UNIVERSITÄT KOBLENZ · LANDAU

IWVI

Development of an Internet of Things architecture framework based on Sensing as a Service

A thesis submitted for the

Master of Science in Information Systems

by

Patrick Nitschke

Student ID: 209210074

E-Mail: nitschke@uni-koblenz.de

Faculty 4: Computer Science

Institute for IS Research

University of Koblenz-Landau, Germany

Supervisors:

Prof. Dr. Susan P. Williams

Prof. Dr. Petra Schubert

Koblenz, February 2017

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Declaration/ Erklärung

I declare that,

This thesis presents work carried out by myself and does not incorporate without acknowledgement any material previously submitted for a degree or diploma in any university. To the best of my knowledge, it does not constitute any previous work published or written by another person except where due reference is made in the text.

Ich versichere,

dass ich die vorliegende Arbeit selbständig verfasst und keine anderen als die angegebenen Quellen und Hilfsmittel benutzt habe.

Mit der Einstellung dieser Arbeit in die Bibliothek bin ich einverstanden. Der Veröffentlichung dieser Arbeit im Internet stimme ich zu.

Patrick Nitschke

Koblenz, March 2017

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Abstract (English)

The Internet of Things (IoT) is a network of addressable, physical objects that contain embedded sensing, communication and actuating technologies to sense and interact with their environment (Geschickter 2015). Like every novel paradigm, the IoT sparks interest throughout all domains both in theory and practice, resulting in the development of systems pushing technology to its limits. These limits become apparent when having to manage an increasing number of *Things* across various contexts. A plethora of IoT architecture proposals have been developed and prototype products, such as IoT platforms, been introduced. However, each of these architectures and products apply their very own interpretations of an IoT architecture and its individual components so that IoT is currently more an Intranet of Things than an Internet of Things (Zorzi et al. 2010). Thus, this thesis aims to develop a common understanding of the elements forming an IoT architecture and provide high-level specifications in the form of a *Holistic IoT Architecture Framework.*

Design Science Research (DSR) is used in this thesis to develop the architecture framework based on the pertinent literature. The development of the *Holistic IoT Architecture Framework* includes the identification of two new *IoT Architecture Perspectives* that became apparent during the analysis of the IoT architecture proposals identified in the extant literature. While applying these novel perspectives, the need for a new component for the architecture framework, which was merely implicitly mentioned in the literature, became obvious as well. The components of various IoT architecture proposals as well as the novel component, the *Thing Management System*, were combined, consolidated and related to each other to develop the *Holistic IoT Architecture Framework*. Subsequently, it was shown that the specifications of the architecture framework are suitable to guide the implementation of a prototype.

This contribution provides a common understanding of the basic building blocks, actors and relations of an IoT architecture.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Abstract (German)

Das Internet der Dinge (IoT) ist ein Netzwerk bestehend aus adressierbaren, physikalischen Objekten, die Sensor-, Kommunikations- und Aktuator-Technologien bereitstellen und mit ihrer Umwelt interagieren (Geschickter 2015). Wie jedes neue Konzept, hat auch IoT Interesse über jeden Anwendungsbereich hinweg, sowohl in Theorie als auch Praxis, geweckt und die verfügbaren Technologien an ihre Grenzen gebracht. Diese Grenzen machen sich insbesondere dann bemerkbar, wenn die Anzahl von Dingen (Things), die über verschiedenste Anwendungsbereiche hinweg verwaltet werden müssen, steigt. Um die neuartigen Anforderungen zu erfüllen, wurde eine Fülle von verschiedenen Systemen entwickelt, die alle ihre eigenen Interpretationen einer IoT Architektur und ihrer jeweiligen Komponenten anwenden. Dies hat dazu geführt, dass IoT aktuell eher ein Intranet der Dinge als ein Internet der Dinge ist (Zorzi et al. 2010). Daher ist es Ziel dieser Arbeit, ein einheitliches Verständnis der Komponenten, die eine IoT Architektur bilden, zu erlangen und generische Spezifikationen in Form eines *Ganzheitlichen IoT Architektur Frameworks* zur Verfügung zu stellen.

Diese Arbeit verwendet *Design Science Research* (DSR), um die genannte Architektur auf Basis der einschlägigen Literatur zu entwickeln. Die Entwicklung des *Ganzheitlichen IoT Architektur Frameworks* umfasst die Nutzung zwei neuer Perspektiven auf IoT Architekturen (*IoT Architecture Perspectives*), die während der Analyse von IoT Architekturen in der Literatur identifiziert wurden. Die Anwendung dieser neuen Perspektiven führte zur Erkenntnis, dass eine weitere, ebenfalls neuartige, Komponente in der Literatur implizit erwähnt wird. Die Beschreibungen der Komponenten von verschiedenen IoT Architekturen wurden vereinheitlicht und mit der neuen Komponente, dem *Thing Management System*, in Beziehung gesetzt, um das *Ganzheitliche IoT Architektur Framework* zu entwickeln. Weiterhin wurde gezeigt, dass die Spezifikationen der Architektur als Vorlage für die Implementation eines Prototypen geeignet ist.

Der Hauptbeitrag dieser Arbeit ist ein vereinheitlichtes Verständnis der einzelnen Komponenten sowie deren Interaktionen einer IoT Architektur.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Table of Contents

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

List of Abbreviations

IoT	Internet of Things
WoT	Web of Things
S2aaS	Sensing as a Service
PaaS	Platform as a Service
IS	Information System
DSR	Design Science Research
GDC	General Design Cycle
AGDC	Aggregated General Design Cycle
IDM	Identity Management
PAD	Personal Identification Device
EPC	Electronic Product Code
RFID	Radio Frequency Identification
BLE	Bluetooth Low Energy
M2M	Machine to Machine
SDK	Software Development Kit
API	Application Programming Interface
IaaS	Infrastructure as a Service
PaaS	Platform as a Service
MQTT	Message Queue Telemetry Transport
CoAP	Constrained Application Protocol
TMS	Thing Management System
CA	Customer Attribute
FR	Functional Requirement

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

List of Figures

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

List of Tables

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

1 Introduction

The first chapter of this thesis provides a brief introduction. The chapter begins with the problem statement which motivates the research conducted throughout and presented in this thesis. Subsequently the research aim, objectives and questions are presented to systematically guide the study of this thesis. An overview of the structure of this thesis is provided in the last section of this chapter.

1.1 Problem Statement

The Internet of Things (IoT) is a network of addressable, physical objects that contain embedded sensing, communication and actuating technologies to sense and interact with their environment. This network creates ecosystems that contain various services and applications (e.g. communication-, sensing, data analysis -services) (Geschickter 2015). Like every novel paradigm, the Internet of Things sparks interest throughout all domains both in theory and practice. Ever since IoT was added to Gartner´s *Hype-Cycle of emerging technologies* in 2011, it was either considered as "*on the rise"* or *"at the peak of inflated expectations"* (Fenn & LeHong 2011; 2012; 2013; 2014). In both hype-cycle-states IoT received high media coverage and an increasing number of companies started to assess how the Internet of Things could be integrated into their business strategies (Linden & Fenn 2003). Due to the novelty of this new paradigm many first-generation products were created under the label of IoT. However, neither businesses nor researchers have agreed upon a common, holistic understanding of the term *Internet of Things* during the hype (Wortmann & Flüchter 2015). These first-generation products are prone to negative publicity and technical issues. This is mainly due to the technology being far from mature and pushed to its limits. In fact, most IoT related projects and applications can still be considered as prototypes. The design of these prototypes may very well have been intuitively guided by the idea to create a network of humans and things alike – *"(…) experimentation at essentially full scale"* (Vaishnavi & Kuechler 2007 p. 10).

In the case of IoT, these limits became visible when having to manage increasing amounts of different things and ensuring connectivity among them and to other internet services (Lee & Lee 2015). By trying to incorporate a plethora of different things into a global network, issues regarding scalability, heterogeneity, interoperability, and standardisation arise (Perera, Zaslavsky, Liu, et al. 2014; Atzori et al. 2010; Moreno-Vozmediano et al. 2013). Additionally, the abstract nature of the value IoT can provide proved to be an issue for starting IoT initiatives in the first place. According to Gartner, the ignorance regarding the value of data provided by IoT applications hampered the adoption of IoT in some cases (Velosa et al. 2015). These issues as well as the insights gained through the first prototyping phase led to a special circumstance of IoT in Gartner´s Hype-Cycle of emerging technologies. Despite the removal of the overall concept of IoT from the Hype-Cycle in 2016, several sub-concepts of IoT have been added in 2015 and quickly reached the brink of the "peak of inflated expectations" in 2016 (Velosa et al. 2015; 2016b). This *"split"* is a special circumstance of a technology in Gartner´s Hype-Cycle, where a

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

technology is split into several sub-concepts (Linden & Fenn 2003). The newly emerged technologies are "*Internet of Things Architecture"* and *"Internet of Things Platform"*. Both technologies aim to tackle the above-mentioned issues. IoT architecture tries to pave the way for future IoT applications. It deals with developing networks and their architecture to support and manage increasing amounts of things. While businesses try to gain a competitive advantage by preparing themselves for IoT, e.g. by reassessing their business models for IoT or gathering knowledge (Velosa et al. 2015), researchers try to provide generic architecture models for the Internet of Things. For example, Khan et al. (2012) propose a generic IoT architecture consisting of five layers, namely business-, application-, middleware-, network- and perception-layer. They assigned each layer to specific tasks, responsibilities and requirements, e.g. the perception-layer is responsible for gathering data regarding the environment. This architecture research tries to introduce a common understanding of the Internet of Things, because all efforts of deploying the Internet of Things on a global scale is futile without a well-defined architecture (Mashal et al. 2015).

IoT platforms aim to enable secure connectivity between things, be it humans, sensor-devices or services of some sort, and are regarded as an integral part of any IoT architecture (IoT Analytics 2015; Mineraud et al. 2016). These platforms provide software suites and various cloud based services to facilitate the operation of *"IoT endpoints"* to enable communication between various, different devices. Currently available IoT platforms provide functions for device and application management (PaaS), data aggregation, transformation, storage and management as well as some means to analyse and visualise data streams (Velosa et al. 2015; Mineraud et al. 2016). However, if a user wants to use the functionalities of any IoT platform he must adhere to some constraints imposed by IoT platforms. Applications built on top of an IoT platform need to adhere to the requirements of that very platform. However, applications adhering the requirements of a specific platform created with a toolkit provided by that platform are tied to that specific IoT platform (IBM 2016a). In order to be able to access and manage things through an IoT platform users must create an application specific *"IoT endpoint"* on the platform and configure their devices to communicate with that endpoint. The communication (e.g. data format, communication technology/ protocol, etc.) are dictated by the IoT platform (Ishaq et al. 2013). These constraints of IoT platforms lead to several lock-in effects. The platform specific application requirements lead to the use of proprietary solutions (e.g. toolkits, protocols, data formats, etc.), which leads to difficulties regarding communication and migration between different IoT platforms (Mosser et al. 2012; Yasrab & Gu 2016; IBM 2016a). Thus, applications developed for one IoT platform are often not portable between platforms. This effect is called *data lock-in* (Mineraud et al. 2016). Another consequence of IoT platforms relying on proprietary solutions is the *vendor lock-in*. To provide a seamless integration of sensors and devices into an IoT platform, gateways and sensors must often be from the same vendor. Additionally, vendors are forced to make their devices compatible to the proprietary interfaces dictated by IoT platforms (Velosa et al. 2015). This results in specific IoT platforms only supporting devices from specific vendors (Ahmad et al. 2016). These lock-in effects are created due to the predominant lack or competition of standards in IoT and lead to two additional issues (Lee & Lee 2015; Yasrab & Gu 2016).

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Firstly, although it is simple for users to create an IoT endpoint, register and configure their things for that endpoint and create IoT applications (e.g. most IoT platforms provide tools for visual application development), the applications and device management functionalities lack dynamics. For example, it is not possible to dynamically add and remove endpoints and devices to applications created with currently available IoT platforms (IBM 2016b).

The reason for this lack of dynamics lies in the low level of abstraction of IoT application development required by current IoT platforms. On the one hand, users have visual tools (e.g. node based and visual development tools) to create an application. On the other hand, users need to directly address devices (e.g. provide a one-to-one mapping of an "IoT endpoint" node to a device talking to that endpoint). Thus, adding a new device requires creation of a new node on the respective application, provision of a device-node mapping, manually normalising of the retrieved data and redeploying the application on the IoT platform (IBM 2016a).

Secondly, applications built on top of IoT platforms cannot communicate with applications, devices or services from other IoT platforms in a simple fashion. This problem is called the *"IoT Gateway Problem"* and was addressed by Zachariah et al. (2015). According to Zachariah et al., each different type of device (e.g. sensors, wearables, etc.) requires a different gateway to be connected to the internet or other services. A gateway is either a device or an application (e.g. an application installed on a smartphone) which handles and normalises communication near the network edge. The reason for this lack of inter-device or cross-platform is two sided. The first reason lies in the fact that there are no uniform data formats available, which are usable across platforms and different types of devices (Mineraud et al. 2016). To cope with these issues of data heterogeneity, data scheme identification and fusion, Mineraud et al. (2016) suggest that IoT platforms should provide catalogues containing semantic indexes and uniform interoperable data models which can be used to identify and manage data schemes. Without these functionalities users are currently required to manually identify data schemes, normalise, transform and store the data retrieved by their devices. Thus, each IoT applications database will likely contain different data structures which makes inter-application and cross-platform communication difficult. However, the second and more severe reason is the lack of actual communication functionalities provided by IoT platforms. Lee and Lee (2015) state that there are currently many different competing standards in the domain of IoT, whereas each enterprise building an IoT platform tries to introduce their preferred standards. Various researchers state that the introduction of standards is critical for the adoption of IoT on a global scale (Atzori et al. 2010; Mashal et al. 2016; Mineraud et al. 2016). Despite the need for standardisation, some researchers (e.g. Katasonov et al. 2008; Tima et al. 2009) argue that semantic technologies are a better approach to cope with the heterogeneity of devices and protocols than enforcing a common standard.

In order to improve system dynamics of IoT platform applications and to remove cross-platform communication barriers, a common understanding of the elements forming an IoT architecture is required. Furthermore, the required functionalities of IoT platforms which are an essential element within an IoT architecture must be defined. Researchers have proposed a plethora of IoT architecture

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

models (Sheng et al. 2012; Mizouni & El Barachi 2013; Zaslavsky et al. 2013; Al Nuaimi et al. 2012) and component definitions (Zachariah et al. 2015; Ha et al. 2015; Petrolo et al. 2017; Serdaroglu & Baydere 2016) to overcome the mentioned issues.

Among these architectures Sensing as a Service (S2aaS), proposed by Sheng et al. (2012), has sparked interest in particular. This architecture combined with various proposals for IoT components (e.g. middlewares, gateways, services, consumers, etc.) will form the basis for the overall research aim of this thesis. The thesis aims to increase the abstraction level of IoT application development by examining S2aaS´s conceptual components and proposing a holistic architecture framework which can be implemented.

This architecture framework aims to combine the various IoT architecture proposals to a common denominator. Based on this holistic framework literature proposing implementations, technologies or behaviours for each component of the architecture can be examined and the definition of each architectural component can be refined.

An increased level of abstraction in the development process addresses both issues of IoT development platforms mentioned previously. The lack of system dynamics (e.g. because a user cannot simply add new IoT endpoints with respective devices on-the-fly) is reduced by introducing and specifying high level architectural elements. These high-level elements, their behaviour, interfaces, roles and data structures are to be defined by the holistic architecture framework. These high-level elements could then be used by users in the development process. An exemplary element may be responsible for managing and detecting sensors based on specific rules (e.g. a gateway-element). Furthermore, by relieving users of the onerous task to manually handle and transform different data structures and enforcing a common data format through the high-level architecture itself (e.g. common specifications for data elements and communication interfaces), barriers hampering cross-platform communication can be removed (Zachariah et al. 2015).

1.2 Research Aim, Objectives and Questions

By introducing a holistic high-level architecture framework, a common understanding of IoT and its respective elements, actors, roles and responsibilities is to be achieved. Based on this common understanding, barriers of cross-platform communication and issues regarding systems dynamics in IoT application development on IoT platforms can be addressed. To achieve the fundamental research aim of this thesis, which is to increase the abstraction level of IoT application development by examining S2aaS's conceptual components and proposing a holistic architecture framework which can be implemented, several research objectives must be fulfilled. As stated before, there are many proposals for IoT architectures and descriptions of specific components. Among these architecture proposals Sensing as a Service (S2aaS) (Sheng et al. 2012; Perera, Zaslavsky, Christen, et al. 2014) has been selected as a foundation, as it describes core concepts, actors and components of IoT on a high level of abstraction. Furthermore, many architecture proposals either aim to provide the application

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

environment for S2aaS or basically describe the very same components of S2aaS (Al Nuaimi et al. 2012; Abdelwahab et al. 2014). S2aaS is mostly considered a cloud service which heavily relies on the interaction with other services (e.g. sensor services, data services, etc.). Therefore, it requires a suitable environment (Zaslavsky et al. 2013). In order to achieve a common understanding of S2aaS along with its components, actors and perspectives the first research objective (RO1) must be fulfilled.

RO1 To identify, synthesise and evaluate Sensing as a Service components, requirements and perspectives.

To achieve RO1, components of S2aaS which are discussed in the pertinent IoT architecture literature must be identified. Since there are multiple architecture proposals relating to Sensing as a Service, whether implicitly or explicitly, common features of these elements must be identified. Additionally, architecture proposals differ regarding their perspective on the Internet of Things. The perspective of the architecture proposal might be rather technical or on a business level and consequently influence the naming, description and nature of the architectures elements. However, the components of the various proposals and perspectives are to be combined into common requirements of architectural elements. Thus, to achieve RO1 the following research questions must be answered.

RQ1.1 Which components of S2aaS are addressed in the pertinent IoT architecture literature?

RQ1.2 Which perspectives on the components of S2aaS are to be considered?

RQ1.3 What common requirements for S2aaS components can be defined?

After having achieved RO1, research regarding IoT components is examined to identify already existing systems, services and concepts which fully or partially fulfil the component´s requirements worked out in RO1. By providing a mapping between architecture components and existing concepts, systems and services, the rather abstract descriptions of the architecture components can be refined. This is achieved by incorporating the specific descriptions and specifications of these existing services, concepts and systems into the definitions of the architectural components created in RO1. In order to be able to select these existing concepts a mapping between services/concepts/systems and architectural components must be realised (e.g. the concept of an IoT gateway or middleware is mapped to an architecture component). Thus, the next objective is to provide a mapping between existing IoT services, systems and concepts and S2aaS architectural components.

RO2 To map architectural components of S2aaS to existing IoT services, systems and concepts.

To accomplish RO2 two research questions must be answered. The first question aims to find out how existing services, systems and concepts can be mapped to architectural components of the Sensing as a Service architecture. This mapping should be able to classify services, systems and concepts (e.g. gateways, middlewares, etc.) and assign them to the respective S2aaS architecture element(s). Based on this mapping some existing services, systems and concepts will be identified and mapped onto architecture elements to answer the second research question of RO2.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

RQ2.1 How can existing services, systems and concepts be mapped to components of S2aaS?

RQ2.2 Which existing services, systems and concepts can be mapped to components of S2aaS?

Based on the high-level S2aaS components and their likewise high level requirements from RO1 and the mapped systems, services and concepts as a result of RO2 a detailed specification of the S2aaS components is to be achieved. Based on the refined descriptions and requirements for each system, service or concept in RO2, which are likely to be much more detailed than the high-level specifications of the components from Ro1, and the mapping between different levels of abstraction the fulfilment of the next research objective becomes possible.

RO3 To propose detailed specifications for the components of IoT architecture framework.

This research objectives can be achieved by simply answering RQ3.1.

RQ3.1 What are the specifications for each component?

Based on the outcome of RO3 existing technologies can be identified which might fully or partially support the required functionalities of each component of the proposed holistic architecture framework. Therefore, the following and last research objective is to be achieved.

RO4 To find technologies supporting the implementation of the proposed architecture framework.

RO4 is achieved firstly by identifying criteria for selecting technologies. These criteria are drawn from the requirements developed in RO1 and the detailed specifications from RO3. Secondly, existing technologies (e.g. communication protocols, existing applications, code libraries, etc.) need to be selected based on the previously defined criteria.

RQ4.1 Which criteria are important for the selection of technologies that support the implementation of the proposed architecture framework?

RQ4.2 Which technologies are suitable to implement components for the proposed architecture?

By accomplishing RO1 to RO4 and answering the respective research questions (see Figure 1) a common understanding of IoT and its elements can be achieved, and thus the overall research aim.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

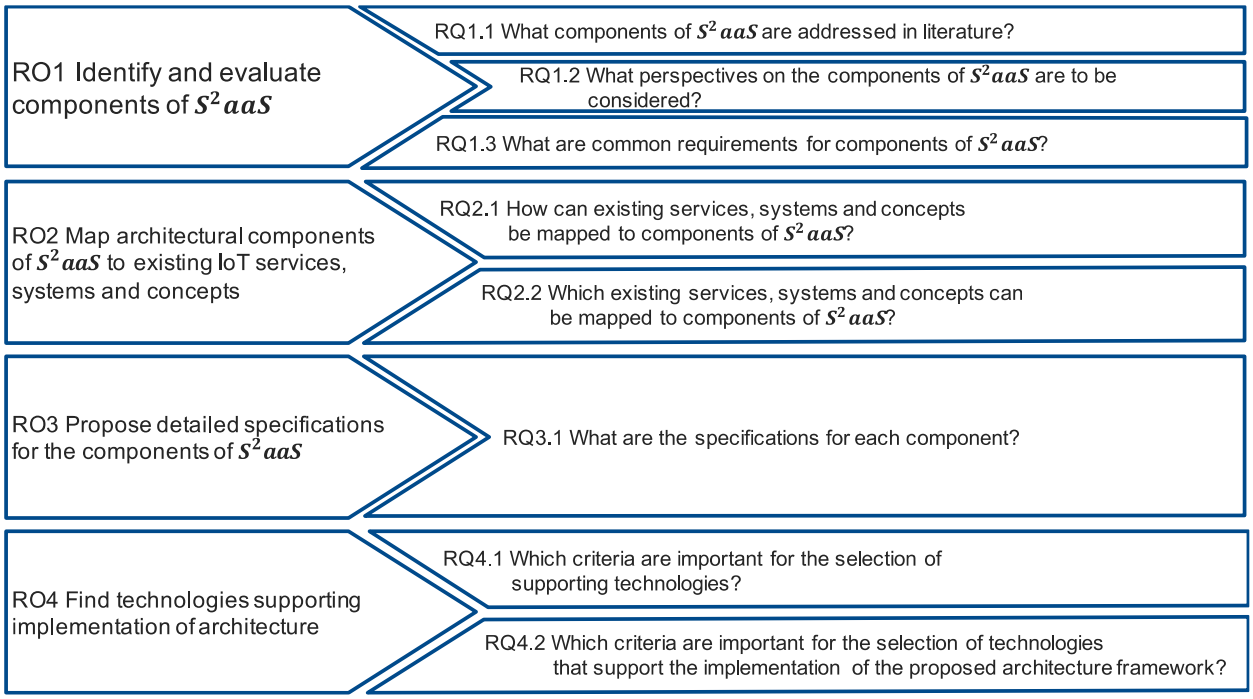


Figure 1. Research objectives with associated research questions (own illustration)

1.3 Outline of the Thesis

This section provides an overview of the structure of this thesis. The thesis consists of six chapters, which are briefly described in the following paragraphs.

Chapter 1 provides a brief introduction, consisting of the problem statement and the research aim and objectives guiding the further research conducted throughout this thesis.

Chapter 2 elaborates on the research design that provides the basis for the research which was carried out in this thesis. The chapter begins with the description of *Design Science Research*, which is used as the methodology (see section 2.1.), followed by a description of the research methods used in each phase of *Design Science Research* in section 2.2. The following sections discuss the data sources and collection methods (section 2.3) as well as the scope and basic theory (see section 2.4). The chapter concludes with an illustration of the individual research steps performed in this thesis.

