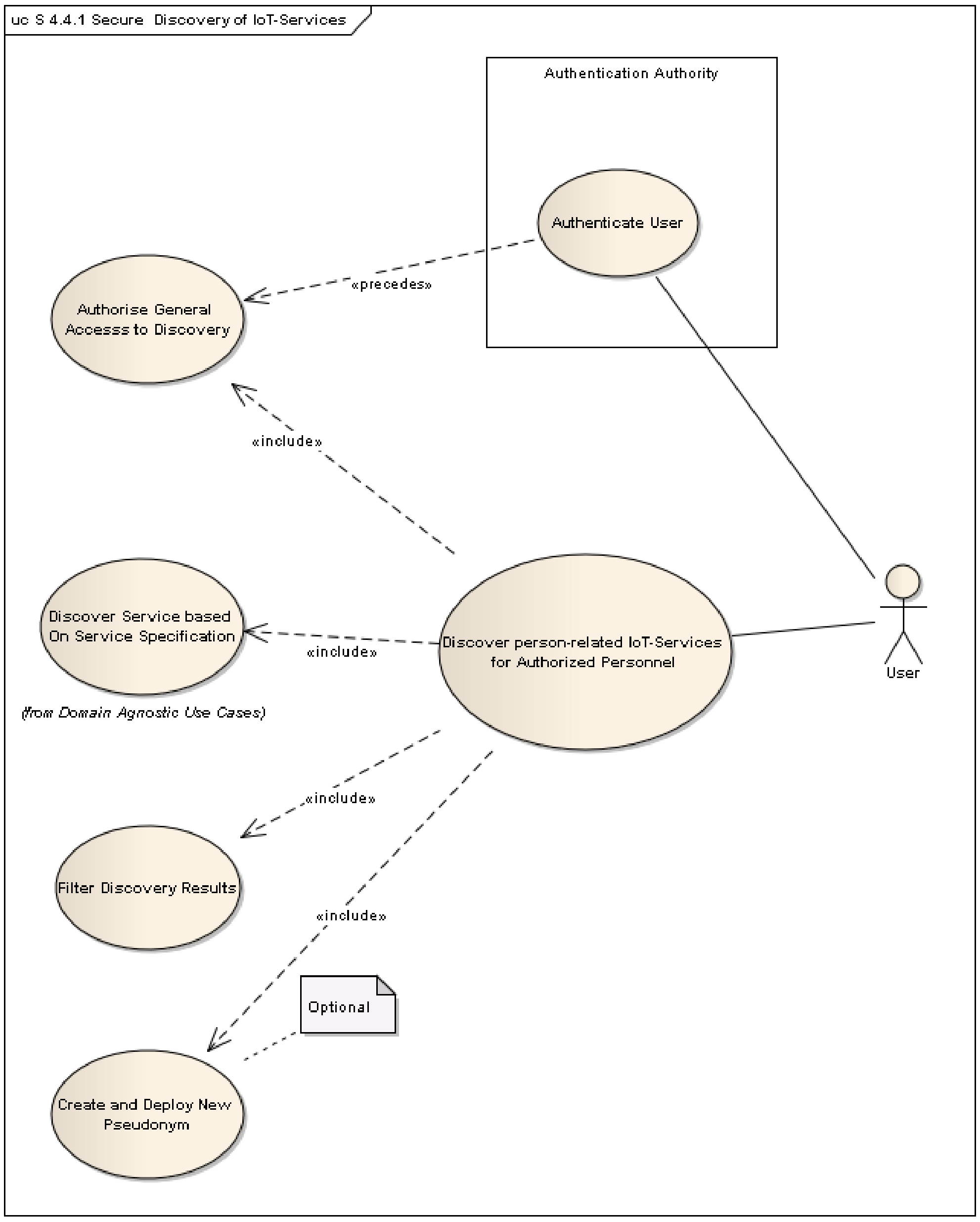
uc S 4.4.1 Secure Discovery of IoT-Services
Authentication Authority
Authenticate User
«precedes»
Authorise General Accesss to Discovery
«include»
Discover Service based On Service Specification
(from Domain Agnostic Use Cases)
«include»
Discover person-related IoT-Services for Authorized Personnel
User
«include»
Filter Discovery Results
«include»
Optional
Create and Deploy New Pseudonym

Figure 34: 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 prior to the discovery.

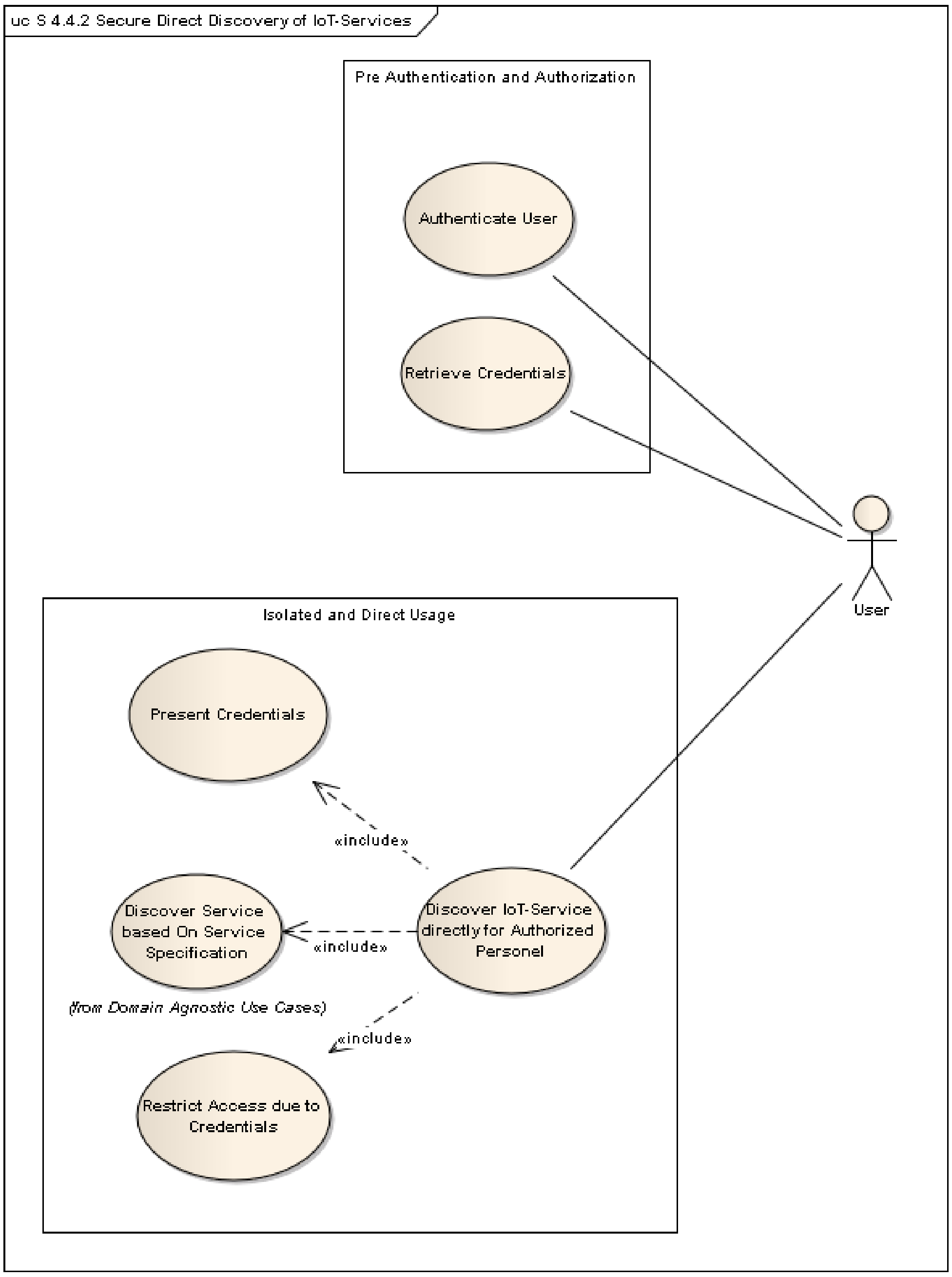
uc S 4.4.2 Secure Direct Discovery of IoT-Services
Pre Authentication and Authorization
Authenticate User
Retrieve Credentials
User
Isolated and Direct Usage
Present Credentials
«include»
Discover Service based On Service Specification
«include»
Discover IoT-Service directly for Authorized Personel
(from Domain Agnostic Use Cases)
«include»
Restrict Access due to Credentials

Figure 35: Secure Direct Discovery of IoT Services

The actor in the uses case shown in Figure 35 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.
  - *Discovery service based on service specification* (see section B.2 for details)
    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.