Chapter 3 introduces the Internet of Things (see section 3.1) and specifically describes IoT platforms in section 3.2 to provide supplementary information for the following chapters.

Chapter 4 represents the main part of this thesis. It describes *Sensing as a Service* in section 4.1, followed by the development of *IoT Architecture Perspectives* and of the components of the *Holistic IoT Architecture Framework* (see section 4.2). Based on the components and perspectives developed in the previous section, a novel component for the *Holistic IoT Architecture Framework* is developed in section 4.3. The chapter concludes with the combination of the components developed in section 4.2 and section 4.3 which forms and finalises the *Holistic IoT Architecture Framework* (see section 4.4).

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Chapter 5 documents the prototype implementation of the components of the *Holistic IoT Architecture Framework* in sections 5.1 to 5.3. The chapter finishes with an evaluation of the implementation of the prototype application of the architecture framework in section 5.4.

Chapter 6 is the final chapter of this thesis and provides summarised answers to the research questions (see section 0) and discusses the research contribution in section 6.2. The last two sections discuss the limitations of the research conducted throughout this thesis (see section 6.3) and possible topics and challenges for potential future work (see section 6.4).

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

2 Research Design

In this chapter the research design used throughout this thesis is presented. This chapter deals with the methodology (see section 2.1), the research method (see section 2.2), the data sources and collection methods (see section 2.3) as well as the scope and basic theory (see section 2.4). The chapter concludes with an overview of the research steps and the corresponding methods for analysis in section 2.5.

2.1 Methodology

This thesis aims to achieve a common understanding of IoT architecture elements. To accomplish this, a holistic architecture framework is to be developed. The development of the framework is based on the analysis of existing architecture proposals for the Internet of Things. Thus, the phenomenon of interest as well as the desired outputs of this thesis are elements of the area of Information Systems (IS) and artificially created (Vaishnavi & Kuechler 2004). The analysis and development of the architecture framework focusses on achieving and answering the research objects and respective research questions stated in section 1.2. The technical and organisational interdependencies and elements of IoT architectures are the phenomenon of interest. The new holistic IoT architecture framework is to be created to improve the knowledge and understanding of IoT and to address the issues, such as system dynamics of IoT platform applications, mentioned earlier in this thesis (see section 1.1). Based on the phenomenon of interest and the desired output of this thesis, Design Science Research (DSR) is used to develop the IoT architecture framework as an artefact and evaluate it. This methodology is deemed to be especially suited for developing IS artefacts and provides a framework for performing said tasks (Vaishnavi & Kuechler 2007). The General Design Cycle (GDC), described by Takeda et al. (1990) and Vaishnavi & Kuechler (2007) i.a., is an integral part of DSR and can be used to structure the research process.

This cyclic model consists of five individual process steps. Research or design using this model begins with the awareness of problem. In this step a problem is to be identified, whether intentionally or not. This first step is the most unstructured part of the GDC. A problem can arise during literature research or in practical engagements with an already existing IS artefact (Vaishnavi & Kuechler 2007). In the original GDC model Takeda et al. (1990) suggest that the designer or researcher becomes aware of a problem, which is taken from a known or unknown set of existing problems, and decides whether this problem is to be solved. The problem addressed in this thesis, which is the lack of a common understanding of IoT architecture elements, responsibilities and related problems (see section 1.1), became clear during the initial and preliminary literature research and analysis as well the explorative use of IoT platforms. In the second process step of the GDC, the suggestion, existing knowledge of the problem domain is used to abductively draw suggestions for solving the problem. Alternatively, the suggestion can also be developed by using appropriate research methods or patterns of DSR (Vaishnavi & Kuechler 2007). The result of the second process step of the GDC is a tentative design which in turn guides the development of a new artefact. In the development step of the GDC most of the design tasks

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

are performed. The tentative and likely incomplete design is further refined and continuously improved and a new artefact is developed. The nature of the artefact depends on the phenomenon of interest and can range from actual implementations of software systems to rather abstract manifestations in form of models or other constructs (Vaishnavi & Kuechler 2004). The next process step of the GDC consists of an evaluation of the artefact. Usually the tentative design already contains some evaluation criteria, whether explicit or implicit. The result of the evaluation are insights regarding the performance of the artefact and its capability of solving the problem state in the first step (awareness of problem). The three steps *suggestion*, *development* and *evaluation* are often repeated multiple times and the results and insights gained in each evaluation and development step are used as input for the next cycle´s suggestion step to further improve the artefact (Vaishnavi & Kuechler 2004). The last process step of the GDC, the conclusion, marks the end of a DSR project as well as the GDC. Additionally, Kuechler et al. (2005) suggest an extension of the GDC which links multiple instantiations of a GDC associated to different research projects and domains together. The result and insights of one GDC is used as a starting point for another research and design project. This extension is called Aggregate General Design Cycle (AGDC). However, this thesis only utilises one instance of a GDC.

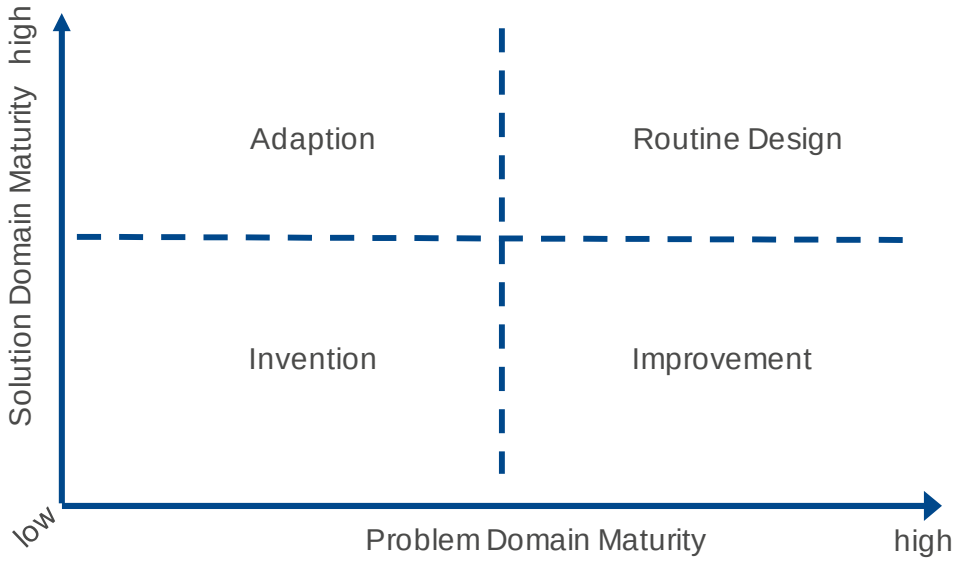


Figure 2. Knowledge Contribution Framework (adapted from Gregor & Hevner 2013)

The kind of knowledge contribution of a DSR projects depends on two factors (see Figure 2), the *maturity* of *problem* and *solution domain* (Gregor & Hevner 2013). The problem domain is the domain in which the problem has been identified during the process step of the GDC *awareness of problem*. In this thesis, the Internet of Things constitutes the problem domain. The solution domain, from which existing knowledge is used to draw conclusions and create a tentative design that guides the subsequent steps, is *Identity Management* and will be thoroughly explained in section 2.4.

The knowledge created throughout a DSR project can be classified as part of one of the following classes of outputs: constructs, models, frameworks, architectures, design principles, methods, instantiations

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

and design theories. For a detailed explanation of each of these classes refer to Vaishnavi and Kuechler (2004). The desired outputs of this thesis are constructs, which are the common components required by RO1 and RO3, as well as models defining the relationship between these components. An instantiation should then operationalise the previously defined constructs and models to aid the *evaluation* step of the GDC.

2.2 Research Method

The methodology chosen and presented in the previous section inevitably guides the selection of appropriate research methods. However, due to the fact that the IS research community is considered a multi-paradigmatic research community, the research method can be chosen relatively freely (Vaishnavi & Kuechler 2007). As already stated, this thesis applies Design Science Research as its research methodology. For each process step of the GDC mentioned previously different research methods or patterns can be applied. This thesis utilizes the patterns presented by Vaishnavi and Kuechler (2007) as research methods or at least as supporting guidelines for performing each process step of the GDC.

The conducting of the first process step of the GDC, *awareness of problem*, is guided by the meta level pattern "*Questioning Constraints"* presented by Vaishnavi and Kuechler (2007). This pattern is labelled as a meta level pattern because Vaishnavi and Kuechler state that it is applicable during each process step of the GDC. This pattern aims to identify research gaps by questioning constraints imposed on a research problem. It does not matter if these constraints are implicitly or explicitly mentioned by the research community dealing with the research problem (Vaishnavi & Kuechler 2007). This pattern is especially suitable when a researcher starts to work in a new field and thus is able have an unbiased view on the field. In addition, the researcher should have some knowledge on adjacent research fields and related technologies that might have impact on the constraints.

The second and third process step, *suggestion* and *development* respectively, of the GDC are directed by the suggestion and development patterns *Theory Development* and *Problem Space Tools and Techniques* also presented by Vaishnavi and Kuechler (2007). The pattern *theory development* can be applied when the researcher intends to draw theory from his work. Theory, according to Vaishnavi and Kuechler (2007), can be new models, concepts and conceptual frameworks i.a. As this thesis intends to create an IoT architecture framework based on existing IoT architectures to provide a common understanding of IoT´s elements, this pattern can guide the development of such theory. Specifically, the *incremental theory development* described by Vaishnavi and Kuechler guides the development of theory, the subsequently created artefact, it´s evaluation and influence on the refinement of the theory (Vaishnavi & Kuechler 2007). The pattern *problem space tool technique* guides the researcher in finding appropriate tools to solve the research problem. These tools, which are applied to the problem domain, then guide the researcher in subsequent tasks. Vaishnavi and Kuechler (2007) state that the researcher should utilise his general knowledge of existing research tools and techniques to identify an appropriate candidate and apply it to the problem domain.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

The third process step, the *development*, of the GDC will additionally utilise another research method. Gero (2000) proposes *axiom based design research* as a research method for developing IS or design artefacts. This research method allows the creation of IS artefacts by first specifying several axioms and then deriving their logical consequences. Suh (2000) further elaborates this method in context of the design of software systems. Designing software systems, and any other design task, is based on the independence- and information axiom as well as the customer-, functional, physical- and process-domain and a mapping between these domains. This design framework requires the designer or researcher to adhere to two axioms. The *independence axiom* states that functional requirements must not interfere with each other. Each functional requirement should be satisfied by a design parameter. Depending on the requirements or dependencies between different design parameters a specific design can be classified as an uncoupled-, a decoupled or a coupled design, whereas an uncoupled design is considered the best solution (Suh & Do 2000; Park 2007). By using the *design matrix*, a supporting method mentioned by Suh (2000), the dependencies between functional requirements and design parameters can be displayed. Furthermore, Suh describes specific steps for designing a software system with axiomatic design.

In the first step the functional requirements are derived from the customer domain. In the next step a design parameter is assigned to each functional requirement that satisfies that functional requirement. Suh (2000) notes that there can be many different designs, which are mappings between different domains, that can satisfy the independence axiom equally. To be able to select the “best” design among a set of existing designs, Suh introduces the *information axiom*. The axiom states that the best design is the design that contains the least information. In the context of software systems Suh considers information as the complexity of the system. Thus the least complex design is the best design according to Suh (2000). Having selected the best design, the next step presented by Suh requires the decomposition of functional requirements. When the functional requirements are unclear or sufficiently complex, they need to be decomposed into sets of simpler requirements. After having decomposed each requirement, new design parameters must be assigned to the simple requirements and new more detailed design must again be evaluated according the information principle. The steps one to three are repeated until each functional requirement and associated design parameter of the design can be implemented without further decomposition (Suh & Do 2000; Park 2007). This procedure ensures that the resulting design or software system is highly modularised. As this thesis aims to create a holistic IoT architecture framework, it is desirable that this framework is highly modularised.

As described in section 1.2, RO3 aims to provide a detailed description of each architecture element. By applying axiom based design as a research method during the development step of the GDC, the elements of the IoT architecture shall be considered encapsulated, not as further decomposable elements. Thus, a separation of concerns between each element of the IoT architecture framework can be ensured.

The evaluation step of the GDC will apply the *demonstration pattern* mentioned by Vaishnavi and Kuechler (2007). The intent of this pattern is to demonstrate that the IoT architecture framework

 © 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

developed through is thesis is indeed implementable. During the implementation, the creation of an instantiation, further insights regarding the practicability and limitations of the architecture framework might be gained.

2.3 Data Sources and Collection Methods

During the initial phase of this thesis a literature research was conducted to become *familiar with the new area* of IoT. This literature research pattern described by Vaishnavi and Kuechler (2007) can be used after a research domain has already been identified. In this thesis, the research domain of interest is IoT. Vaishnavi and Kuechler suggest that researchers should review internet resources, literature on the research domain of interest as well as attend conferences of that domain. However, reviewing literature regarding IoT in general is deemed sufficient to become familiar with the topic in this thesis. For the first literature research the databases SpringerLink[1], ACM Digital Library[2], IEEE Xplore Digital Library[3], Science Direct [4]as well as ResearchGate [5]and Google Scholar[6] are used. This initial literature research was focused on IoT in general and on IoT test beds or simulation environments. Therefore, the following groups of keywords were used. The first group contained the keywords “Internet of Thing*” and “IoT”, whereas the “*” represents that the keyword may be stemmed (e.g. “Internet of Thing” or “Internet of Things” may be used). The next group contains “Simulation*” and “Testbed*”. The third group consists of the keywords “Information*” and “Context”. The groups are combined with “and” and each keyword in a group are combined with an “or” operator.

The result of this first literature research yielded 33 articles, ranging from 1990 to 2016 (year of publication). When reviewing these articles three loosely connected time periods stood out. The first period, ranging from 1990 to 2006 contained articles that dealt with the foundations of simulation techniques in general. The second period, from 2006 to 2011, focussed on low level issues regarding simulation and network environments. This period aimed to provide the basis for management and implementation of Wireless Sensor Networks (WSN), their simulation and other networking issues. The last period, from 2012 to 2016, mainly contained articles dealing with high level and architectural issues regarding large sensor or device networks, management of information as well as the value and trust of this automatically generated and processed information. During the first two periods, mainly technical issues were addressed in literature. The third period additionally tries to incorporate humans into the

[1] http://link.springer.com/

[2] http://dl.acm.org/

[3] http://ieeexplore.ieee.org/

[4] http://www.sciencedirect.com/

[5] https://www.researchgate.net/

[6] https://scholar.google.de/

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

newly proposed applications and architectures, so that information, privacy, trust, and security became apparent and important topics.

Among the articles assigned to the third period, was an article by Sheng et al. (2012) that presented a novel architecture model for IoT. This particular architecture, further extended by Perera et al. (2014) combined with the above described issues becoming nascent during the first exploratory use of IoT platforms is responsible for the shift in focus regarding the literature search of this thesis.

The subsequent literature research focusses on articles on IoT architectures, their corresponding elements and related concepts, systems and services. The keyword groups used to obtain articles dealing with the said topics are as follows. The first group again consists of “Internet of Thing*”, “IoT” and “Web of Service*”. The second group contains the keywords “architecture*”, “framework”, “model”. The third group consists of the keywords “mobile phone sensing”, “sensing service”. This second literature search yielded several articles, from which six articles where specifically dealing with IoT architectures. The articles and respective architecture proposals, element descriptions and containing systems, services and concepts form the basis for the analysis of IoT architecture frameworks and the identification of common elements, as required for accomplishing RO1, RO2 and RO3.

2.4 Scope and Basic Theory

Based on the articles proposing or describing IoT architectures, single elements or other concepts, systems or services discovered as part of the second literature search described in the previous section, an analysis and comparison of the IoT architectures is to be conducted. The analysis focusses on the architecture elements and their relations (e.g. interfaces with other elements). Since this thesis aims to provide a *high-level* architecture framework, aspects regarding the detailed inner workings of elements (e.g. energy efficiency or context detection techniques for gateways) are not taken into account for the intended analysis.

The basic theory that shall be incorporated into the construction of the IS artefact by applying the GDC is *Identity Management* (IDM). Although it is often stated that IDM has no clearly defined meaning, especially in the digital world, a preliminary definition can be built upon the commonly agreed constructs of IDM (Jahankhani et al. 2010). In essence, IDM deals with issuing credentials to users during the registration phase and subsequently identifying these users with the identifiers to grant or refuse access to systems, services or other digital systems (Jøsang & Pope 2005). This superficial definition also contains the common constructs used in IDM. Users are considered as *Entities*, credentials as *Identities* and a single attribute of these credentials (e.g. username, password or other information) as *Identifier*. The set of identifiers that are used to identify a user in a system is considered an *Identity Domain*, which imposes certain criteria on the selection of identifiers (Jøsang & Pope 2005). These constructs are then used by an *Identity Management System*, which provides several services (e.g. authorisation, authentication, enterprise directory, etc.) to users or entities (Jahankhani et al. 2010). The constructs

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

and their relationships as well as requirements for an *Identity Management System* are explained subsequently.

The core of identity management consists of *entities, identities* and *identifiers* (see Figure 3). An *entity* represented by one or more identities, whether completely or partially. Entities can be real-life persons, enterprises or other legal bodies or digital services/ things (Jøsang & Pope 2005). Entities can be represented by multiple identities (Bhargav-Spantzely et al. 2006). There also exists the concept of shared entities, where several individual entities act as a single entity in a specific context. An example for this are families or companies. They consist of individual entities (e.g. family members, employees), which in turn can have multiple identities themselves, but act as a single entity in certain contexts, where an outsider cannot tell which "sub-entity" he is dealing with (Jøsang & Pope 2005). Entities are then described or represented by *Identities.* Identities are context specific, thus an entity has different identities in different contexts, e.g. an identity of an entity in a social network might significantly differ from the identity provided by the passport (Jahankhani et al. 2010). Furthermore, identities can describe their corresponding entity only partially or completely, again depending on the context or application domain. The identity consists of a set of characteristics referring to the entity. Only the complete set of characteristics (e.g. the required fields in a registration form) assemble an identity in a context or application domain. Depending on the context, an identity can be unique or ambiguous (Jøsang & Pope 2005). *Identifiers,* or characteristics, are the building blocks of identities. Each *identifier* is a claim of the corresponding entity that is not necessarily verified or certified by a third party, they are merely assertions that an entity has a specific characteristic (Bhargav-Spantzely et al. 2006; Cameron 2005). Identifiers have several properties. They are either transient or permanent (e.g. a student-number is transient, a social security number is permanent).

An identifier can be self-selected or issued by an authority (e.g. usernames are self-selected, social security numbers are issued) (Jøsang & Pope 2005). Which identifiers are required and/ or used to uniquely identify an identity depends on the *identity domain* or context the identity is in. In an identity domain, all identities are unique. It is a *namespace* which enforces a one-to-one relationship between identifiers and identities. Thus, not every kind of identifier is suitable to build a namespace, e.g. the date of birth usually does not provide a one-to-one mapping. Designing such a namespace is challenging and the problem scales with the size of the *identity domain*. Additionally, identifiers selected for the namespaces must usually be easy to remember, because they are primarily used by humans (Cameron 2005). Identity domains, containing identities uniquely identified by a namespace are an integral part of *Identity Management Systems*. These systems manage identities and provide authentication, authorisation and directory services. Entities authenticate with the system by using the identifiers defined by the namespace of the identity management system´s identity domain. After being authenticated entities are authorized to use specific services according to roles they have been assigned to or according to some rules defined otherwise. These roles or rules are stored in the enterprise directory of the identity management system (Jahankhani et al. 2010).

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

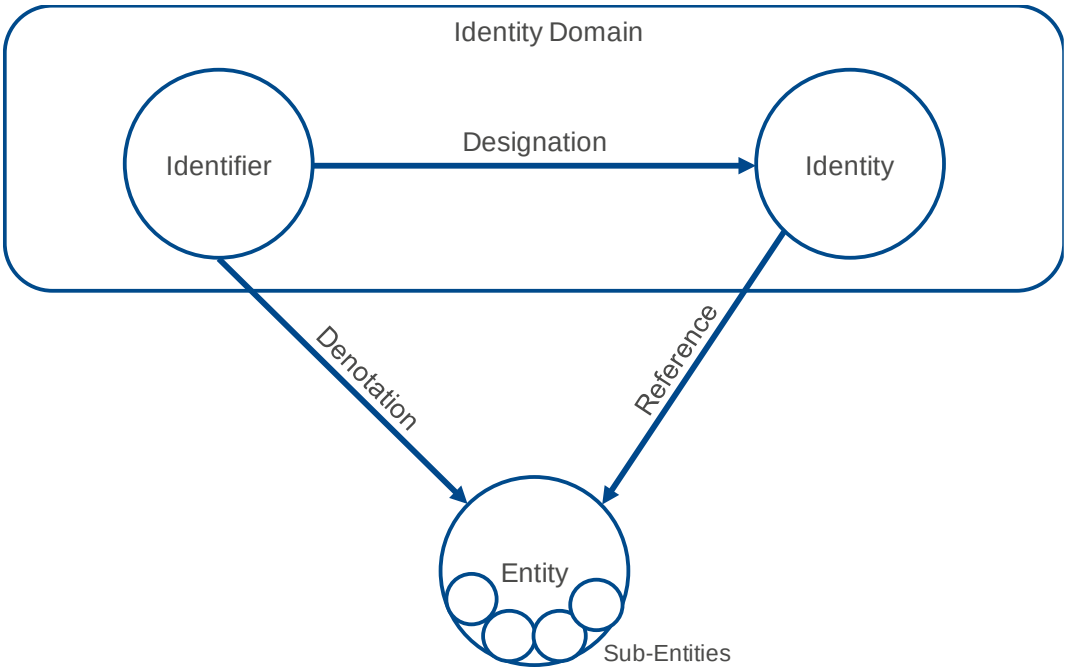


Figure 3. Relationship between Entity, Identity and Identifier (adapted from Sarma & Girão 2009)

According to Cameron (2005), identity management systems must adhere to the "*Laws of Identity*" that specify requirements for dealing with identities. The first law states that the end-user should always stay in control and must be asked for his consent when exposing any kind of information. The second law, *the law of minimal disclosure*, states that only a minimum amount of information must be shared with third parties. The third law requires that the third parties, with whom the information is shared, must have a necessary and justifiable requirement for obtaining that information. The fourth law requires that the mode of use for the identifiers must be able to be specified. This means that a user must be able to define in which contexts a specific identifier of one of his identities can be used. The fifth and sixth law require heterogeneity and human usability respectively. The seventh and last law states that the handling of identities must provide the same user experience across contexts (Cameron 2005).

Additionally, the research domain of identity management distinguishes between *User Centric-*, *Federated Identity Management Models* and *User Centric Federated Identity Management Models*, which is a mixture of both. However, Bhargav-Spantzely et al. (2006) note that these terms and especially *User Centricity* lack a common understanding.

A *Federated Identity Management Model* shares and maps the identifiers and identities provided by users (entities) among service providers. Normally, each service provider would have its own identity domain. In a federated identity management model however, the service providers have agreed upon a set of standards and technologies to share the identity information (Jøsang & Pope 2005). This creates a single shared identity domain. As soon as a user is authenticated with one service provider who is member of the identity federation, he is authenticated with all other service providers as well. However, due to the implicit sharing of identity information, the federated identity model violates the first three

 © 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

laws of identity in favour for a certain degree of “ease of use” for the user (Cameron 2005; Bhargav-Spantzely et al. 2006).

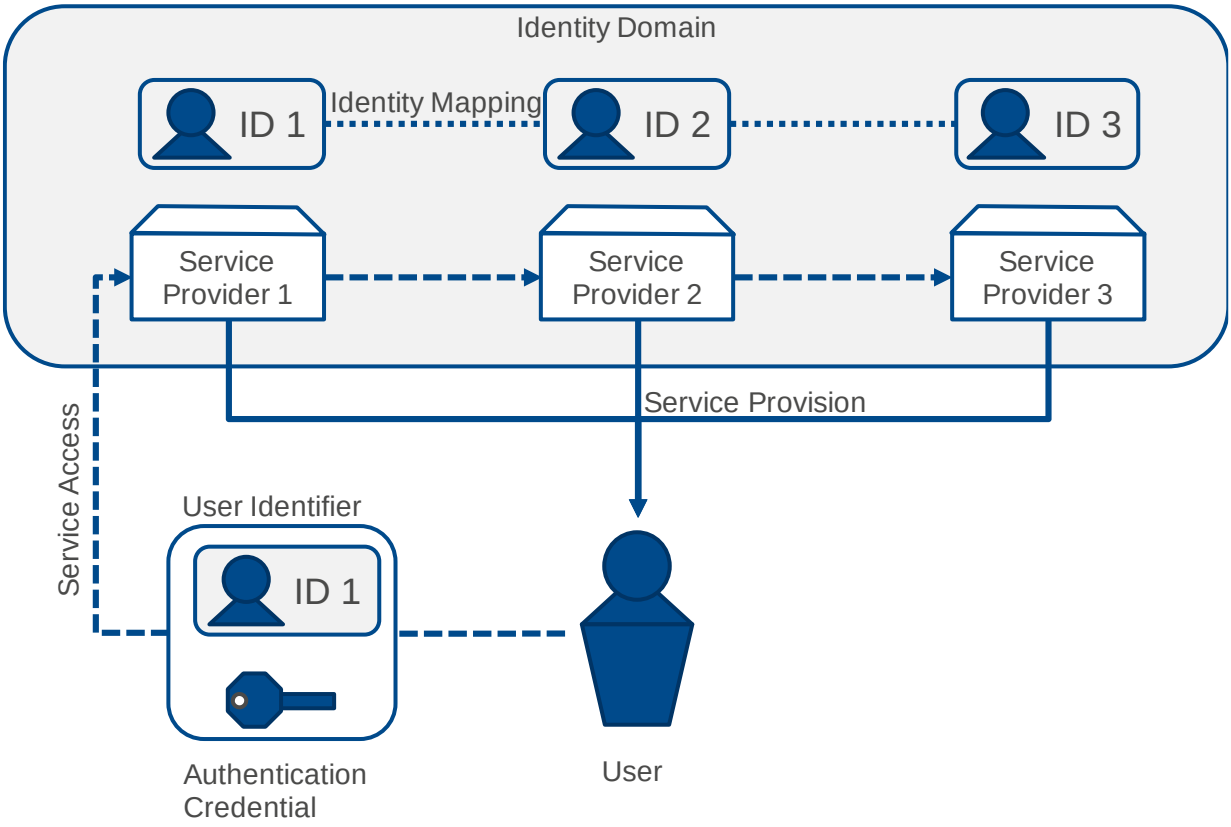


Figure 4: Federated Identity Management Model (adapted from Jøsang & Pope 2005)

The *User Centric Identity Management Model* comprises the idea that the user should always be in full control over transactions containing his identity. Bhargav-Spantzely et al. (2006) distinguish between two different types of user centric identity management, namely *relationship-* and *credential-focussed* identity management. In a *relationship-focussed* identity management approach a user, or entity, merely maintains a relationship with the identity management systems that has issued his credentials. The identity management system, sometimes called identity provider, is invoked in every transaction made with these credentials. The identity management system then handles the communication of identity information with the respective service provider used in the transaction. A good example provided for this type of user centric identity management is given by Bhargav-Spantzely et al. (2006). Credit cards, issued by a bank, are considered as a relationship between the issuer (the bank) and the holder (customer of the bank). These credentials/ identifiers used to identify the customer, or the entity, are usually the credit card number, the card validation code and a signature. The customer is in full control over his credentials and invoked in every transaction. When the customer uses the credit card to make a transaction to a service provider (i.e. when he wants to pay with the credit card), the bank must first validate the card and thus needs to identify the holder by using the provided credentials. Upon successful validation and authentication, the bank then handles the transaction of the relevant identity information to the respective service provider (Bhargav-Spantzely et al. 2006). A *credential-*

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

focussed identity management approach aims to obtain permanent credentials. These credentials are managed by the user (or entity) and can be used in transaction without invoking the identity management system that has issued these credentials (Bhargav-Spantzely et al. 2006). Jøsang and Pope (2005) suggest to use a *Personal Identification Device* (PAD) to locally store and manage credentials (see Figure 5).

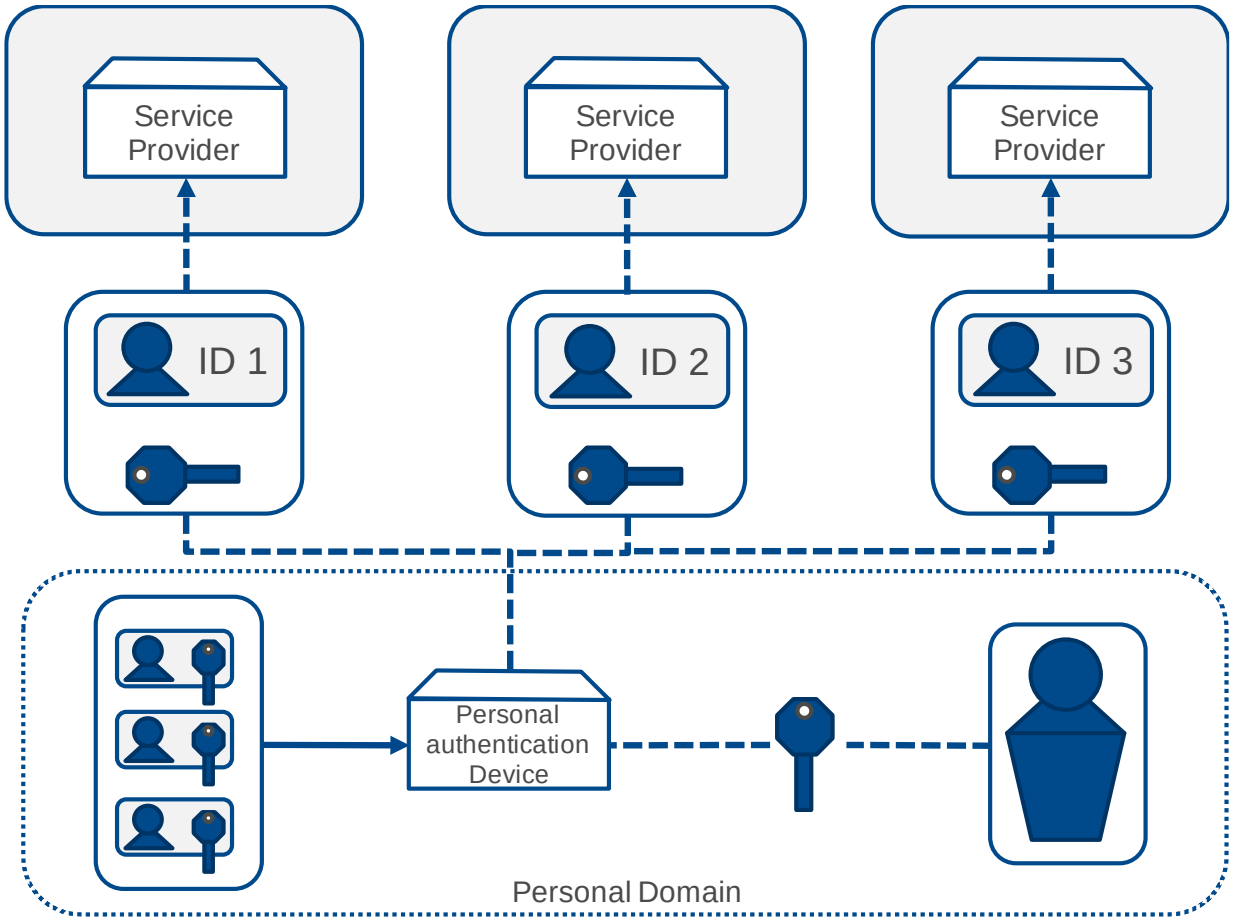


Figure 5: User Centric Identity Management Model (adapted from Jøsang & Pope 2005)

In this case the PAD is responsible for storing different credentials for accessing different service providers, whereas each has a different identity domain. However, the PAD can also be used to store long term credentials. Again, Bhargava and Spantzely (2006) provide an example for a *credential-focussed* identity management model where the example of credit cards need to invoke the corresponding identity managemen system. In contrast, passports are valid on their own. Passports are also issued by an identity management system, usually the government of the country a person lives in, and contains a set of identifiers to identify the holder of the passport without invoking the country´s government. All identifiers are stored locally, the user is also invoked in every transaction requiring his identity information and he is also in full control of this information (Bhargav-Spantzely et al. 2006).

The concepts provided by identity management are used in the development phase of the GDC described in chapter **Error! Reference source not found.** of this thesis. Since, the internet of things i ncorporates a vast number of devices, or things, and each of these devices has inevitably an owner in

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایـســـازی علـــمی»

the real world, one can conject that each of these things can be regarded as an identity representing its associated entity (Mazhelis et al. 2013). When a family decides to install various smart sensors in their house, each of these sensors exposes characteristic information, an identifier, regarding that family, be it simple temperature sensors or advanced cameras tracking and recognizing faces. Contrastingly, smart things can also act as service providers, where identities can be authenticated to perform certain actions. Fremantle et al. (2014) describe the scenario of a smart lock where the owner can grant access to the lock so that multiple persons, which are authenticated via their identities of the smart lock's identity domain, can unlock it. Additionally, some rules, such as access in a predefined period of time, can be defined. Thus, things can be both identities and service providers in terms of identity management.

2.5 Research Steps and Methods for Analysis

Since this thesis aims to create an implementable high level IoT architecture framework by applying design science research and the corresponding general design cycle as the research methodology, the selected method for analysis is the *demonstration pattern* as described in section 2.2.

For an overview of the research steps see Figure 6. In the first and second step of the GDC a tentative design based primarily based on academic literature and IoT platform reviews is created. As described in section 2.1, the *awareness of problem* and *suggestion* step of the GDC result in the research problem as well as a suggestion for the solution for that problem, which was suggested in section 1.1. The research problem that motivates this thesis is the lack of a common understanding of IoT architecture elements and resulting problems, such as a lack of system dynamics and insufficient interoperability due to heterogeneity issues. Thus, this thesis suggests to synthesise different IoT architecture proposals into a holistic IoT architecture framework. The research problem and the tentative design are elaborated on in section 1.1 and section respectively 1.2.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

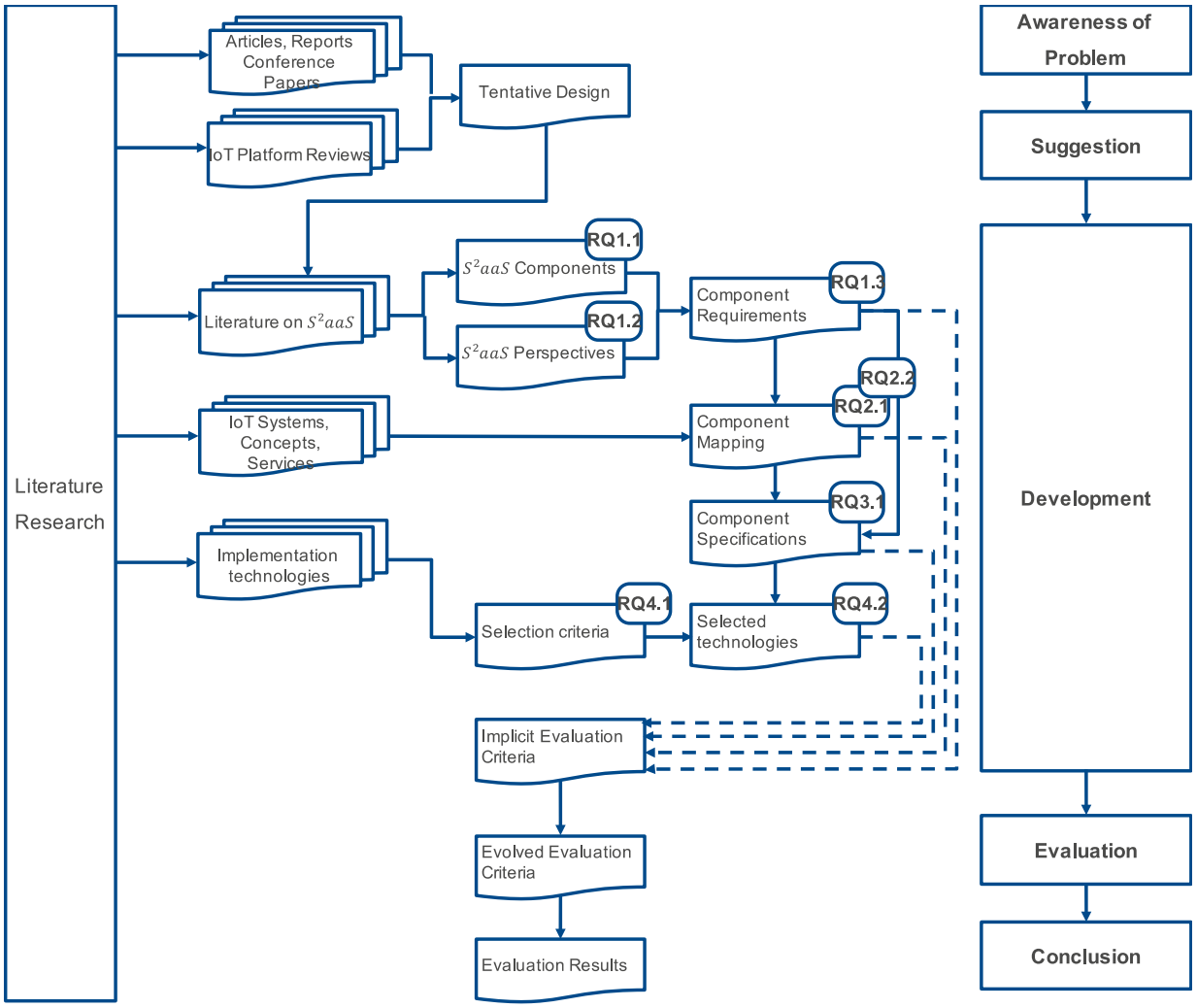


Figure 6: Research steps (own illustration)

Based on the tentative design, literature research on IoT architectures and Sensing as a Service (S2aaS) is performed to answer both RQ1.1 and RQ1.2. The data sources and methods of collection for the literature research is described in section 2.3. Based on the answers of RQ1.1 and RQ1.2, which are the S2aaS perspectives and common elements respectively, the actual component requirements are to be created in the next research step. This research step aims to answer RQ1.3. Having completed this step, RO1 is achieved and a first sketch of the high level IoT architecture framework is achieved. Knowing the requirements for each element, the next research objective can be addressed. Based on the component requirements (see RO1) and a literature research on IoT systems, concepts and services a mapping between existing IoT systems, concepts and services and the previously identified components is to be achieved in the next research step. This step aims to answer RQ2.1 as well as RQ2.2 and thus to achieve RO2. Now having detailed information of the relationships and perspectives of the architecture elements as well as an understanding about which existing IoT systems, services and concepts can be mapped to these elements, this information can then be combined to create detailed component specifications of the high level IoT architecture framework (see RO3). The last research objective (RO4) is achieved by conducting a review of suitable implementation technologies that could support the

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

prototypical implementation of the high level IoT architecture framework and the detailed component specification which are the result of achieving RO3. Based on the identified technologies, selection criteria are created and subsequently, based on these criteria and the detailed component specification suitable technologies are selected. The selected technologies are then used to create the prototype implementation of the high level IoT architecture framework. During the development of the architecture as well as the prototype several implicit evaluation criteria are likely to occur. After the development, the evaluation step of the GDC will make these implicit criteria explicit and evaluate the architecture framework as well as the prototype.

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

3 Theoretical Foundations

This chapter gives a definition of the Internet of Things, presents its history and different perspectives in section 3.1. Furthermore, this chapter provides a brief definition of IoT platforms in section 3.2.

3.1 Internet of Things

The Internet of Things is deemed to have a high impact on every aspect of everyday life (Barnaghi et al. 2012). It is envisioned to be the integration of the physical world into the digital world or vice versa (Fleisch 2010). Enterprises and countries alike have already assessed the importance of IoT and started to move into strategically advantageous positions to exploit the most value of IoT. For example, the National Intelligence Council (2008) sees IoT as one of the *Five Disruptive Civil Technologies*, which has a potentially important impact on the country´s interests out to 2025. Gartner (2015) advises enterprises to start or keep focussing on augmenting their business processes and models with IoT solutions and obtain expert knowledge as soon as possible. Gartner (2016a) also speculates that by 2020 the majority of new business processes will be supported by IoT technologies and solutions in one way or another. At the same time, Gartner (2016a) predict that by 2020 more than 20 billion devices are connected to the internet. On a side note, Gartner (2016a) also assumes that by 2020 there will be a black market worth of five billion US$ for fake sensor data, which emphasises the need for information reputation and evaluation techniques in IoT. For IoT to be successful, a trust environment has to be established (Botterman 2009). As already elaborated on (see section 1.1) and highlighted again in this chapter, IoT can very well be considered as a hyped technology (Fenn & LeHong 2011; 2012; 2013; 2014; 2015; 2016b). With IoT being a relatively new concept and under high coverage of both media, businesses and researchers it is not surprising that the term itself and related research field lack a commonly accepted and established definition. The commonly mentioned core ideas of IoT are the seamless integration of virtual and physical objects into a network, their contextual interaction and cooperation to reach common goals and their pervasive as well as ubiquitous presence in the real and digital world (Atzori et al. 2010; Mazhelis et al. 2013; Barnaghi et al. 2012; Fleisch 2010).

The very first idea that led to today's concept of IoT was the goal to integrate physical *things* into *digital* systems. The development was driven by the idea to enhance supply chains, trade and inventory management by applying *Electronic Product Codes* (EPC) to products and items. These EPCs can be used to store and share information regarding the items and products they are attached to. The information is shared by using standardised interfaces and utilizes *Radio Frequency Identification* (RFID) as well as the internet and related communication systems. Early research and efforts regarding standardisation were performed by EPCglobal (2009), which aims to introduce a global standardised architecture for EPC, and by the Auto-ID Labs (Auto-ID Lab 2017), which focusses on research regarding RFID networks and emerging sensor techniques (Atzori et al. 2010). Objects or *Things* equipped with RFID tags, usually consisting of an antenna and a microchip storing information, can be tracked by RFID readers which can read the information stored in the RFID tags. Thus, computer systems can obtain information of the real

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

world, i.e. of physical objects. The term *Internet of Things* is attributed to the Auto-ID Labs (Ashton 2009) and was later formally defined by the International Telecommunications Union (2005). As previously mentioned, the driving idea behind this early development stage of IoT was the goal to mesh the physical and the digital world. Atzori et al. (2010) theorise that this early research is guided by the *Thing Oriented Vision of IoT*, which will be elaborated on later in this chapter.

However, Atzori et al. (2010) also say that IoT can and will not merely be a global EPC system based on RFID. More different and kinds of *things* will be added and connected by using different communication technologies (e.g. NFC, Bluetooth Low Energy (BLE), etc.). Things will need to be managed and organised into networks (e.g. Wireless Sensing and Actuating Networks (WSAN)). The upcoming heterogeneity was already considered in the formal definition of IoT provided by the International Telecommunications Union (2005). The definition states that the internet cannot only connect anyone at any time and any place but *anything* at any time and any place. This view focusses more on the networking aspect of IoT and Atzori et al. (Atzori et al. 2010) name it the *Internet Oriented Vision of IoT.* Having agreed upon these ideas and with IoT gaining significant interest, both research and industry tried to develop relevant use cases for IoT. Among others, advantageous use cases in supply chain management, logistics and inventory management were identified. Apart from that, the *smart fridge* presented by LG in the year 2000 is used as a prominent example of IoT for consumers (Sone 2001; Rothensee 2008). With decreasing costs and increasing availability of relevant IoT technologies (e.g. RFID tags, low energy sensors and BLE, etc.), the number of *things* connected to the internet increased and rapidly outgrew the number of addressable devices used by the current addressing scheme of the internet (IPv4). With the introduction of IPv6 to accommodate a practically inexhaustible number of addressable devices in 2011 and the introduction of IoT to Gartner´s *Hype Cycle of Emerging Technologies* in the same year, the concept of the Internet of Things finally became visible for the broad public (Fenn & LeHong 2011; Madakam et al. 2015). Gartner considered IoT at this time as an *Innovation Trigger*, thus many prototype products, systems and supporting technologies were developed (see section 1.1) (Fenn & LeHong 2011). With a vast number of connected *things*, issues regarding information search, organisation and storage became apparent. As already noted, IoT inevitably implies a certain degree of heterogeneity. Therefore, traditional means of managing and searching information are rendered impractical (Atzori et al. 2010). Semantic technologies (e.g. Resource Description Framework (RDF), Web Ontology Language (OWL), etc.) are regarded as a potential solution for these challenges. This is why the *Semantic Oriented Vision of IoT* emerged (Atzori et al. 2010).

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

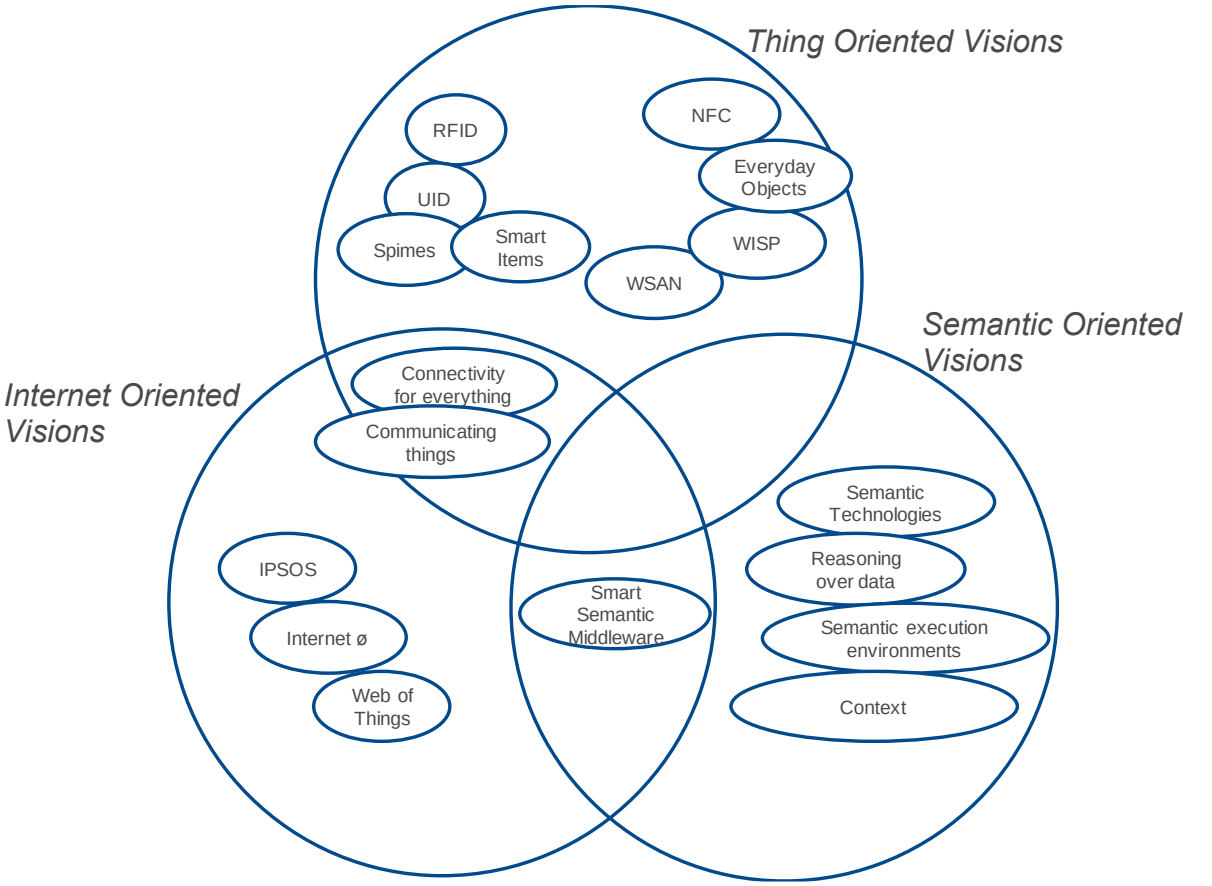


Figure 7: Perspectives of IoT (adapted from Atzori et al. 2010)

As mentioned earlies in this chapter, IoT is said to lack a commonly accepted and established definition due to high media coverage and interest from all different domains as well as the novelty of the research domain. Each of these domains or groups of interest, which focus on different aspects of the core ideas of IoT and enter different stages of the "rise of IoT", can loosely be assigned to one of the mentioned *Perspectives of IoT* (Atzori et al. 2010). These perspectives are the *Thing Oriented* -, *Internet Oriented* - and the *Semantic Oriented Perspective of IoT (see* Figure 7*).* In the following paragraphs, each perspective will be explained in more detail.

Thing Oriented Vision of IoT

As already mentioned, this perspective can be regarded as the first perspective of IoT. Focussing on RFID, WSANs and Smart Items it considers *Things*, along with their identity, functionality and information as the core of IoT (International Telecommunication Union 2005). According to Atzori et al. (2010) the core technologies of this vision are RFID and NFC, which provide wireless short to medium range communication. Everyday objects are then enhanced with these technologies to become *Spimes*. *Spimes* are theoretical objects that can be tracked throughout space and time, beginning from their manufacturing until their disposal (Atzori et al. 2010). A *Spime* is associated with an owner and contains information regarding their previous owners and their contents (e.g. which materials it is made of, etc.) (Sterling 2005). Furthermore, these objects are uniquely identifiable. This requires advanced addressing schemes, capable of maintaining great numbers of objects (Atzori et al. 2010). The concept of *Spimes* is

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

rather theoretical, however it already has practical application in form of *Smart Items*. *Smart Items* are devices with sensing-, storing-, (wireless-) communication- and elaboration capabilities. Moreover, *Smart Items* should be capable of autonomous actions based on contextual awareness and collaborative communication (Botterman 2009). These requirements for *Smart Items* are in line with the previously mentioned core ideas of IoT. Additionally, Mazhelis et al. (2013) state that the concept of a *Thing* doesn't have to be limited to physical items, they argue that virtual entities can also be a *Thing*. The IoT Architecture Reference Model, proposed by Bassi et al. (2013), backs this idea by allowing entities to either be physical or virtual. *Things* are responsible for gathering, (short term) storing and transmitting information and can be categorised according to their purpose (see Figure 8) (Mazhelis et al. 2013). *Identifying Things* assign a unique identity to an object and thus assign it to an addressing scheme. *Sensing Things* are not only identifiable but are also able to gather information regarding their environment and convert, store and communicate this information. *Embedded Things* have access to this sensed information and can process it and may be able to act upon the processed information. As *Things* must not necessarily be physical objects, *Embedded Things* can be some kind of service (Mazhelis et al. 2013). In conclusion, the *Thing Oriented Vision of IoT* focusses on the nature, purpose and use of the basic element of IoT, the *things*.

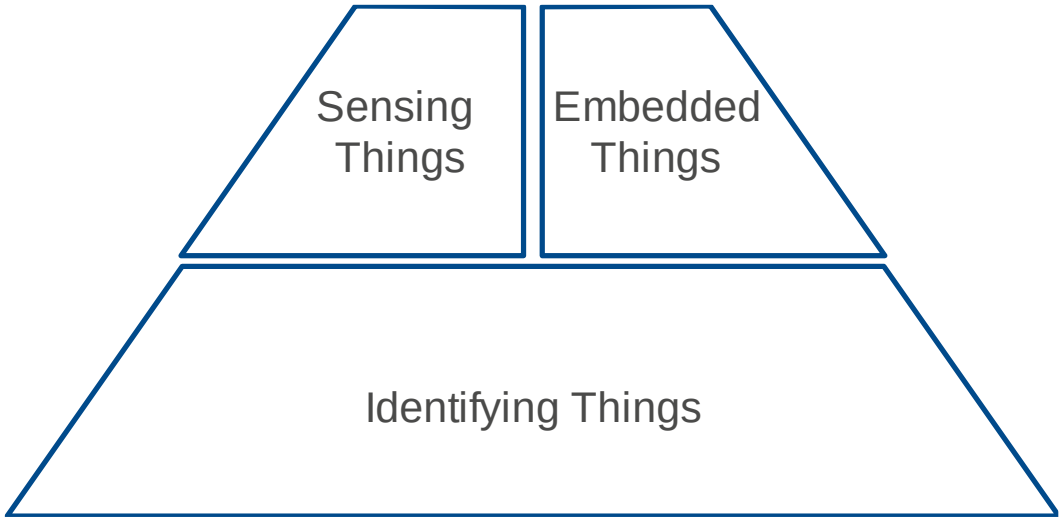


Figure 8: Categories of Things in the *Thing Oriented Vision of IoT* (own illustration, concept based on Mazhelis et al. 2013; International Telecommunication Union 2005)

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Internet Oriented Vision of IoT

This vision of IoT aims to incorporate existing developments of the current internet into the new infrastructure for IoT or vice versa. IoT can be very well considered as merely an extension of the current internet, with prosumers[7] being not necessarily humans anymore. However, with the inevitable heterogeneity of IoT, the current internet must adapt to be able to not only connect to anyone at any time and place, but to anything (International Telecommunication Union 2005). The majority of devices being connected to the internet will be constrained in either power consumption, computing power or data storage and will very likely need to rely on opportunistic communication availability. With the TCP/IP stack requiring relatively large amounts of power and computing capacity, connecting constrained devices to the internet via IP is a challenging task. Hence, one of the core ideas of this vision is to simplify the IP-stack to accommodate constrained devices (Atzori et al. 2010). Furthermore, this vision favours "IP over anything". Both the *IP for Smart Objects Alliance* (IPSOS) and *Internet Ø* aim to propagate the use of the IP stack as a light weight communication protocol for all kinds of devices (Mazhelis et al. 2013). Hui et al. (2009) argue that IoT vendors have embraced the use of proprietary protocols to connect their constrained devices, which creates the *"Gateway Problem"* as described in section 1.1 (Zachariah et al. 2015). One solution, that will exploit the already existing infrastructure, which is the aim of the *Internet Oriented Vision of IoT*, is the adoption of 6LoWPAN, which can be deployed on constrained devices due to its novel adaption layer (Atzori et al. 2010; Hui & Corporation 2009). In conclusion, the *Internet Oriented Vision of IoT* aims to use the internet and its related technologies as the drivers and solutions for IoT networking issues.

Semantic Oriented Vision of IoT

With interfaces mainly designed for humans or "simple", well defined and known services, the current architecture of the internet faces challenges when dealing with many kinds of interfaces or actors. The large amounts of devices further increase the difficulty of this challenge. Thus, the *Semantic Oriented Vision of IoT* focusses on semantic technologies to represent, search and store information in IoT. Furthermore, this vision aims to use semantic descriptions for interfaces, services, *things* and their corresponding identities (Mazhelis et al. 2013; Atzori et al. 2010). Bottermann et al. (2009) consider *ontology languages, flexible storages* (e.g. schema-less databases) and *reasoning engines* as important key technologies for this vision. The use of these technologies promotes semantic interoperability of IoT resources and the use of information models and semantic annotation of data (Barnaghi et al. 2012). Another focus of this vision is the use of *semantic execution environments* or context and semantic middlewares. Semantic middlewares are capable of negotiating between different kinds of devices with each device possibly having different data or information models (Katasonov et al. 2008; Botterman 2009). The use of context, for example the provisioning of services based on environmental information

[7] With the current internet (Web2.0) being focussed on collaboration, the borders between producers of content and consumers of content are blurred. Hence, *prosumers* create and consume information/ content (Ferna et al. 2015; Vazquez & Lopez-de-ipina 2008).

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

(e.g. location, weather, etc.) for a user or contextual collaboration based on common tasks, also relies on semantic annotations and descriptions (Atzori et al. 2010).

Having briefly described the history of IoT along with the interwoven development of the different visions of IoT, one can now better understand why different researchers emphasise different aspects in their respective attempts to define the Internet of Things. Mazhelis et al. (2013, p.9) consider IoT as *"A world-wide network of interconnected objects uniquely addressable based on standard communication protocols"* (INFSO D.4 Networked Enterprise et al. 2008, p.6). This definition can be associated with both the *Thing Oriented*- as well as the *Internet Oriented Vision of IoT*. In fact, Atzori et al. (2010) suggest that each vision overlaps in certain areas with other visions. In the case of this definition the focus is on *communication objects* and the suggested use of standardised communication protocols. Another definition provided by EpoSS (2008, p.6) states that *"Things having identities and virtual personalities operating in smart spaces using intelligent interfaces to connect and communicate within social, environment, and user contexts."*. While this definition specifies the nature of *Things,* it further emphasises the need for communication and intelligent interfaces tied to environments or *contexts*. Hence this definition can be associated with both the *Thing Oriented* – as well as the *Semantic Oriented Vision of IoT*. A definition aimed to encompass all three visions given by Tarkoma and Katasonov (2011, p.6) states that IoT is *"A global network and service infrastructure of variable density and connectivity with self-configuring capabilities based on standard and interoperable protocols and formats. IoT consists of heterogeneous things that have identities, physical and virtual attributes, and are seamlessly and securely integrated into the Internet."*.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

3.2 IoT Platforms

IoT platforms aim to simplify communication, storage, transformation, processing and control of data as well as devices that gather this data (Efremov et al. 2015). As stated in section 1.1, IoT platforms are considered an integral part in IoT architectures because they act as the middleware, translating and orchestrating communication between devices and (on-platform-) applications (Mineraud et al. 2016).

Kim et al. (2014) propose an "ideal Machine to Machine (M2M) platform model" (see Figure 9) that addresses various business models for the platforms as well as meets most requirements for IoT platforms identified in the literature. The terms IoT platform and M2M platform can be used equivalently. Mineraud et al. (2016) specify requirements for IoT platforms in order to perform a gap analysis of current IoT platforms. These platform requirements are presented in the following paragraphs, followed by the description of the "ideal Machine to Machine platform model".

Security and privacy

Enabling secure communication between devices and platforms is one key requirement IoT platforms need to meet. Hence, IoT platforms must include security and privacy mechanisms (Mineraud et al. 2016). This requirement is especially difficult to fulfil because IoT currently lacks communication standards (Perera, Jayaraman, et al. 2014). To meet these requirements, an IoT platform must provide means of device authentication, privacy of physical devices and communication, protection of data storage and devices, trust management, governance and fault tolerance. To authenticate devices IoT platforms mostly use keys that are transmitted with every communication to authenticate and identify the devices (Mineraud et al. 2016). The authentication of users can be handled by using the OAuth 2.0 as a standardised protocol. Additionally, IoT platforms should provide different levels of granularity for authorization e.g. for stored data or devices.

Integration of sensing and actuating technologies

As IoT platforms aim to simplify communication between heterogeneous devices, they need to provide toolkits and Software Development Kits (SDK) supporting a pool of standardised communication protocols (Mineraud et al. 2016). Due to the lack of standardised protocols in the current stages of the development of IoT, the platform should offer as many protocols as possible to accommodate as many different types of devices as possible. Mineraud et al. (2016) relate the value of an IoT platform to the variety of supported devices, which highlights the requirement to support many kinds of devices. Furthermore an IoT platform should make the integration of devices as simple as possible and provide the respective owner of the device with full control over the device by means of device management (Mineraud et al. 2016; Kim et al. 2014).

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Data ownership

Data generated by devices attached to an IoT platform must be owned by the user of the respective data generating device. Large volumes of data, being an important aspect of IoT and having potentially high financial value, are of utmost importance for the users of IoT platforms (Mineraud et al. 2016; Perera et al. 2013). Users must have full control over the data collected by their devices and must be able to decide with whom the data is shared. Furthermore, it must be legally guaranteed that the data is in their possession. Unfortunately, Mineraud et al. (2016) state that the data ownership is rarely guaranteed among current IoT platforms.

Data processing and sharing

With various kinds of devices, each collecting and transmitting data in different formats with different and sometimes unknown quality, an IoT platform must provide some means of processing and transforming this data (Mineraud et al. 2016). Additionally, it must be possible to share this data with other users, services or entities, so that this requirement is related to the *data ownership* requirement. In addition, processing and transforming data must be able to handle large bandwidths to process the vast amounts of data collected (Mineraud et al. 2016). In addition to sharing the data or data streams, users must be able to search for data and data streams. This requires that the data is annotated and the IoT platform provides some means of searching (daCosta 2013).

Application Development

IoT platforms must provide standardised application programming interfaces (APIs) that facilitate the development of IoT applications. These APIs should provide high-level access to the functionalities of the IoT platforms (e.g. querying data streams, requiring access to data or devices, retrieving data, etc.) (Mineraud et al. 2016). Ideally, these APIs should be uniform across different platforms. Fortunately, Mineraud et al. (2016) find that most IoT platforms provide APIs that follow the same principle, which are RESTful[8] APIs. However, the data models and specific characteristics of each API differs between each platform (Mineraud et al. 2016).

[8] Representational state transfer (REST or RESTful) is an architectural principle that uses a predefined set of stateless operations. These state transfer operations are usually tied to request methods of HTTP and abbreviated as CRUD (Create, Read, Update, Destroy) operations.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

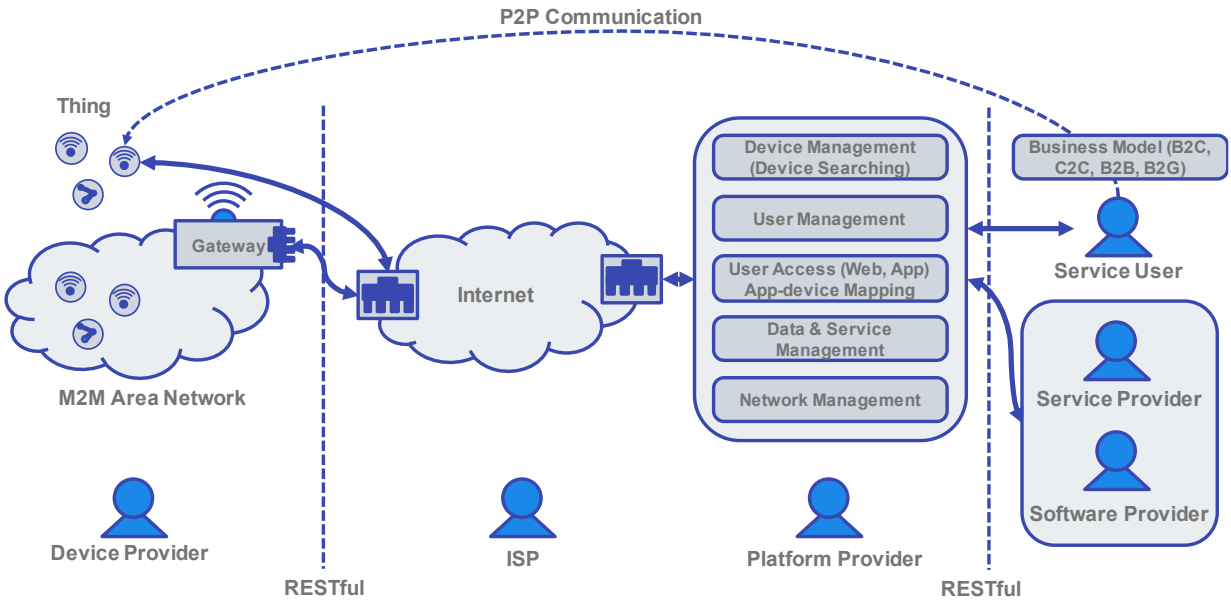


Figure 9: Ideal M2M platform model (adapted from Kim et al. 2014)

The “ideal Machine to Machine platform model”, proposed by Kim (2014), aims to meet the previously described requirements. The model is used by *service users*, which can be either customers, businesses or governmental bodies. The *service users* gain access to the functionalities of the platform via web- or mobile-applications. Additionally, the IoT platform provides a means of direct communication between *service users* and devices or *things*. The web- and mobile-application as well as direct API access to the platform is achieved by using a RESTful API. *Service* and *Software providers* may use RESTful APIs as well provide pre-built applications or additional, possibly external services or consumer data. The entities an IoT platform consists of are *device management* along with *device searching, user management, data & service management, user access* and *network management*. In general, *service users* register their devices with the platform. The devices can then be shared with other users and transmit their data to the platform in various intervals, when events occur or continuously (Kim et al. 2014). The transmitted data is then converted into meaningful knowledge that can be accessed by using dedicated services over the web- or mobile-application. Devices or things can be either individually addressed or are managed in M2M Area Networks and accessed via gateways which in turn are special devices registered with the platform. The functionalities offered by the proposed platform model are shown in more detail in Figure 10. The modules or groups of functionalities are explained in the following paragraphs.

Device Management

The *Device Management* part of an IoT platform provides the means for registering, managing, monitoring and searching for devices. It consists of the modules *Device Profile Management*, *Device and M2M Area Network Management* and *Device Searching*. The *Device Profile Management module* allows users to register devices. Information regarding the registered devices is stored in a database. Additionally, users can manage the profiles of the devices. The profile of a device contains information regarding its status, its owners, the type of data, access limitations and location. To ensure security, the

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Device Profile Management can provide devices with authentication keys and a authentication-key management system (Kim et al. 2014). The *Device and M2M Area Network Management* monitors the status of devices and controls them (e.g., it ensures connectivity between devices or performs low-level network organisation tasks). By having all device information stored in a database, the *Device Searching* module can execute queries to search for devices (Kim et al. 2014).

User Management

The user management entity of an IoT platform consists of the modules *User Profile Management, Authentication* and *Charging* and closely interoperates with the *Device Management*. The *User Profile Management* module allows users to register, modify their profiles and manage access to (shared-) devices and data. Users who registered devices are the owners or administrators of these devices and can grant or revoke access to their devices (e.g. direct P2P connection, access over IoT platform, etc.). The *Authentication* module is responsible for authenticating users and authorizing access according to their respective rights and roles. The *Charging* module monitors which resources users have consumed (e.g., which applications, services, etc. they used) and charges the users accordingly. Most IoT platforms facilitate the *pay-as-you-go* approach for charging (Perera et al. 2015; Mosser et al. 2012).

Application Management

The *Application Management* entity of IoT platforms is responsible for *Data Collection* and *Control*, *Services and Mash-up Management* as well as *Connection Management*. It provides access to a variety of services which either can be created by users, provided by external service and software providers (see Figure 9) or built-in into the IoT platform (Kim et al. 2014). The *Data Collection* and *Control* modules are responsible for collecting and reasoning over data. Based on the data collected by the devices connected to the IoT platform, these modules suggest appropriate services. The *service and mash-up management module* manages the services provided by the IoT platform. As the platform is either considered as an Infrastructure as a Service (IaaS) or as a Platform as a Service (PaaS), users can allocate instances of services (e.g. data storage, data processing, reasoning engines, etc.) or computing capacity for custom applications (Kim et al. 2014). The *connection management* module is responsible for the seamless integration into the various networks (see Figure 9 and Figure 10).

Access Platform

The purpose of the *Access Platform* is to provide users with access to the platform by either using a web application, a mobile application or an open RESTful API.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

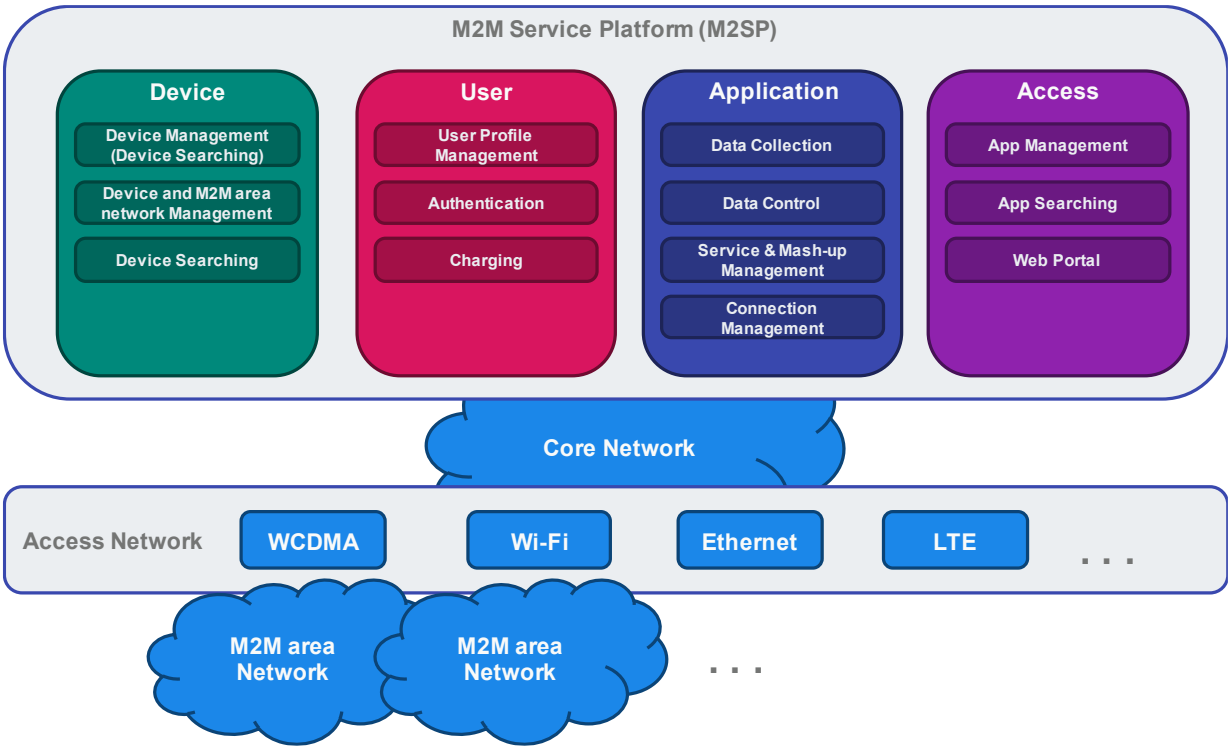


Figure 10: Ideal M2M platform architecture (adapted from Kim et al. 2014)

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

4 Developing the Holistic IoT Architecture Framework

This chapter covers the development process of the *Holistic IoT Architecture Framework*. It begins with a discussion why S2aaS was selected as a baseline in section 4.1. Subsequently, both *IoT Architecture Perspectives* and *IoT Architecture Components* are identified and discussed in section 4.2. Based on the findings and insights gained in section 4.2, a novel *IoT Architecture Component* is developed in section 4.3. The last section of this chapter concludes with the presentation of the *Holistic IoT Architecture Framework*.

4.1 Sensing as a Service as a Baseline

This section introduces the Sensing as a Service (S2aaS) architecture which is used as a baseline for the analysis of existing IoT architecture proposals as well as the development of the holistic IoT architecture framework to be performed in the subsequent sections. Furthermore, this section highlights why S2aaS was selected as the baseline in the first place.

Sensing as a Service is a novel concept first presented by Sheng et al. (2013) to refine existing, proprietary mobile phone sensing applications and propose a generic, reusable and extendable architecture for mobile phone sensing applications. Mobile phones as a sensing platform have become popular due to their widespread availability and the extensive sensor array built in to most of them. Abdelwahab et al. (2016) state that with today´s availability of smartphones and the population density in urban areas very high densities of sensors per square kilometre can be achieved (e.g., it is suggested that the smartphone-sensor-density in London could exceed 14000 sensors per square kilometre, based on an approximation of Abdelwahab et a. al (2016, p.1) , which is based on the assumption that London has 4000 inhabitants per square kilometre while the smartphone penetration in the UK reaches 55%). With today's smartphones being a powerful platform for a variety of applications, outsourcing the collection of data to smartphones is a reasonable concept to quickly collect large amounts of location specific data. However, Sheng et al. (2012) state that mobile phone sensing applications are mainly designed for a single purpose or domain. To collect a variety of data, in diverse contexts or locations and for different "customers" would require smartphone users to install a plethora of different applications to perform sensing tasks. This issue motivated Sheng et al. (2013) to propose a new model for mobile phone sensing. In addition, this new model aims to leverage the advantages of the cloud computing model. Sheng et al. (2013) state that the core component of their model are smartphone users, which can perform both the roles of a content consumer and cloud service. Smartphone users act as a cloud service when they accept and perform sensing tasks and act as content consumers when they issue sensing tasks. Figure 11 illustrates a S2aaS cloud as envisioned by Sheng et al. (2013). Cloud users issue *sensing requests* by using a web application or some other application having access to the cloud´s external API. These *sensing requests* are then pushed to smartphone users who fulfil the sensing task and send the collected data back to a *sensing server* which stores the sensing data. The data can

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

then be sent back to the cloud user who issued the request. During this process the task of issuing sensing tasks is both crucial and complex.

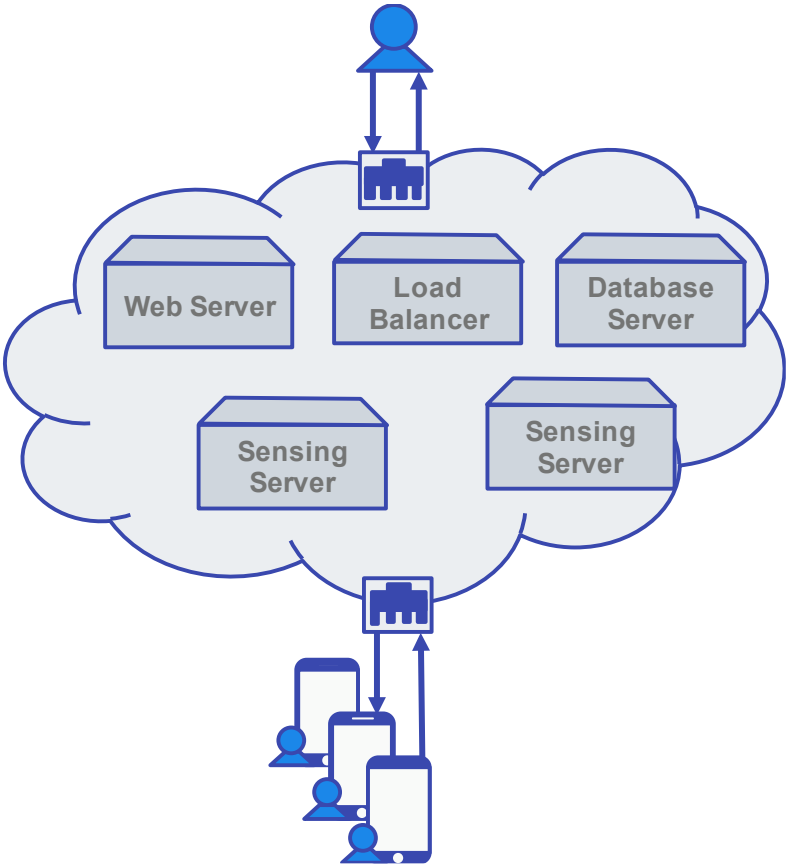


Figure 11: Sensing as a Service Cloud (adapted from Sheng et al. 2013)

The S2aaS architecture considers two paradigms of mobile phone sensing, *participatory* and *opportunistic sensing*. *Participatory sensing* lets smartphone users decide when, where and what data they want to gather. Smartphone users actively participate in the sensing network by accepting or declining sensing tasks or offering their sensing services (Burke et al. 2006). *Opportunistic sensing* aims to involve smartphone users as little as possible when accepting and performing sensing tasks. Thus, this paradigm requires either the smartphone or the sensing server, who issues the sensing requests, to determine when, where and under which circumstances which smartphone user automatically should perform which sensing tasks. Furthermore, the sensing server must consider smartphone user preferences as well (e.g. active hours, locations, battery thresholds, etc.). While this automation is a complex task, it is the least impairing approach, in the sense that smartphones users don't get distracted from their everyday tasks. However, it basically gives the smartphone user only a limited control in selecting sensing tasks (Campbell et al. 2008). Regardless of whether *participatory* or *opportunistic* sensing is used to assign sensing tasks, the underlying *incentive mechanism* must be considered during the assignment of sensing tasks as well (Yang et al. 2012).

Sheng et al. (2013) state that S2aaS heavily relies on smartphone users to perform sensing tasks and transmit their data. When depending on such a crowdsourcing solution smartphone users must be

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

sufficiently motivated or rewarded for performing sensing tasks. Smartphone users use their energy, their devices and their own time to gather the information, hence *incentive mechanisms* are important (Yang et al. 2012). These mechanisms determine how and which smartphone users are selected to perform a sensing task and how much they are rewarded. Yang et al. (2012) mention two paradigms of incentive mechanisms, a *platform centric* and a *user centric incentive mechanism*. In the *platform centric incentive mechanism* approach, a platform (e.g. the sensing server), publishes a sensing task. The sensing task specifies how long, where, when and which kind of data is to be sensed and how much reward is provided. Users in turn try to maximise their reward by being willing to provide a specific amount of time and energy for performing the task, however they also try to minimise their amount of time and energy. The *platform* then tries to find the optimal reward which allows the maximum sensing time. Yang et al. (2012) provide an exemplary mechanism using the *platform centric* approach in the *Stackelberg game*, where the *platform* is the *leader* and *users* are *followers*. In the *user centric incentive mechanism* approach, a set of sensing tasks is published. Each of these tasks again contains information regarding required time and energy. Additionally, each task has a *value* for the platform (e.g. for the sensing server). Users then select tasks and their individual costs for performing the tasks. The platform then tries to maximise the number of tasks to be performed and minimise the required rewards for the tasks (Yang et al. 2012). In the original S2aaS architecture envisioned by Sheng et al. (2013) the *sensing server* is responsible for issuing the sensing tasks while taking the *incentive mechanisms*, energy consumption and *sensing paradigms* into account.

Based on the need for a variety of *incentive mechanisms* and the assumption that a S2aaS cloud should support both *participatory* and *opportunistic* sensing paradigms as well as the requirement that the S2aaS cloud must support a wide array of different sensor devices (e.g. different smartphones with different operating systems and different sensors), Sheng et al. (2013) established a set of general requirements for a S2aaS cloud:

- A S2aaS cloud must be general in the sense that it supports both *opportunistic* and *participatory* sensing paradigms.
- A S2aaS cloud must support a variety of different sensors and mobile phones.
- A S2aaS cloud must be quickly and easily reconfigurable (e.g. to change algorithms incorporated into the architectural elements).
- A S2aaS cloud should minimise energy consumption when issuing sensing tasks.
- A S2aaS cloud must support various incentive mechanisms to foster smartphone user participation.

Beside the already explained requirements to support different *sensing paradigms, incentive mechanisms* and types of sensors and devices, the requirement to be able to *quickly reconfigure* the inner workings of a S2aaS cloud´s component is interesting in particular. In order to comply with this requirement, the behaviour exposed to the “outside” must be meticulously specified (Sheng et al. 2013). One advantage of this approach is that the detailed inner workings can then be omitted. By following

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

this *behaviour driven*[9] approach one could concentrate on the high-level aspects of the architecture while indirectly providing a set of criteria for the inner workings of each component. These implicit criteria for the inner workings of the conceptual components yielded by this approach may be helpful for achieving RO2 (see section 1.2).

Sheng et al. (2013) mention three different high level components of S2aaS, namely *mobile phone users*, *S2aaS cloud* and *cloud users.* The components *mobile phone user* and *cloud user* can be performed by the same entity as mentioned earlier. *Mobile phone users*, or smartphone users, collect data and are recruited by the *S2aaS cloud*. The *S2aaS cloud*, consisting of web applications and frontends, databases, and sensing servers (see Figure 11), manages sensed data and acts upon the *sensing tasks* issued by the *cloud users*. These components are further refined by Perera et al. (2014) (see Figure 12).

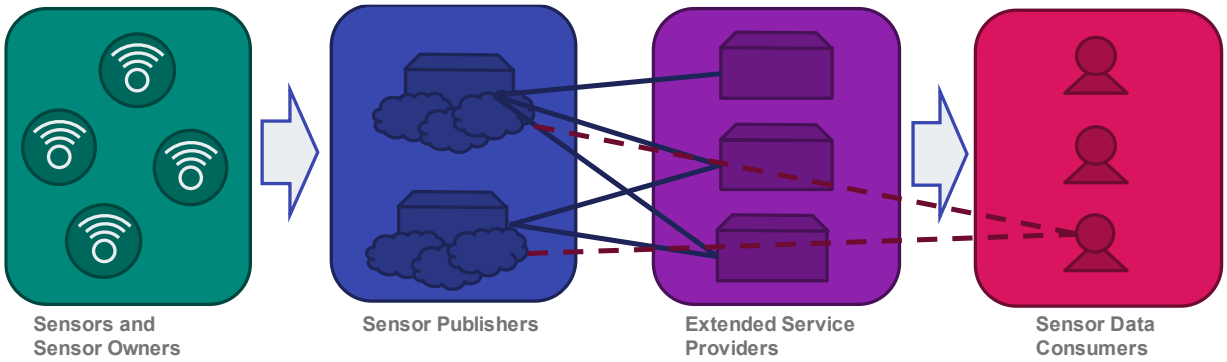


Figure 12: Refined S2aaS architecture (adapted from Perera, Zaslavsky, Liu, et al. 2014)

Perera et al. (2014) distinguish between *sensor owners*, *sensor publishers*, *extended service providers* and *sensor data consumers.* The *sensor owners* in Perera et al.´s (2014) architecture can be mapped to *mobile phone users* in Sheng et al.´s (2013) architecture. The same applies to *sensor data consumers* and *cloud users*. However, Perera et al. (2014) distinguish between *sensor publishers* and *extended service providers* while Sheng et al. (2013) combine these single components into their *S2aaS cloud.* The *sensing server* and the *web server* in the architecture by Sheng et al. (2013) are the sub-components providing the respective functionalities. While Sheng et al. (2013) focus on *mobile phone sensing*, where all sensing tasks are performed by mobile phones or smartphones, Perera at al. (2014) do not limit these tasks to mobile phones but address sensors in general by including the additional component *sensor publisher*. The next paragraphs will provide a brief overview of the components envisioned by Perera et al. (2014) for a S2aaS architecture (see Figure 12).

[9] This *behaviour driven* approach for the development has great similarities with the *behaviour driven development* (BDD) pattern used in software development, as this pattern encourages focussing one the structured specification of the *behaviour* of a software system to be developed (Solis & Wang 2011).

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Sensor owner

Sensor owners have full control over sensors they are owning, thus the name of this component. Essentially this component consists of two elements, the *sensor owners* and the *sensors*. Sensors are devices or *things* that can measure physical phenomena (Perera, Zaslavsky, Christen, et al. 2014). Furthermore, multiple sensors can be attached to the same physical object which in turn can be owned by a *sensor owner*. This relation between physical objects is in line with the description of *things* from section 3.1 and Figure 8. Sensors, regardless of whether attached to or built into objects, are always *owned* by some entity, which is either a person, private organisation or public organisation (Perera, Zaslavsky, Christen, et al. 2014). However, ownership of sensors can change over time. The fact that *sensors* always have an identifiable owner and that they contain information regarding the data they gather, their type etc., supports the conclusion that a *sensor* in Perera et al.´s (2014) definition shares many characteristics with the concept of *spimes* mentioned in section 3.1 (Sterling 2005). Sensors are classified according to the type of their corresponding *owner* (see Figure 13).

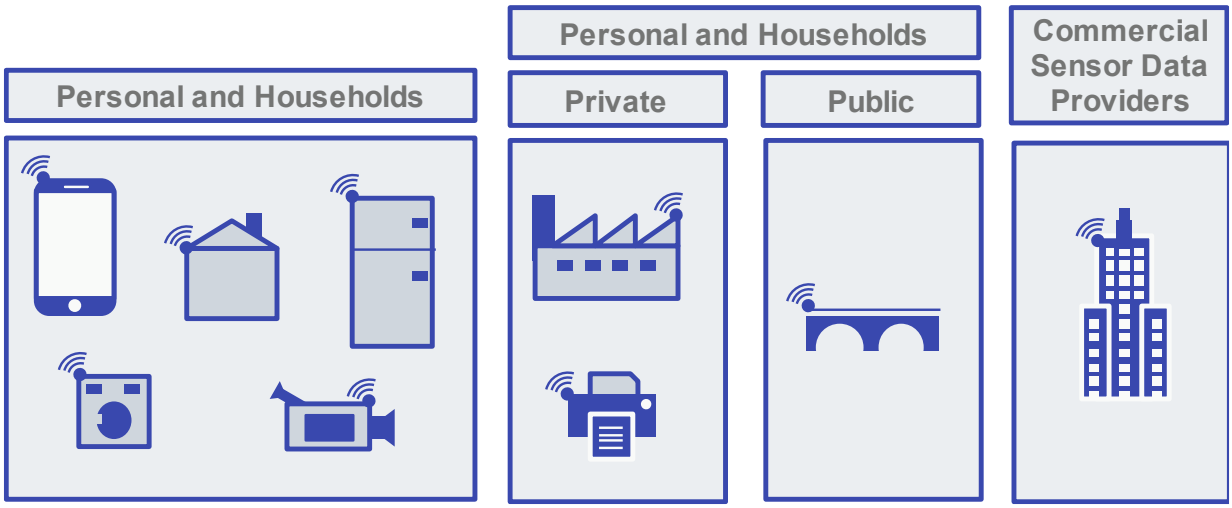


Figure 13: Sensor classification based on ownership (adapted from Perera, Zaslavsky, Christen, et al. 2014)

Sensors owned by private entities, such as natural persons or households, are classified as *personal and household sensors*. These sensors may be built into other devices, such as mobile phones, laptops or other consumer electronic devices. In short, every sensor of which ownership can be attributed to a single person or family can be classified as a *private and household sensor* (Perera, Zaslavsky, Christen, et al. 2014). Before *sensors* become *personal and household sensors* they have very likely been owned by a *private organisation*. All objects belonging to a private organisation with built-in sensors that cannot be attributed to a single person can be classified as a *private organisation sensor*. This includes sensors built in or attached to any other object the private organisation owns (e.g. real estates, offices, factories, products). When a private organisation manufactures products with built-in sensors, the organisation is responsible for these sensors. However, when these products are sold, these sensors change ownership and are considered *personal and household sensors* (Perera, Zaslavsky, Christen, et

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

al. 2014). The new owner of the sensor can then decide if the data of the sensor is published or not. Sensors built into public infrastructures, such as bridges, streets, and other public installations, are classified as *public organisation sensors*. A special category of sensors and sensor owners are the *commercial sensor data providers.* These are private organisations that own, deploy and manage large sensor networks. The gist of the business model of these *commercial sensor data providers* is to gather and sell sensor data on a large scale. Regardless of the type of *sensor owner*, each sensor owner is responsible for the data each of the sensors gathers in his possession (Perera, Zaslavsky, Christen, et al. 2014). Thus, reputation and value of a sensor´s data is related to its respective owner. *Sensor owners* form contracts with *sensor publishers* who offer the sensor´s data on behalf of his owner.

Sensor publisher

Sensor publishers are responsible for detecting sensors, determining the respective *sensor owners* and communicating with them. Furthermore, *sensor publishers* request permission to access and publish the data sensed by the sensors. *Sensor owners* form a contract with *sensor publishers*. Upon forming this contract, the *sensor publisher* gathers metadata (e.g., sensor type, data scheme, information model, owner preferences, availability, etc.). This metadata is later used to determine if a sensor belonging to a *sensor owner* is suitable to perform a specific sensing task (Perera, Zaslavsky, Christen, et al. 2014). When a sensing task is received by the sensor publisher, the task is being forwarded to the respective owner of the sensor. Sensing tasks are usually issued by *extended service providers*. A sensor publisher negotiates between *extended service provider* and *sensor owner*, thus it can ensure privacy and security. The owner of a sensor must not necessarily be known to the entity that issued a sensing task. When a *sensor owner* receives a sensing task, he can decide if he wants to accept the terms of the task. Upon accepting the sensing task, the corresponding *sensor publisher* transmits the sensed data to the requesting *extended service provider* (Perera, Zaslavsky, Christen, et al. 2014).

Extended service provider

Extended service providers provide value added services to *sensor data consumers*. Perera et al. (2014) consider this component of S2aaS as the most “intelligent” one, as it provides a wide variety of different services, where each requires different methods, technologies and approaches to transform the requirements set by *sensor data consumers* to sensing tasks and analyse as well as present the resulting data. One of the main tasks of an *extended service provider* is to formalise the informal sensing requests of *sensor data consumers* into *sensing tasks*. These tasks must be generic and universal, so that each *sensor publisher* can understand and handle the task (Perera, Zaslavsky, Christen, et al. 2014). Additionally, *extended service providers* need to be able to support different *sensing paradigms* as well as *incentive mechanisms*. While Sheng et al. (2013) assigned these tasks to the *sensing server*, these tasks are to be performed by the *extended service provider* in Perera et al.´s (2014) architecture. Additionally, Perera et al. (2014) note that a *sensor* publisher as well as an *extended service provider* can be realised by the same entity (e.g. a business offering both services). An existing example for an

 © 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

extended service provider envisioned by Perera et al. (2014) is the *Azure Data Market* [10]mentioned by Mineraud et al. (2016). On this market, businesses can publish data streams, which essentially are sensors. Access to these data streams can then be purchased for a specified amount of time. However, the *Azure Data Market* does not support creating specific sensing tasks at all. It basically combines the sensor data base of a *sensor publisher* and the service model of an *extended service provider.*

Sensor data consumer

The *Sensor data consumer* is the component which creates sensing tasks. These tasks will be created on a higher level of abstraction (e.g. "Air pollution in a period of time in some city"). These high-level sensor task descriptions are transformed by the *extended service providers* as mentioned before (Perera, Zaslavsky, Christen, et al. 2014). However, *sensor data consumers* can also directly communicate with *sensor publishers* when they have sufficient technical abilities and capabilities. Directly communicating with *sensor publishers* can be a difficult task because the *sensor data consumer* needs to transform his high-level sensor task manually (e.g. measuring "air pollution" requires different sensor types, time periods must be specified and geo-fences must be created to find sensors in a city). Before an entity can become a *sensor data consumer* it must obtain a certificate that certifies their identity. This identity is embedded into the respective sensing task and forwarded to each *sensor owner*, who then can decide if he accepts the task or not. According to Perera et al. (2014) the majority of *sensor data consumers* are mostly governments, businesses and academic or scientific institutions.

Both S2aaS architectures, the mobile sensing focussed approach proposed by Sheng et al. (2013) and the extensions suggested by Perera et al. (2014), specify behaviours and relationships between components. Furthermore, they define responsibilities and tasks for these components and suggest patterns, approaches and concepts (e.g., incentive mechanisms, sensor search techniques, sensing paradigms) that should be embedded into these components. However, they do not restrict or define the inner workings of each component in detail. Both approaches merely require that new concepts must be able to be "plugged in" to the respective components (Sheng et al. 2013). This high level of abstraction used to design the S2aaS architecture is one reason why it was selected as a baseline for further analysis in this thesis. Furthermore, S2aaS is an architecture that focusses on participation and considers both *commercial sensor data providers* as well as single users and the sensor-enabled devices they own. Besides regarding the different characteristics, intentions and requirements of personal-, private-, and public owned sensors, the S2aaS architecture does allow users to assume both the roles of data consumers and producers. Especially the focus on private sensors (e.g. sensors in mobile phones) is notable, because it facilitates dealing with concerns of privacy and trust management.

In conclusion, the S2aaS architecture was selected as a baseline for the further analysis and development performed as part of this thesis, because it is an extendable, high level architecture which focusses on participation of every kind of actor (e.g. data producer or consumer, mediating business

[10] http://datamarket.azure.com/browse/data

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

entities). In addition, it addresses a wide array of concerns ranging from privacy and trust to energy efficient sensor search techniques and economic sensor task scheduling without limiting the architecture through too much detail.

4.2 IoT Architecture Perspectives and Components

This section marks the beginning of the *development step* of the GDC described in section 2.2. It elaborates on both the development of the IoT architecture perspectives and components on which the further development of the holistic IoT architecture framework is based. As a first step, the IoT architecture perspectives, which were identified based on the literature, are presented (see section 4.2.1). As a second step the components and conceptual elements of various IoT architecture proposals are analysed and uniformly described (see section 4.2.2). Both steps rely on a literature review of which the collection method was described in section 2.3 and 2.5.

4.2.1 IoT Architecture Perspectives

In contrast to the IoT visions presented in section 3.1, which distinguish between the *Thing Oriented-*, *Internet Oriented-* and *Semantic Oriented Vision of IoT*, the analysis in this section uses another approach to determine architectural perspectives. The interpretation of IoT Architecture Perspectives is based on a combination of both the *Visions of IoT* presented by Atzori et al. (2010) and the *Generic IoT Architecture* proposed by Khan et al. (2012). The five layer architecture proposed by Khan et al. (2012) consists of the layers *Perception-*, *Network-*, *Middleware-*, *Application-*, and *Business Layer* (see Figure 14b). The *Perception Layer* consists of physical objects, *Things*, and is responsible for gathering environmental data and providing object related information. How the information is gathered, stored, and later transmitted to the *Network Layer* depends on properties of each individual *Thing* (*e.g.* communication technology and protocols)(Khan et al. 2012). The *Network Layer* is responsible for transmitting data that is provided by the *Perception Layer*. The communication and networking technology used in this layer varies and depends on the technologies supported by the *Things* of the *Perception Layer*. Both the *Perception-* and *Network Layer* are closely connected and highly fragmented according to the variety of communication technology. Establishing and maintaining reliable communication with a variety of different types of *Things* using different communication technologies (e.g. Wi-Fi, Bluetooth, etc.) and different protocols (e.g. MQTT[11], CoAP[12]) is a challenging task as highlighted by Zachariah et al. (2015). The *Middleware Layer* is built on top of the *Network Layer*. The main task of this layer is to provide uniform communication end-points used by the *Application Layer* by combining the fragments of the *Network Layer* (Mashal et al. 2015). Besides these mediating tasks,

[11] Message Queue Telemetry Transport (MQTT) is a lightweight publish-subscribe communication protocol especially suited for constrained and unreliable environments (Stanford-Clark & Nipper 2017).

[12] Constrained Application Protocol (CoAP) a lightweight RESTful based communication protocol designed for the use on highly constrained devices (ARM Limited 2017).

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

some proposals for IoT architectures and models do not only use this layer to cover mediating tasks but also to represent the capability to store data, process and normalise data streams, to reason over data, and to manage services (Mashal et al. 2015; Khan et al. 2012). However, some of these requirements for middlewares in the context of IoT can be considered inappropriate when referring to the original concept and definition of middlewares (Naur & Randell 1968; Emmerich et al. 2008). Originally, the concept of a middleware was introduced in Naur & Randell (1968) as an additional abstraction layer to decouple applications from underlying operating system and thus to manage heterogeneity issues (Emmerich et al. 2008). Hence the requirement to store data in databases, as stated by Mashal et al. (2015) inter alia, may require further explanation. The applications which are part of the *Application* Layer, consume the normalised web services and API endpoints exposed by the *Middleware Layer* to provide actual IoT applications that can be used by users or other services (Mashal et al. 2015). IoT applications in general revolve around data-acquisition, -integration, and -representation as well as autonomous actions based on the gathered data. The *Business Layer* is on top of the *Application Layer* and manages an IoT system, its related services and business models (Khan et al. 2012; Miao Wu et al. 2010). Additionally, Wu et al. (2010) consider the *Business Layer* as a driver for the development of IoT applications. They argue that the development of successful business models facilitates the advancement of IoT related technologies. In conclusion, the responsibilities of each layer of the five layer IoT architecture is as follows (based on Miao Wu et al. 2010; Khan et al. 2012; Mashal et al. 2015).

- **Perception Layer**, gathers environmental data and annotates/ identifies physical objects with digital properties.
- **Network Layer**, transports information through various channels with various technologies.
- **Middleware Layer**, normalises heterogeneous data and exposes uniform interfaces.
- **Application Layer**, provides domain specific applications which use sensed data.
- **Business Layer**, drives application development and defines “value” for information obtained through processed and sensed data.

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

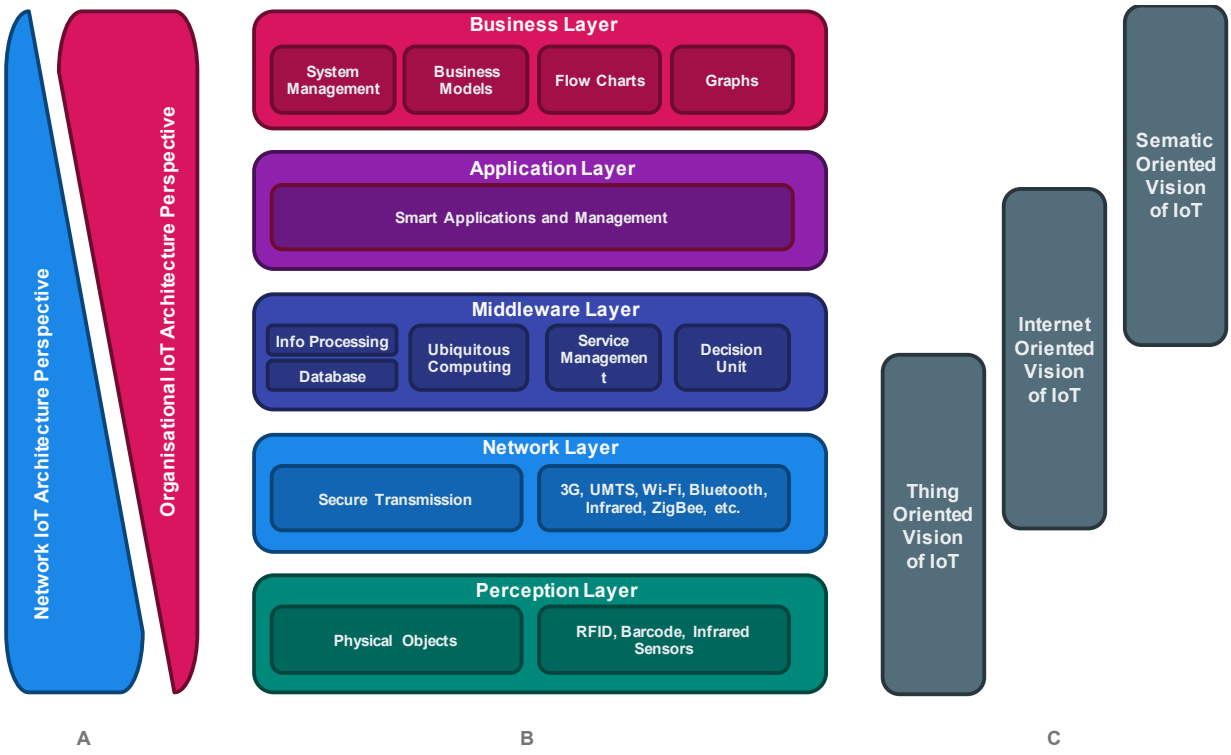


Figure 14: IoT Architecture Perspectives combined with Visions of IoT and five layer IoT architecture (own illustration, b) based on Khan et al. 2012; c) based on Atzori et al. 2010)

Based on the responsibilities and tasks assigned to each layer and the focus of each IoT vision mentioned in section 3.1 a mapping between the layers and visions can be performed. With the *Perception Layer* consisting of a multitude of different sensory devices and dealing with the collection of environmental data or identification of real world objects, its aspects resemble the focus of the *Thing Oriented Vision of IoT*. As mentioned, this vision focusses on *Things* as the basic building blocks of IoT (Atzori et al. 2010). Providing addressing schemes, enabling *Things* to transmit data and to enrich everyday objects with digital properties are additional topics the *Thing Oriented Vision of IoT* addresses. However, the *Thing Oriented Vision of IoT* additionally aims to deal with heterogeneity and networking issues. As illustrated in Figure 7, both the *Thing Oriented*- as well as the *Internet Oriented Vision of IoT* share the common topics of enabling *things* to communicate with each other (Mazhelis et al. 2013). With the heterogeneity of *Things* and the resulting variety of supported communication technologies and protocols, the *Thing Oriented Vision of IoT* additionally shares topics, tasks and responsibilities with the *Network Layer*. The *Internet Oriented Vision of IoT* aims to incorporate the novel requirements and technologies into the current internet. It mainly addresses communication technology and protocol issues, e.g. it aims to facilitate the use of 6LoWPAN as a communication technology for constrained devices. With the *Network Layer* being responsible for ensuring the transportation of data from the *Perception Layer* / the *Things* to the *Middleware Layer,* both the *Internet Oriented Vision of IoT* and the *Network Layer* share common tasks and goals. Additionally, the *Internet Oriented Vision of IoT* partly addresses issues or responsibilities of the *Middleware Layer*, as both aim to cope with heterogeneity (e.g. with the gateway problem mentioned in section 1.1 and 3.1). The *Semantic Oriented Vision of IoT* also tries to specifically

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

address heterogeneity issues, however it utilises a different approach. While the *Internet Oriented Vision of IoT* suggests using the existing internet architecture, modifying it where necessary (e.g. 6LoWPAN), and using it as a standardised communication approach for all new devices and scenarios, the *Semantic Oriented Vision of IoT* suggests using semantic technologies to deal with heterogeneity issues. For example, this means that this vision of IoT proposes to use "semantic adapter components" as suggested by Katasonov et al. (2008). In essence, instead of relying on syntactical standards to ensure heterogeneous communication and interaction, the semantic representations of devices, data and services are used to dynamically create interfaces to access the respective devices, data and services (Katasonov et al. 2008). The *Semantic Oriented Vision of IoT* can additionally be assigned to the *Application Layer*, especially the intention of this vision to promote the development and use of context aware applications and reasoning over data is relevant for this layer. The relations between each vision of IoT and the respective layers of the generic IoT architecture are illustrated in Figure 14b and 14c.

Based on the visions of IoT and the layers of the generic IoT architecture, an additional type of perspective became apparent during the analysis of IoT architecture proposals presented in the literature obtained through the literature search described in section 2.3. The architecture proposals regarded in this thesis can either be assigned to the *Organisational IoT Architecture Perspective* or the *Network IoT Architecture Perspective* (see Figure 14a).

The *Network IoT Architecture Perspective* is focussed on establishing network communication between devices or things. Devices or things are considered as network nodes in this perspective and are decoupled from other aspects like ownership, identity, privacy, security, or reputation. Architecture proposals implicitly using this perspective address issues on the *Perception-, Network-* and on the *Middleware Layer* and are likely using either the *Thing Oriented-* or *Internet Oriented Vision of IoT.* Zachariah et al. (2015) discuss the "gateway problem", which is a heterogeneity issue of IoT. They state that most constrained devices rely on low power wireless communication (e.g. BLE) instead of using Wi-Fi or other more well connected technologies (Zachariah et al. 2015). These constrained devices require an application layer gateway to communicate with the internet (e.g. a smartwatch needs to be connected to a smartphone to be able to download updates from the internet). This connection between a constrained device and a gateway is usually achieved by using a device specific proprietary application. For each different constrained device a new application must be installed on the gateway, which is usually a smartphone, in order to connect the constrained device to the internet (Zachariah et al. 2015). Zachariah et al. (2015) define this multitude of different, proprietary gateways with a narrow application context as the *IoT Gateway Problem*.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

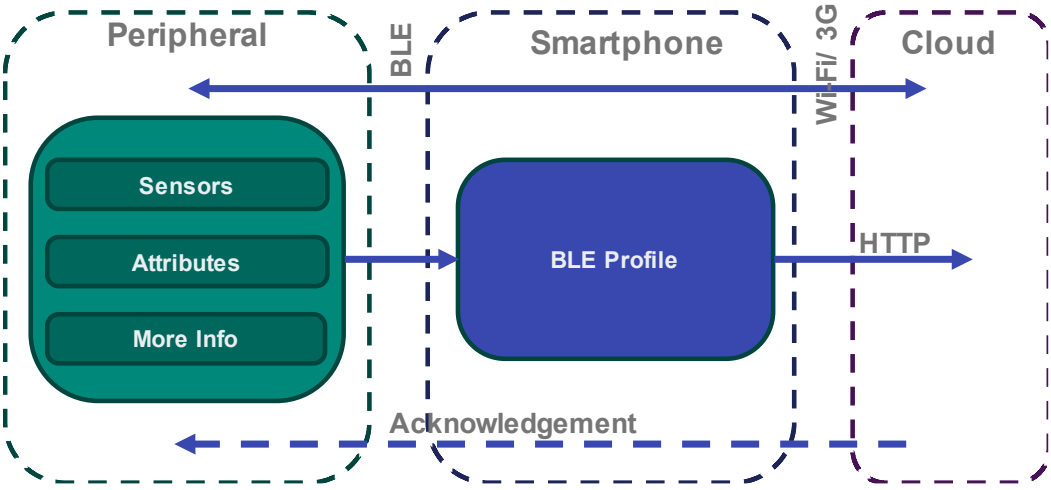


Figure 15: IoT Smartphone Gateway Architecture (adapted from Zachariah et al. 2015)

To solve this problem, Zachariah et al. (2015) propose a smartphone-centric architecture (see Figure 15). Constrained devices should be connected to the internet by using a smartphone as a general-purpose gateway. Constrained devices communicate with the smartphone via BLE, while the smartphone uses cellular networks to connect to the internet. Zachariah et al. (2015) suggest two possible communication mechanisms to be used in this architecture. The first mechanism considers the smartphone as a temporary IPv6 router, which forwards all communication to the connected IPv6 enabled constrained devices. This mechanism allows a direct communication between IPv6 enabled constrained devices and the internet. However, for these constrained devices to be directly addressable, they need to be able to run a full IPv6 stack which requires relatively much computational power (Zachariah et al. 2015). However, 6LoWPAN, as a simplified approach for IPv6 communication, is deemed to be a suitable solution for enabling constrained devices for IPv6 (Hui & Corporation 2009). Alternatively, Zachariah et al. (2015) suggest an approach based on the “web of things” paradigm, that aims to use device metadata to expose RESTful endpoints. In this approach, the constrained devices send additional metadata (e.g. which kind of data they provide, their location, the address where sensed data is to be sent to, etc.) to the smartphone which currently acts as the gateway for the constrained device. The gateway then transforms this metadata into a RESTful http endpoint. When the metadata of the constrained device contains an address where the sensed data should be send to, the gateway may send a HTTP request to the specified address containing the information provided by the constrained device (Zachariah et al. 2015). The architecture proposed by Zachariah et al. (2015) addresses communication issues at the *Perception-*, *Network-,* and *Middleware Layer*. This architecture aims to empower smartphones to act as gateways for constrained devices. Due to the mobility of smartphones the connection between a constrained device and a gateway cannot be permanent, thus ensuring and acknowledging that data was transmitted successfully remains an issue (Zachariah et al. 2015). The architecture proposed by Zachariah et al. (2015) does not consider ownership issues of constrained devices or things. They suggest that mobile smartphone gateways connect to every constrained device within range of the gateway and forward the data provided by the respective device,

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

thus one can infer that gateways and constrained devices in Zachariah et al.´s (2015) proposal can be interpreted as network nodes. Additionally, they suggest the usage of existing internet technologies (e.g. BLE, IPv6, 6LoWPAN) to enable heterogeneous connectivity between constrained devices and things. Hence, their approach can be assigned to the *Internet Oriented Vision of IoT.*

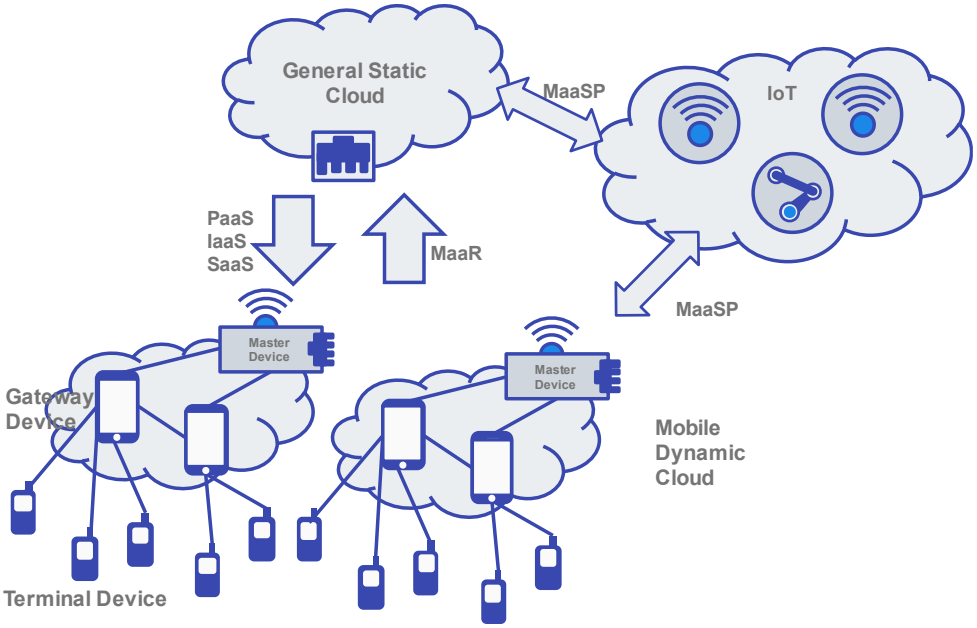


Figure 16: Heterogeneous Network Architecture (adapted and simplified from Jo et al. 2015)

The architecture proposed by Jo et al. (2015) (see Figure 16) follows a similar approach. They present an architecture focussed on heterogeneous device-to-device communication with four types of devices and three modes of usage, with each type of device being considered as a personal item such as wearables, mobile phones, smartphones, tablets or personal computers. *Terminal-* and *Sensing Devices* are the most constrained devices in the architecture proposed by Jo et al. (2015). *Terminal Devices* can only act as *Mobile as a Service Consumers,* which means that they cannot share their own resources but depend on the shared resources, which are offered as a service, of other devices in the architecture. The services these type of device consumes are provided by the *General Static Cloud*, which is a network of high performance data centres offering various kinds of services. In order to consume a service offered by the *General Static Cloud*, a *Terminal Device* communicates with a *Gateway Device*. This type of device operates as a *Mobile Device as a Service Broker* mode, which in the context of Jo et al.´s (2015) means that this type transfers data from *Terminal Devices* to *Master Devices* and vice versa. *Sensing Devices* are similar to *Terminal Devices* but are specialized in performing sensing tasks. However, in contrast to the *Terminal Device*, this device operates in the *Mobile as a Service Provider* mode and thus provides only the sensed data and is not able to communicate or collaborate otherwise. These sensing services are provided to the *Mobile Dynamic Cloud*, which encompasses all mobile devices and is introduced to offload and decrease network traffic to the *General Static Cloud*. While the *General Static Cloud* relies on large scale data centres, the *Mobile Dynamic Cloud* utilises computing capacities near

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

the network edge. The fourth type of device, the *Master Device* operates in the *Mobile as a Service Representer* mode (Jo et al. 2015). This type of device has the highest computing capacity and supports a wide array of different wireless communication technologies and protocols. The *Master Devices* are directly connected to the *Gateway Devices*, the *General Static Cloud* and to other *Master Devices*. They solely represent their respective *cloudlet* which consist of all *Gateway Devices* and *Terminal Devices* and offload any task they cannot perform on their own to services in the *General Static Cloud*. Likewise, they represent the services provided by *Sensing Devices* and enable all other types of devices access to sensed data. Jo et al. (2015) additionally suggest that each type of device maintains a backup connection to the *General Static Cloud* in case no reliable connection through the *Mobile Dynamic Cloud* can be achieved. By heavily relying on computing capacities near the network edge, this architecture is supposed to scale well with increasing amounts of devices. Hence, the architecture proposed by Jo et al. (2015) provides a possible solution for scalability issues of IoT. Additionally, Jo et al. (2015) suggest that each type device uses either Bluetooth, cellular networks or Wi-Fi in combination with IPv6 for communication purposes. This architecture is specifically designed to address scalability and heterogeneity issues by relying on standardised communication based on existing internet technologies. Hence, it can be assigned to the *Internet Oriented Vision of IoT* as well as addresses issues relating to the *Perception-*, *Network-*, and *Middleware Layer*.

Both architectures presented by Zachariah et al. (2015) and Jo et al. (2015) can be assigned to the *Network IoT Architecture Perspective* as both focus on “low level” network issues of IoT, and to the *Internet Oriented Vision of IoT* and address issues on the *Perception-, Network-* or *Middleware Layer* of the *Generic IoT Architecture* (see Figure 14). It must be noted that the architectures presented previously and subsequently are only described with regard to their respective elements and their corresponding relations as defined in section 2.4.

The *Organisational IoT Architecture Perspective* focusses on the organisational aspects of an architecture and specifically addresses organisational relationships between each element, e.g. the stakeholders of an element or the characteristics of the relationship between two components. In contrast to the *Network IoT Architecture Perspective* it focusses on the organisational flow of data and not on network traffic or routing issues. The reputation of the data producers and the value of data in context of its intended use are important, while the format of the data or the required communication technologies are of secondary importance. The *Organisational IoT Architecture Perspective* interprets *Things* as “business entities” and focusses on ownership, security, identity and reputation rather than communication technologies or network communication aspects. Architecture proposals that implicitly refer to the *Organisational IoT Architecture Perspective* address issues on the *Middleware-*, *Application-*, or on the *Business Layer* of the *Generic IoT Architecture.* Furthermore, these proposals tend to use either the *Internet Oriented-* or *Semantic Oriented Vision of IoT*.

The original architecture proposals for S2aaS presented by Sheng et al. (2013) and Perera et al. (2014) apply the *Organisational IoT Architecture Perspective*. Both proposals are specifically concerned with the organisational relationships between the different components and intentionally omit details on

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

inner workings of the components as well as technical aspects (see section 4.1). Abdelwahab et al. (2015) as well as Chang et al. (2015) follow the same approach. The *Cloud of Things Architecture for S2aaS* presented by Abdelwahab et al. (2016) consists of four components (see Figure 17). *Cloud Users* create sensing tasks that are managed and transformed by *First Tier Clouds*. Sensing tasks contain information regarding the type of data to be collected, the area to be covered and various other aspects depending on the sensing tasks domain (Abdelwahab et al. 2015). *First Tier* Clouds provide unified interfaces for *Cloud* Users and abstract as much complexity as possible (e.g. a user does not have to specify sensor types or deal with scheduling or discovering sensors). After being converted into *Sensing Task Requests*, which are merely a formalisation of the task created by a *Cloud User*, the *First Tier Cloud* issues this request to their *Cloud Agents*. The *Cloud Agents* are highly connected components, vary in terms of computing capacity (e.g. a *Cloud Agent* can be a server-cluster, a normal personal computer or a smartphone) and create as well as manage *Virtual Sensor Networks* based on the respective *Sensing Tasks* they currently perform (Abdelwahab et al. 2015). The *Cloud Agents* presented in Abdelwahab et al.´s (2015) architecture proposal share similarities with the *Smartphone Gateways* and the *Master Devices* presented by Zachariah et al. (2015) and Jo et al. (2015), respectively. However, whereas the latter focus on enabling and managing network connections between different components of their respective architecture, Abdelwahab et al. (2015) focusses on patterns for resource or sensor discovery and allocation of sensor networks, regardless of any communication technology and protocol, or network structure. *The Cloud Agent* in Abdelwahab et al.´s (2015) proposal are connected to a variety of individual sensing devices and create a virtual network based on the *Sensing Request* they are currently performing based on these connected sensing devices. The data gathered by these virtual networks is then sent to the *Cloud User* via the *First Tier Cloud*.

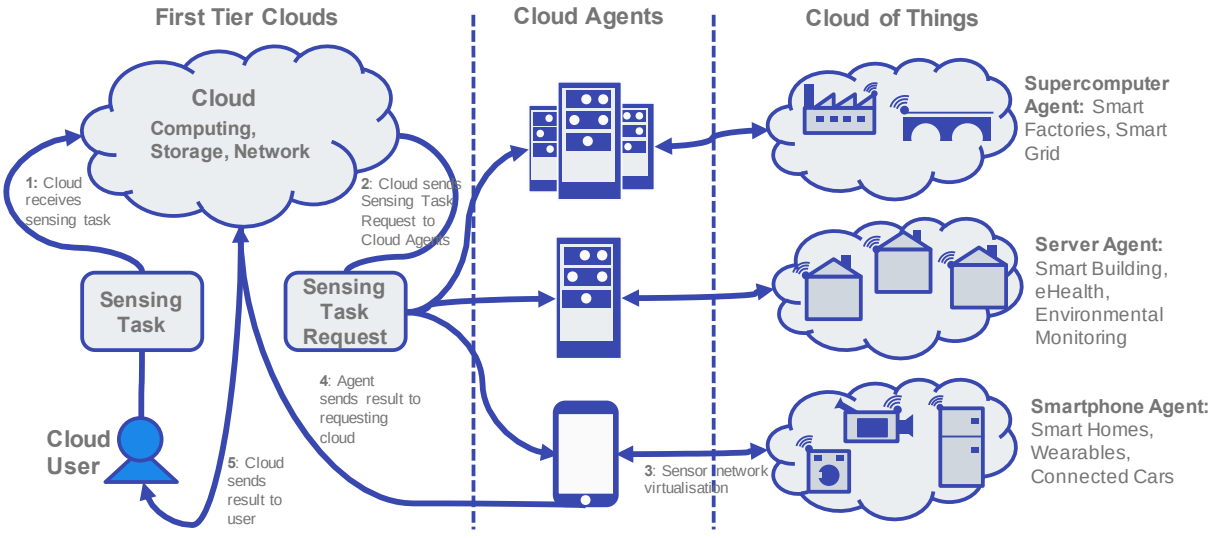


Figure 17: Cloud of Things Architecture for S2aaS (adapted from Abdelwahab et al. 2015)

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

The architecture proposed by Chang et al. (2015) follows a similar approach (see Figure 18). In the *Mobile Device as a Sensory Service Mediation* framework, *Clients* issue sensing requests to *Mobile Hosts* that provide access to sensor networks. The *Mobile Hosts* are either discovered by direct peer-to-peer communication or via *Discovery Servers* which manage a database of available *Mobile Hosts* along with metadata describing the services provided by each *Mobile Host* (Chii Chang et al. 2015). The *Mobile Host* provides three modes of service access: one-time sensing, real-time sensing and periodical sensing. The data collected through the sensor networks managed by the *Mobile Host* is stored in the *Utility Cloud Service*, which acts as an on-demand data storage. Clients who have issued *Sensing Tasks* are granted access to the respective *Utility Cloud Service* instances by the *Mobile Hosts*. When multiple *Clients* have issued similar *Sensing Tasks* to a *Mobile Host*, the data is stored on a single *Utility Cloud Service* instance and all *Clients* are granted access to the data. This reduces network load and energy consumption (Chii Chang et al. 2015). Similar to the proposal provided by Abdelwahab et al. (2015), this proposal focusses on the organisational relationships, on resource discovery as well as data and access flow rather than network and communication issues. As both architecture proposals focus on the either the *Application-* or the *Middleware-Layer* of the *Generic IoT Architecture* they can be assigned to the *Organisational IoT Architecture Perspective.*

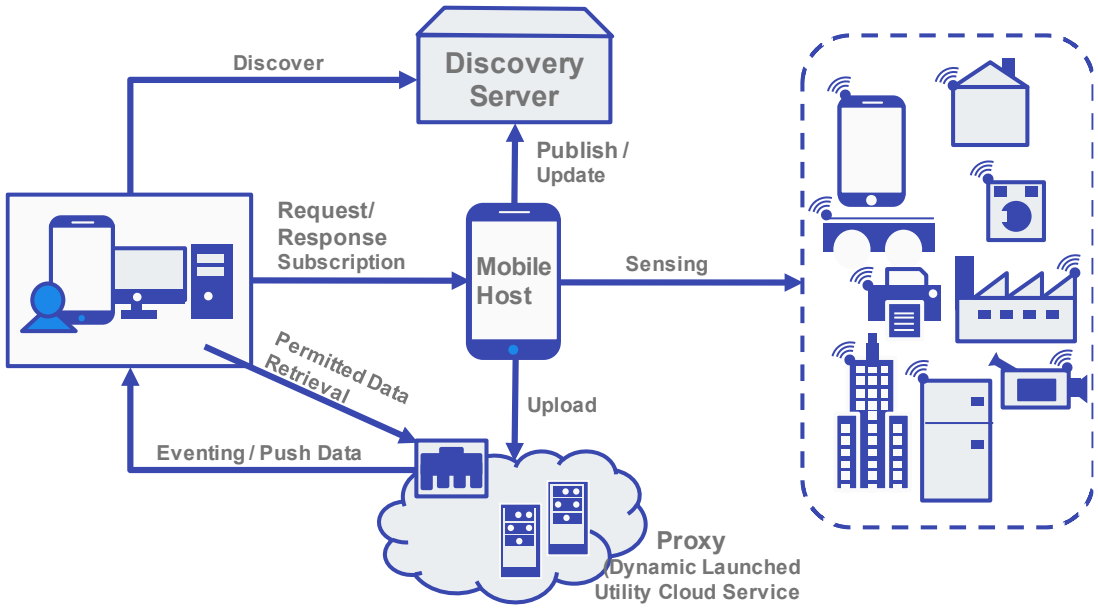


Figure 18: Mobile Device as a Sensory Service Mediation (adapted from Chii Chang et al. 2015)

Mizouni and El Barachi (2013) propose a business model for *Mobile Phone Sensing as a Service* that illustrates different modes of usage and payment models for mobile phone users (see Figure 19). This business model is an extension of the currently used mobile internet (3G/4G) business model (Mizouni & El Barachi 2013). At its core, the *Mobile Network Operator* provides network services which are consumed and paid by *End-Users*. *Value Added Service Providers* then provide additional services which are billed via the *Mobile Network Operator* and can be used by the *End Users*. In order to provide sensing services, Mizouni and El Barachi (2013) extend the core model with two additional components, the

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایـســـازی علـــمی»

Mobile Sensing Terminal Operator and the *Mobile Sensing Network*. Entities belonging to the *Mobile Sensing Network* are sensors which capture environmental data and are assigned either to specific users (persons), private or public organisations or communities. This categorisation is similar to the classification of sensors related to their respective *Sensor Owners* of the S2aaS architecture described by Perera et al. (2014) (see Figure 13 and section 4.1).

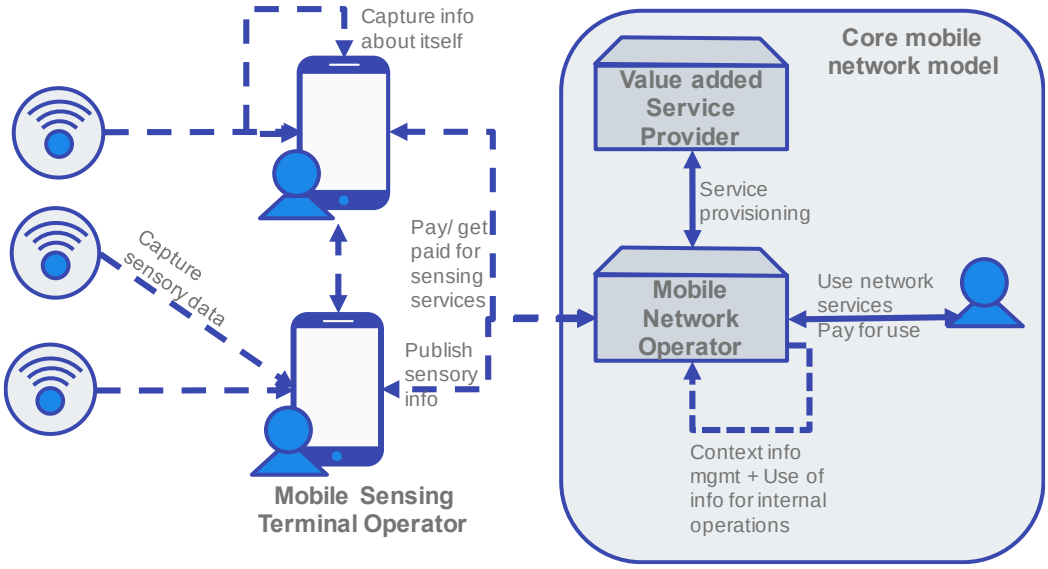


Figure 19: Mobile Phone Sensing as a Service Business Model (adapted from Mizouni & El Barachi 2013)

The data captured by the entities of the *Mobile Sensing Network* is collected by a special type of *End-User*, the *Mobile Sensing Terminal Operator*. Besides being a normal mobile network user, the *Mobile Sensing Terminal Operator* is able to collect data from the entities of the *Mobile Sensing Network* and publish the data to the *Mobile Network Operator* (Mizouni & El Barachi 2013). The *Mobile Sensing Terminal Operator* performs this task proactively, without a sensing task being issued to him. In contrast to the previously presented architectures and frameworks, the business model proposed by Mizouni and El Barachi (2013) relies on mobile phone users continuously collecting and publishing data on their own.

When a *Mobile Sensing Terminal Operator* collects the data provided by a *Mobile Sensing Network* entity, he pays for having access to the data of the entity. Upon publishing this data to the *Mobile Network Operator,* the *Mobile Sensing Terminal Operator* gets paid depending on the value of the collected data (Mizouni & El Barachi 2013). Thus, this approach relies on the mobile phone user to proactively gather environmental data rather than reacting to sensing tasks. The business model proposed by Mizouni and El Barachi (2013) distinguishes between four charging models for *Mobile Sensing Terminal Operators.* The first model considers the operator to be operating as a self-representative, thus he benefits from the sensed data and pays the entity owner of the *Mobile Sensing Network* entity for the usage of the data. The second model considers the *Mobile Sensing Terminal Operator* as a peer-representative who acts on behalf of other users, collects data for them and in return

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

gets paid for his services by the other *End-Users*. The third charging model considers the operator as a representative of the *Mobile Network Operator*. The *Mobile Network Operator* benefits from the sensed data and in return pays the *Mobile Sensing Terminal Operator.* In the last charging model, the operator either acts alone or collaborates with other operators to collect data in the first place; this includes paying the owners of sensors. When the data is accessed, the individual operator or the group is paid for their services. This proposal focusses primarily on the *Business Layer* of the *Generic IoT Architecture*. Additionally, it aims to use mobile networks as existing technologies to enable IoT sensing services, thus it can also be assigned to the *Internet Oriented Vision of IoT*.

All architectures, frameworks and models presented in relation to the *Organisational IoT Architecture Perspective* can either be assigned to the *Business-, Application-* or *Middleware Layer* of the *Generic IoT Architecture*. Furthermore, they are either using the *Internet Oriented* or *Semantic Oriented Vision of IoT*. In conclusion, two *IoT Architecture Perspectives*, namely the *Network IoT Architecture Perspective* and the *Organisational IoT Architecture Perspective*, have been identified and explained. Additionally, these new perspectives were combined with the *Visions of IoT* and embedded into the *Generic Architecture of IoT* (see Figure 14). These tasks where performed in order to answer research question RQ1.2 (see section 1.2).

4.2.2 IoT Architecture Components

Based on the *Generic IoT Architecture*, proposed by Khan et al. (2012), described in section 4.2.1, and illustrated in Figure 14b, the individual components of IoT architecture proposals are analysed in order to work out common concepts of each component´s tasks, responsibilities and requirements. Therefore, each IoT architecture proposal is examined and the descriptions, requirements, tasks and intentions of each component are categorised based on the *Generic IoT Architecture's* layers. Ideally, each component is described with each layer of the *Generic IoT Architecture* in mind, which would mean in conclusion that the component is holistically described. A holistic IoT architecture component description will consider the *Business-*, *Application-*, *Middleware-*, *Network-* and *Perception Layer* of the *Generic IoT Architecture*. However, depending on the component not all layers are necessary for describing all aspects of an IoT architecture component, e.g. the *Mobile Network Operator* in Mizouni and Barachi´s (2013) architecture proposal does not require a description on the *Perception Layer* because it does not directly interact with the environment to sense data. The IoT architectures which are regarded as part of this thesis are shown in Table 1. The components are grouped with the architecture they belong to. Each component is described on different layers, whereas the *Business Layer* is abbreviated with the character B, the *Application Layer* with the character A and so forth.

The descriptions for each component are extracted from the articles proposing the respective architecture, normalised, and assigned to one of the layers. For example, Perera et al. (2014) and Zachariah et al. (2015) i.a. suggest that *Sensor Owners* and *Smartphone* (-owners) must be incentivised to perform their assigned task and that these components usually seek compensations for the services they provide. Hence, one part of the common, normalised description of these components states that

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

they want to be compensated for providing access to sensing data. When reviewing the extracted component descriptions (see Table 1) it becomes apparent that all IoT architecture proposals share several common elements or components. These common components are presented in the next paragraphs.

Consumer

All architecture proposals considered in this thesis mention the component *Consumer* or *Data Consumer*, respectively, which is essentially characterised by its demand for data (Perera, Zaslavsky, Christen, et al. 2014; Mizouni & El Barachi 2013; Abdelwahab et al. 2015). Regarding the *Business* Layer, Consumers are always legal persons who have an interest in data and its potential information or knowledge and are generally willing to compensate for the data and services they consume (Mizouni & El Barachi 2013; Jo et al. 2015). Besides the demand for data, Jo et al. (2015) mention that *Consumers* are also interested in offloading computational tasks onto cloud infrastructures. *Consumers* generally do not provide any applications, services or are involved in managing communications between different devices. Thus, the *Application-, Middleware-*, and *Perception Layer* are not considered in the architecture proposals analysed in this thesis. However, *Consumers* either use web applications offered by *Service Providers* to obtain sensing data or directly connect to components which provide sensing data. For example, Perera et al. (2014) state that the *Sensor Data Consumers* can directly access the *Sensors* maintained by a *Sensor Publisher*. However, this approach requires the *Sensor Data Consumer* to strictly formalise his sensing requests. The architecture proposed by Chang et al. (2015) allows the *Client* to directly issue sensing requests to the *Mobile Host* in a peer-to-peer connection. Depending on the communication interfaces, technologies and protocols provided by the *Mobile Host* or the directly accessed *Thing*, which can be as powerful as a smartphone or a highly constrained device, the *Consumer* must support compatible communication interfaces, technologies and interfaces. Thus, the *Network Layer* is addressed in some proposals when describing the *Consumer* component (e.g. *Consumers* must support BLE or Wi-Fi to connect to *Things* in the vicinity) (Perera, Zaslavsky, Christen, et al. 2014; Chii Chang et al. 2015). In conclusion, *Consumers* are entities that are mainly interested in data or consumption of services, are willing to provide incentives for the data or services, and regularly use services provided by *Service Providers* to issue sensing requests or rarely directly access *Things* to obtain the desired data.

Thing

Every architecture proposal presented in Table 1 contains components that either resemble a single *Thing* or a network of *Things*. These components can be virtual services or physical devices (Abdelwahab et al. 2015). On the *Business Layer*, *Things* act as service providers, they provide access to their sensing or data gathering capabilities. In return, *Things* can, but need not, demand compensation for the services they provide. Additionally, the access to a *Thing´s* services must follow certain rules that the *Owner* of the *Thing* might define. Such rules can specify access schedules, energy consumption thresholds, restrict the data that can be accessed (e.g. no private data can be accessed, location must be anonymised) and define what kind of compensation is required to access the services (Jo et al. 2015;

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Mizouni & El Barachi 2013; Perera, Zaslavsky, Christen, et al. 2014). Some architecture proposals consider *Things* as personal devices (e.g. smartphones) (Chii Chang et al. 2015), while others consider *Things* as highly constrained devices that have little computational power or storage capabilities. However, every IoT architecture proposal either explicitly or implicitly mentions that *Things* are always managed by an *Owner*, who defines access rules, compensations and other properties. On the *Perception Layer*, *Things* merely have unique sensing capabilities which no other component of the respective architecture has. Due to the constrained nature of most *Things*, descriptions addressing the *Network Layer* describe the limited communication technologies and protocols of *Things*. The architectures considered in this analysis suggest that highly constrained *Things* should be connected via BLE, 6LoWPAN or Wi-Fi (Jo et al. 2015; Zachariah et al. 2015). Less constrained *Things*, such as smartphones, can additionally be connected via cellular networks (Abdelwahab et al. 2015; Chii Chang et al. 2015). In conclusion, *Things* provide unique sensing capabilities within the respective architecture, are connected via limited communication channels and are mainly defined or shaped (e.g. properties, metadata, access rules) by their respective *Owner*.

Owner

Although the component *Owner* is included in every IoT architecture proposal considered in this thesis, it is rarely explicitly defined as an individual component. The reasons for this lack of a differentiated view on the *Owner* component is twofold. At first, due to the origin of *S2aaS* in *Mobile Phone Sensing, Owner* and *Thing* were basically considered as the same component (Sheng et al. 2013). In a traditional *Mobile Phone Sensing* approach, a smartphone is considered as a *Thing* which provides unique sensing capabilities. Additionally, the smartphone is considered as a personal device, which is owned by a person who can immediately decide if a sensing task is performed (participatory sensing) or who can define a set of rules for automatically performing sensing tasks (opportunistic sensing) (Yang et al. 2012; Sheng et al. 2013). The second reason for the lacking differentiation between *Things* and their respective *Owners* relates to the respective *IoT Architecture Perspective* of the corresponding IoT architecture. For example, the architecture proposals of Jo et al. (2015) and Zachariah et al. (2015), which were assigned to the *Network IoT Architecture Perspective* (see section 4.2.1), assume that access rules and expected compensations are provided by the *Thing*, regardless who defines these rules. Hence, Jo et al. (2015) and Zachariah et al. (2015) abstract and thus elude the ownership relation between *Thing* and *Owner* and focus only on the metadata (e.g. access rules, data provided, etc.) exposed by the *Things* and how to establish connections to these *Things*. On the *Business Layer* the *Owner* wants to be compensated for the services his *Things* offer. Additionally, the *Owner* wants to specify access rules and restrictions for the *Things* he owns. Access rules and restrictions include specifications on which kind of data and in what detail the data is provided by a *Thing*. For example, an *Owner* might not want to expose the location of a device and requires that the location is either anonymised or not exposed at all (Krause et al. 2008). In conclusion, the *Owner* is the main stakeholder of a *Thing* and imposes his requirements and specifications onto the *Things* he owns.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Service Provider

The main purpose of this component across all architecture proposals is the provision of value added services to *Consumers*. Individual *Service Providers* can specialise in providing domain specific services (e.g. weather information, industrial sensor networks, etc.), however the gist of each service is the discovery of *Things* and the collection, storage, transformation, analysis and presentation of data (Perera, Zaslavsky, Christen, et al. 2014; Abdelwahab et al. 2015). To be able to provide these services, most IoT architecture proposals consider this component as the most intelligent (Perera, Zaslavsky, Christen, et al. 2014) component which is very well connected and has vast computing capabilities (Jo et al. 2015). In general, *Service Providers* transform high level and informal sensing tasks made by *Consumers* into formalised *Sensing Requests* and issue these requests to available and matching *Things* via an arbitrary infrastructure. *Consumers* must provide compensation for using the services offered by *Service Providers*. The *Service Provider* in turn provides compensation for accessing the data provided by the involved *Things*. Furthermore, the *Service Provider* can also be interested in collecting or issuing *Sensing Tasks* on his own (Mizouni & El Barachi 2013). Besides the previous descriptions on the *Business Layer*, the *Application Layer* of *Service Providers* is also addressed in the literature on IoT architecture proposals. To fulfil their respective *Business Layer* oriented tasks, *Service Providers* offer a variety of applications that are used by *Consumers*. A *Service Provider* usually offers a web application that allows *Consumers* to easily create *Sensing Tasks*. Thus, this application aims to support the main business task of the *Service Provider*, which is to formalise *Sensing Tasks* (Chii Chang et al. 2015; Abdelwahab et al. 2015; Perera, Zaslavsky, Christen, et al. 2014). Furthermore, the web application handles compensations as well as data representation. In addition, the *Service Provider* offers a variety of API endpoints, e.g. for registering, discovering *Mobile Hosts* (Chii Chang et al. 2015), or issuing/ offering *Sensing Tasks* (Abdelwahab et al. 2015). These “back end” applications or API endpoints allow the *Service Provider* to be well connected with other components of the IoT architecture. To conclude, the *Service Provider* collects, transforms and presents sensing data to *Consumers* based on *Sensing Tasks*.

The components *Consumer*, *Thing*, *Owner* and *Service Provider* are used in all IoT architecture proposals, regardless of the *IoT Architecture Perspective* or the *Vision of IoT* the respective authors of the architecture proposals used. However, the subsequently presented *Gateway* and *Publisher* are either valid in the *Network IoT Architecture Perspective* or in the *Organisational IoT Architecture Perspective*. Albeit having some similarities, these components must be distinguished according to the *IoT Architecture Perspectives* in order be able to highlight the differences between them, the corresponding architectures and the consequences arising from these different perspectives on an IoT architecture.

Gateway

The *Gateway* component is associated with the *Network IoT Architecture Perspective* and its main purpose is to establish and maintain communication between *Things* and *Service Providers*. Both IoT architecture proposals which were previously assigned to the *Network IoT Architecture Perspective* include a component of which the sole purpose is to establish and maintain communication (Zachariah et al. 2015; Jo et al. 2015). Zachariah et al. (2015) suggest that a *Smartphone* acts as a gateway to

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

establish communication between the *Peripherals* and the *Cloud,* which can be mapped to *Things* and *Service Providers,* respectively. On the *Business Layer*, Zachariah et al. (2015) as well as Jo et al. (2015) suggest that the *Gateway* component offers the establishment and maintenance of communication as a service for which they demand compensation. In Jo et al.´s (2015) IoT architecture proposal both *Gateway Devices* and *Master Devices* provide a network infrastructure that allows the *Terminal Devices* to communicate with other devices. In return, *Terminal Devices* offer compensation for the services they consume. The *Business Model for Mobile Sensing as a Service* proposed by Mizouni and Barachi (2013), albeit assigned to the *Organisational IoT Architecture Perspective* (see 4.2.1), also describes a component that offers gateway services. The *Mobile Sensing Terminal Operator* proactively establishes connections to *Mobile Network Sensing Entities* in its vicinity, retrieves data and publishes it to the *Mobile Network Operator* (Mizouni & El Barachi 2013). In all mentioned cases, the *Gateway* establishes a peer-to-peer connection with the respective *Things* via BLE, Wi-Fi or a similar low energy wireless communication technology. Thus, the *Gateway* always establishes a direct network connection with a *Thing* in order to collect data. Due to this reason, the *Middleware Layer* is the most important aspect of this component throughout the IoT architecture proposals considered in this thesis. Regarding this layer, the *Gateway* is responsible for normalising communication between *Things* and *Service Providers*. It supports a wide array of different communication protocols and provides sufficient computing capacity to perform near network edge data transformation and routing tasks (Zachariah et al. 2015; Mizouni & El Barachi 2013). To summarise, the *Gateway´s* main purpose is to enable communication between *Things* and *Service Providers*.

Publisher

The *Publisher* component is associated with the *Organisational IoT Architecture Perspective* and maintains a database of *Things* along with their metadata and offers applications as well as interfaces to query or discover *Things* based arbitrary criteria. *Publishers* act as a proxy between *Things* and their respective *Owners*, and between *Service Providers* or *Consumers* (Perera, Zaslavsky, Christen, et al. 2014). On the *Business Layer*, a *Publisher* maintains a set of *Things* which have been registered with the *Publisher* by the *Owner* of the respective *Thing*. During the registration, the *Owner* can define access restrictions and other rules for accessing his *Thing* through the *Publisher (e.g.* the *Owner* wants to remain private or defines a threshold for a minimum compensation). The *Publisher* then either advertises the *Thing´s* services, e.g. sensing capabilities, or waits for *Sensing Tasks* issued by *Service Providers*. Depending on the specifications the *Owner* provided during the *Thing´s* registration, the *Publisher* can automatically perform the *Sensing Task* (opportunistic sensing) or requests the *Owner´s* permission beforehand (participatory sensing, see section 4.1). Additionally, the *Publisher* provides services for discovering or querying *Things* (Chii Chang et al. 2015). For all these services the *Publisher* may demand compensations (Perera, Zaslavsky, Christen, et al. 2014). In essence, the *Publisher* acts as a *Thing* repository and as a proxy for the *Thing* he manages. In contrast to the *Gateway*, the *Publisher* must not necessarily establish a peer-to-peer connection with *Things*. In contrast, *Publishers* and *Things are* connected via contracts or registrations defined by the *Owners* of *Things*.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Table 1 summarises the previous component descriptions and assigns each IoT architecture component to one of the synthesised *Consumer*, *Thing*, *Owner*, *Service Provider*, *Gateway*, and *Publisher* components. The first three columns of Table 1 contain the source for the architecture proposal, the considered architecture component in terms of the respective architecture proposal and the mapped component in terms of the *Holistic IoT Architecture Framework* to be developed in this thesis. The fourth column contains the description of the respective architecture component. The description characterises each component using the *Generic IoT Architecture* layers (see Figure 14). The architecture proposals and each component were examined with each of these layers in mind, thus each component´s description consists of specialised descriptions addressing a specific layer. The layers and respective descriptions are colour coded (see Figure 14 for reference) and contain the abbreviated name of the respective layer (e.g. B for Business Layer, etc.). The last column lists requirements which have been derived from the component´s descriptions. Each requirement is considered as either a “must-have”, a “should-have” or a “could-have”. Whereas a “must-have” requirement is critical for successfully fulfilling the expectations for a system and a “could-have” requirement merely provides supplementary functionalities that are not critical for the system (Bradner 1997; Clegg & Barker 1994).

Figure 20 and Figure 21 utilise these components and highlight the differences between IoT architecture proposals applying either the *Network IoT Architecture Perspective* or the *Organisational IoT Architecture Perspective*. The differences between both *IoT Architecture Perspectives* are discussed in more detail in the next section. The analysis of IoT architecture components was conducted in order to answer RQ1.1, RQ1.3 and RQ3.1 as well as achieve RO1 and RO3.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Table 1: IoT Architecture Component Descriptions based on the *Generic IoT Architecture Layers* and component requirements (own listing)

Architecture	Component	Mapping		Description	Requirements
Sensing as a Service (Sheng et al. 2013; Perera, Zaslavsky, Christen, et al. 2014)	Sensor Owner	Owner	B	• Offers access to sensing data • Wants to be compensated for providing access to sensing data • Wants to be able to remain private/ anonymous • Registers individual sensors with *Sensor Publishers*	• Must be able to remain private/ anonymous • Must be able to control/ manage sensors/ things he owns • Must be able to manage access to sensors/ things he owns • Must be able to define usage constraints on *Things* he owns
			N	• Provides direct communication access to owned sensors (e.g. home network, mobile phone internet access)	
			P	• Owns an arbitrary number of sensing devices that can capture various kinds of environmental data	
	Sensor Publisher	Publisher	B	• Forms contracts with *Sensor Owners* regarding individual sensors/ things • Maintains a database of contracted sensors along with metadata describing the sensors • Ensures privacy and anonymity of contracted *Sensor Owners* • Gets compensated by *Sensor Owners* or *Extended Service providers* for mediation services	• Must be able to forward *Sensing Tasks* to *Sensor Owners* • Must be able to anonymise data provided by sensors of contracted *Sensor Owners* • Must be able to detect/ discover *Sensor Owners* based on sensors/ things detected in the network • Must support various *Incentive Mechanisms* • Must expose API unified endpoints for accessing sensors
			A	• Provides direct access to sensors/ things for *Sensor Data Consumers* (API) • Provides access to sensors for *Extended Service Providers* (API)	
			M	• Normalises communication between *Things* • Provides unified endpoints for various types of sensors/ things	
			N	• Discovers and establishes communication with sensors/ things	

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

	Extended Service Provider	Service Provider	B	• Provides various domain specific value added services for *Sensor Data Consumers* • Translates and issues sensing tasks to *Sensor Publishers* • Presents sensing data to *Sensor Data Consumers* • Compensates *Sensor Publishers* for using their services and sensors	• Must be able to access endpoints exposed by *Sensor Publishers* • Must provide web applications for *Sensor Data Consumers* • Must support various *Incentive Mechanisms* • Must be able to transform arbitrary information provided by *Sensor Data Consumers* into uniform *Sensing Tasks*
			A	• Provides web applications and interfaces for creating *Sensing Tasks* and representing data • Provides systems and services capable of discovering and communicating with *Sensor Publishers*	
	Sensor Data Consumer	Consumer	B	• Consumes and compensates for value added services provided by *Extended Service Providers* • Can directly access sensors/ things through *Sensor Publishers,* thus can bypass the *Extended Service Provider* and directly compensate the *Sensor Publisher*	
Smartphone Gateways (Zachariah et al. 2015)	Peripheral	Thing	N	• Advertises its presence via BLE • Supports BLE as a communication technology	• Must have any kind of communication interface • Must be able to store and provide characteristics describing itself
			P	• Collects various kinds of environmental data • Expose certain characteristics (e.g. type of sensor, attributes, additional meta information)	
	Smartphone	Gateway	B	• Smartphone users want to be compensated for transmitting data between *Peripherals* and the *Cloud*	• Must be able to support various *Incentive Mechanisms* • Must transform characteristics of connected *Peripherals* into unified API endpoints • Must be able to scan/ discover *Peripherals* in vicinity and establish connections
			A	• Exposes unified API endpoints when acting as a BLE gateway	
			M	• Normalises communication between *Cloud* and *Peripheral* by abstracting proprietary interfaces of *Peripheral* devices • Acts as IPv6 gateway and allows *Peripherals* to be directly addressable • Acts as BLE gateway for *Peripherals* • builds API endpoints based on the characteristics of the *Peripherals*	

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

			N	• Discovers and establishes communication with *Peripherals* • Detects advertisements of BLE enabled *Peripherals*	
	Cloud	Service Provider	B	• Compensates *Smartphone* owners for acting as gateways	• Must be able to support various *Incentive Mechanisms* • Must provide uniform application/ service endpoints
			A	• Provides arbitrary endpoints for data sensed by *Peripherals* and transmitted by *Smartphones* to be sent to	
Heterogeneous Mobile Network (Jo et al. 2015)	General Static Cloud	Service Provider	B	• Has large computing capacities and is permanently reachable/ accessible • Wants to offload tasks, communication and services to *Mobile Dynamic Cloud*, which consists of *Master Devices, Gateway Devices, Terminal Devices*	• Must be able to support various *Incentive Mechanisms* • Must provide uniform application/ service endpoints • Must be well connected
			A	• Provides various services (IaaS, PaaS, SaaS), which are consumed and brokered by *Master Devices* and *Gateway Devices* • Provides endpoints to be used by *Master Devices* for acting as service providers • Provides endpoints to be used by *Terminal Devices* • Provides endpoints to be used by *Sensing Devices*	
			N	• Is connected to *Master Devices* and *Terminal Devices* via cellular networks or Wi-Fi • Is connected to *Terminal Devices* via BLE or Wi-Fi	
	Terminal Device	Consumer	B	• Consumes services provided by *General Static Cloud* directly or through *Gateway*- and *Master Devices* • Outsources computational tasks through *Gateway Devices* to *Master Devices*	• Must be able to support various *Incentive Mechanisms* • Must be able to scan/ discover *Gateway Devices* in the vicinity
			N	• Is connected to *Gateway Devices* via BLE or Wi-Fi • Is connected to *General Static Cloud* via cellular network	

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

	Sensing Device	Thing	B	• Acts as a service provider, provides access to sensing capabilities, wants to be compensated for service access • Has *Owner* who controls and defines restrictions, properties and access to the services provided by the device	• Must be able to support various *Incentive Mechanisms* • Must be able to define access rules and restrictions • Must be able to remain private/ anonymous • Must be discoverable
			N	• Is connected to *General Static Cloud* via Wi-Fi • Is connected to *Mobile Dynamic Cloud* through *Master Devices* via Wi-Fi	
			P	• Has limited computational, networking and storage capabilities • Collects various kinds of environmental data	
	Gateway Device	Gateway / Publisher	B	• Acts as a service broker, forwards services provided by *Master Devices* to *Terminal Devices*, wants to be compensated for brokering services/ communication	• Must be able to support various *Incentive Mechanisms* • Must be able to scan/ discover *Master Devices* and *Gateway Devices* in the vicinity • Must be able to keep the *Terminal Devices* private and anonymous
			N	• Is connected to *Terminal Devices* via BLE, Wi-Fi • Is connected to other *Gateway Devices* via BLE, Wi-Fi • Is connected to *Master Devices* via BLE, Wi-Fi	
	Master Device	Gateway / Publisher	B	• Represents services (service endpoints) provided by the *General Static Cloud*, service requests are either forwarded to the *General Static Cloud* or processed locally • Wants to be compensated for service provisions • Represents, connects and manages *Gateway Devices* and *Terminal Devices* connected to itself – which forms a cloudlet managed by the corresponding *Master Device*	• Must be able to support various *Incentive Mechanisms* • Must provide uniform endpoints for services • Must be able to collaborate with other *Master Devices* • Must be well connected
			N	• Is connected to *General Static Cloud* via cellular network, BLE, Wi-Fi • Is connected to other *Master Devices* via cellular network, BLE, Wi-Fi • Is connected to *Gateway Devices* via BLE, Wi-Fi	
Cloud of Things	Cloud User	Consumer	B	• Wants to obtain environmental data and is willing to provide certain compensations for the data	

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

	First Tier Cloud	Service Provider	B	• Provides various cloud services • Maintains database of available *Cloud Agents* and coordinates their actions • Translates *Sensing Tasks* and forwards them to available and suitable *Cloud Agents*	• Must be able to transform arbitrary information provided by *Cloud Users* into uniform *Sensing Tasks* • Must be able to store various types of data and data formats
			A	• Provides unified user interfaces (e.g. web applications, endpoints) • Provides applications for creating *Sensing Tasks* • Provides access to stored data retrieved by *Cloud of Things*	
	Cloud Agent	Gateway / Publisher	B	• Manages large amounts of connected devices which are part of the *Cloud of Things* • Handles *Sensing Tasks* and selects appropriate connected device from the *Cloud of Things*	• Must be able to forward *Sensing Tasks* to *Sensor Owners* • Must be able to support various *Incentive Mechanisms*
			M	• Normalises communication between various types of devices belonging to the *Cloud of Things* • Selects devices of the *Cloud of Things* to perform *Sensing Tasks*, adheres to various selection requirements (e.g. energy consumption, distance, etc.) • Manages *Virtual Sensor Networks* created based on the criteria of the *Sensing Tasks*	
			N	• Supports many kinds of communication technologies and protocols	
	Cloud of Things	Service Provider	B	• Provides various degrees of available computational power or sensing capabilities • Wants to be compensated for computational power or sensing capabilities	• Must be able to support various *Incentive Mechanisms*
			N	• Individual devices are dynamically assigned to virtual networks • Provides limited communication capabilities and protocols	
			P	• Can gather environmental data (from physical devices) • Can gather virtual data (from virtual devices, e.g. services providing insights on system usages, internet traffic, etc.)	

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Mobile Device as a Sensory Service Mediation (Chii Chang et al. 2015)	Client	Consumer	B	• Wants to obtain environmental data by issuing *Sensing Tasks* to *Mobile Hosts* • Uses *Discovery Server* to find suitable *Mobile Hosts*	
			N	• Directly connected/ subscribed to *Mobile Hosts* via peer-to-peer connections (e.g. via local Wi-Fi discovery, BLE advertising) • Connected to *Discovery Server* via internet	
	Discovery Server	Publisher	B	• Manages many registered *Mobile Hosts* • Selects appropriate *Mobile Hosts* based on *Sensing Tasks*	
			A	• Provides web-applications to discover/ search for *Mobile Hosts* • Maintains database containing metadata describing *Mobile Hosts* • Provides web-applications for *Mobile Hosts* to register themselves with the *Discovery Server* and manage their metadata.	
	Mobile Host	Thing	B	• Provides sensing services (e.g. one-time-, real-time-, and periodical sensing) • Wants to define what data is shared • Wants to remain private and transfer as less data as possible • Dynamically uses *Utility Cloud* to upload sensed data • Is registered with one or more *Discovery Servers* • Authorises users to access the data stored on the *Utility Cloud*	
			N	• Is directly connected to *Clients* via peer-to-peer connections • Is connected to *Discovery Servers* via the internet (e.g. Wi-Fi, cellular network)	
			P	• Can gather various kinds of environmental data	
	Utility Cloud Service	Service Provider	B	• Provides data storage for sensing data based on pay-by-use • Is dynamically provisioned by *Mobile Host* • Limits access to data to authenticated and authorised users	• Must be able to store various types of data and data formats

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

			A	• Provides applications to access stored sensing data (web-applications or endpoints)	
Mobile Phone Sensing as a Service Business Model (Mizouni & El Barachi 2013)	Mobile Network Operator	Service Provider	B	• Provides access to a cellular network • Provides access to services of *Value-Added Service Providers* through his network • Wants to obtain environmental data and is willing to compensate *Mobile Sensing Terminal Operators* for collecting the data	• Must be able to support various *Incentive Mechanisms*
	End-User	Consumer	B	• Accesses cellular network and compensates *Mobile Network Operator for the access* • Uses *Value-Added Services* through the cellular network • Wants to obtain environmental data and is willing to compensate *Mobile Sensing Terminal Operators* for collecting the data	• Must be able to support various *Incentive Mechanisms*
	Mobile Sensing Terminal Operator	Gateway/ Publisher	B	• Proactively gathers data from *Mobile Sensing Network Entities* and aims to be compensated for the collected data • Compensates *Mobile Sensing Network Entities* access to their data	• Must be able to support various *Incentive Mechanisms*
			A	• Provides applications to scan/ discover *Mobile Sensing Network Entities* in the vicinity • Provides applications to collect data from *Mobile Sensing Network Entity* and provide compensation	
			N	• Connects to *Mobile Sensing Network Entities* via various communication technologies and protocols	

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

	Mobile Sensing Network Entity	Thing	B	• Provides access to sensing data and wants to be compensated for the access • Wants define restrictions for data access (e.g. kind of data, access times, etc.)	• Must be able to support various *Incentive Mechanisms* • Must be able to controlled/ managed by the respective owner • Must support to define usage constraints
			P	• Provides various kinds of environmental data	

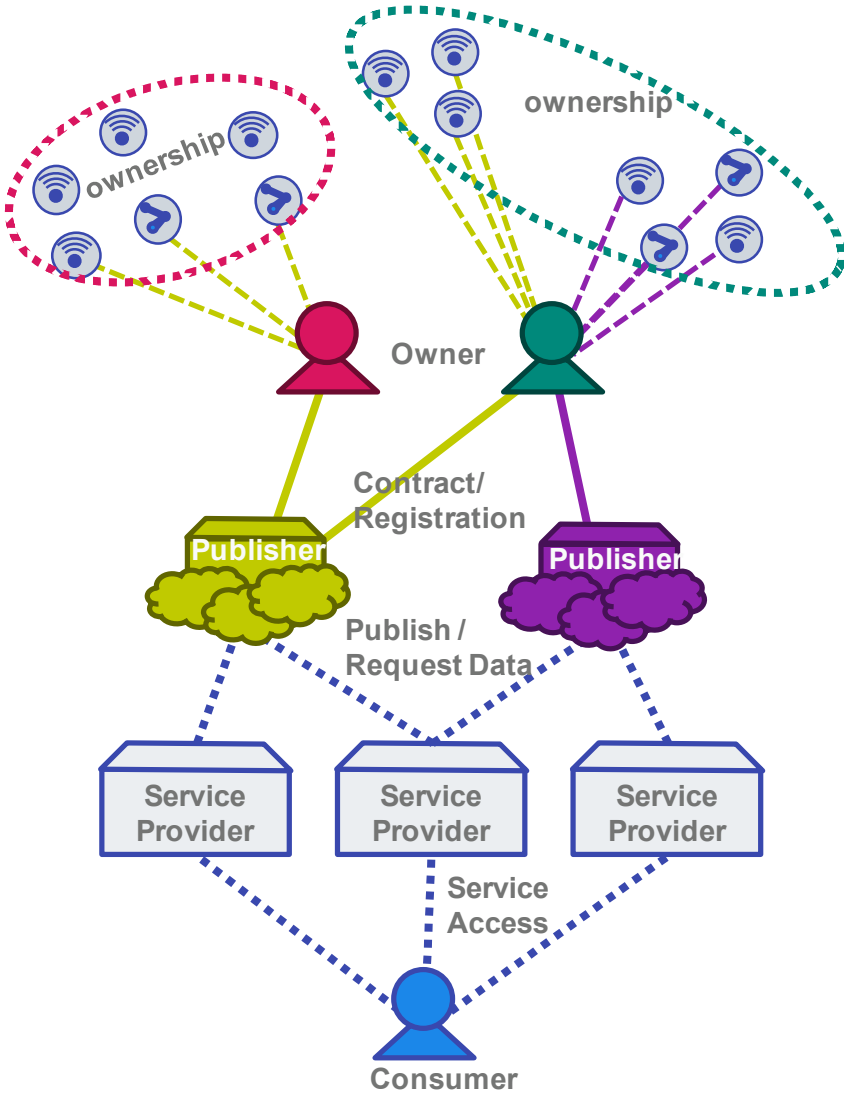


Figure 20: Preliminary generic IoT architecture applying the *Organisational IoT Architecture Perspective (*own illustration)

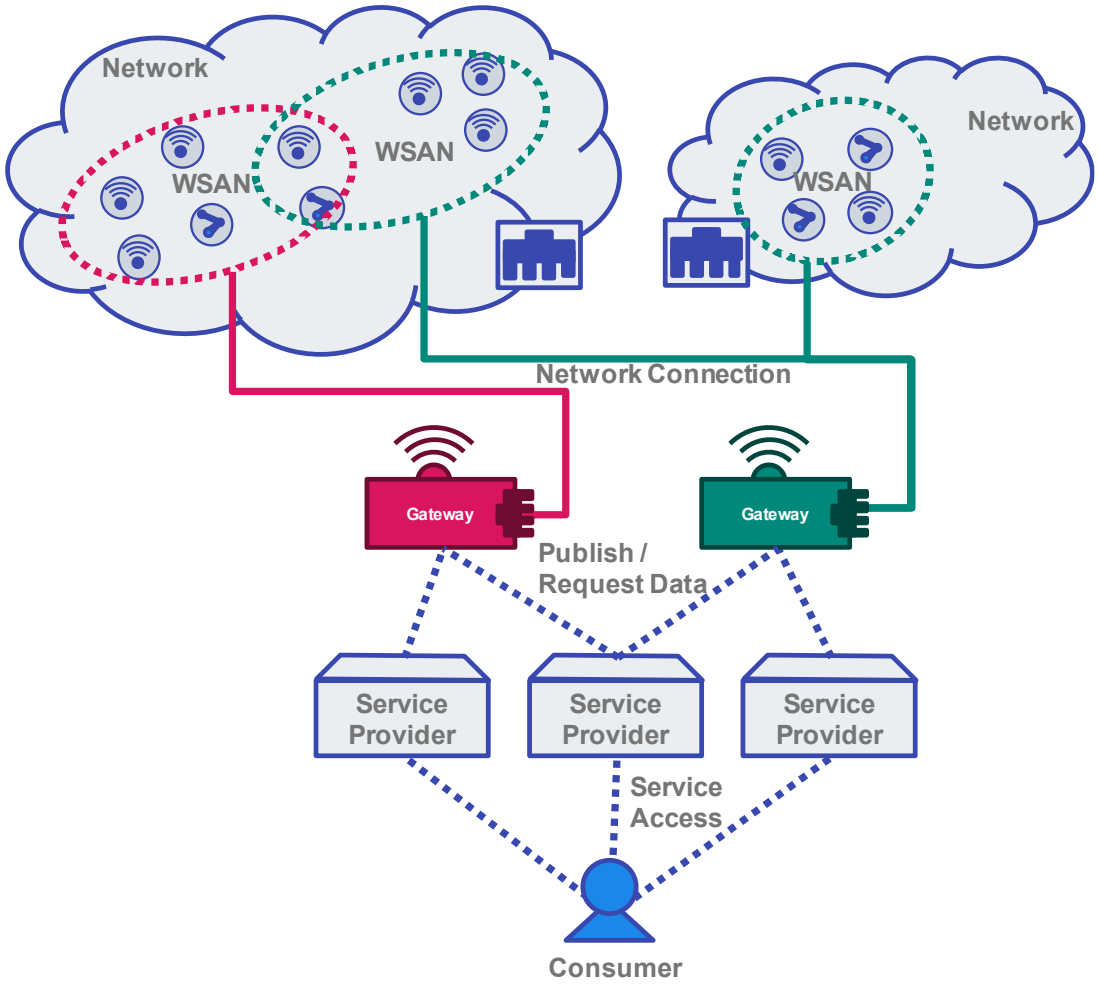


Figure 21: Preliminary generic IoT architecture applying the *Network IoT Architecture Perspective* (own illustration)

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

4.3 Thing Management – An underdeveloped component

Based on the *IoT Architecture Perspectives* presented in the previous sections, this section further elaborates their differences and draws conclusions about potentially missing components or features in the IoT architectures analysed in this thesis. Based on the conclusions presented in section 4.3.1, the principles of *Identity Management* are discussed for the further development of a new component for the *Holistic IoT Architecture* to be developed as part of this thesis (see section 4.3.2). Based on the findings of the previous sections, the development of the new component *Thing Management* is described in section 4.3.3.

4.3.1 Differences between Network- and Organisational IoT Architecture Perspectives and Conclusions

To highlight the differences between the architectures applying either the *Organisational IoT Architecture Perspective* or the *Organisational IoT Architecture Perspective*, two scenarios regarding the deployment-environment of *Things* are described (see Figure 22). The main difference between the scenarios is the degree of control an *Owner* has regarding the environment his *Things* are deployed in.

In the first scenario, which is essentially described by Perera et al. (2014), the *Owner* of a *Thing* has full control over the environment. In Perera et al.´s (2014) example, an *Owner* deploys a smart fridge in his home network. Thus, the *Owner* controls and establishes the connectivity of the smart fridge and can directly connect or register the smart fridge with a *Publisher*. Furthermore, the *Owner* can directly access his *Things* and modify restrictions, privacy settings and other parameters of his *Things* with ease. In this scenario, the *Owner* additionally acts as the *Gateway* because he establishes a permanent network connection between his *Things* and the internet. *Publishers* can act as proxies between *Things* and *Service Providers*. This is possible because *Publishers* are directly connected to the *Things* registered to them and they do not need to rely on external or uncontrolled *Gateways* to establish a network connection (see Figure 22).

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

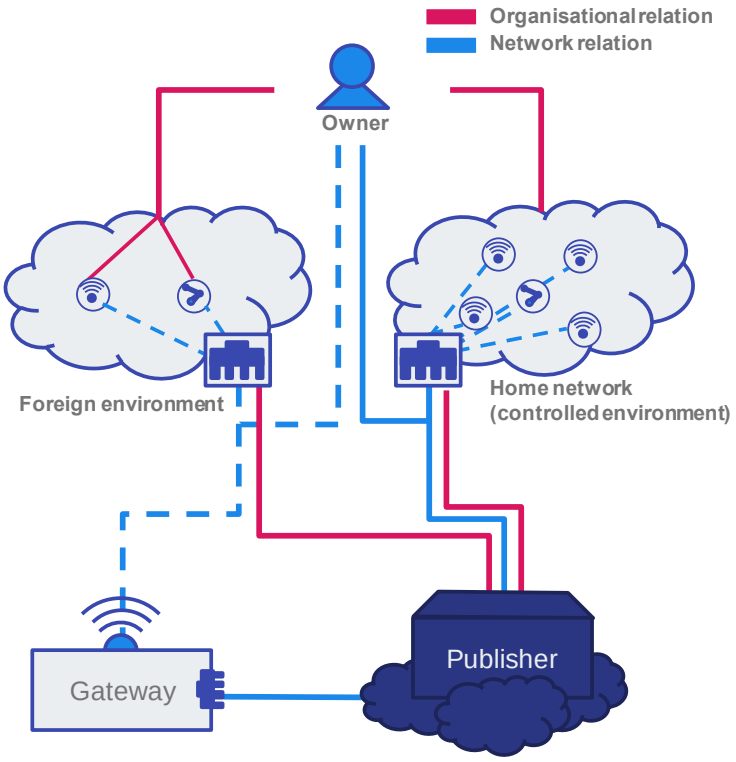


Figure 22: IoT Architecture Perspectives on different scenarios based on the Owner´s degree of control in the deployment environment (own illustration)

Contrasting the first scenario, the second scenario regards *Things* deployed in a "foreign" environment (see Figure 22). *Owners* still own their *Things* from an organisational view, but deploy their *Things* in an environment they do not completely control (e.g. commercial sensor data providers deploy sensors in public spaces). The deployment of *Things* in "foreign" environments indicates that *Owners* do not have permanent access to their *Things* and that they neither provide nor control the communication between their *Things* and the corresponding *Publishers*. This scenario is described by almost every architecture proposal regarded in this thesis. For example, the IoT Smartphone Gateway presented by Zachariah et al. (2015) (see section 4.2.1 and Figure 15) indicates that *Peripherals,* which are *Things*, need to rely on the opportunistic and unreliable communication channels provided by mobile gateways. The Mobile Phone Sensing as a Service Business Model presented by Mizouni and El Barachi (2013) also relies on a gateway component to establish a non-permanent connection between *Things* and *Service Providers*. In all the above-mentioned cases, the respective *Publisher* of a *Thing,* which should act as a proxy between *Things* and *Service Providers,* is not considered at all. In essence, in this scenario *Gateways* directly communicate with *Things*. This leads to two consequences. First, *Publishers* cannot fulfil their role of managing access rules and restriction, or maintain the privacy of the respective *Owner* of a *Thing*. This consequence indicates that these restrictions must be stored and managed directly by the respective *Thing*. However, due to the deployment of *Things* in a "foreign" environment, which includes limited, uncontrolled and unreliable access, managing *Things* could become a onerous task for *Owners*. Second, considering the fact that a *Thing* should always be registered to at least one *Publisher*, the

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

corresponding *Publisher´s* metadata must also be stored on the *Thing* itself (Perera, Zaslavsky, Christen, et al. 2014). This further complicates the management of *Things* and increases the technical requirements for them as well (e.g. additional storage capacities, additional computing capacities to support simplistic authentication, etc.). *Things* need to store their *Publisher´s* and *Owner´s* metadata. *Gateways* need to determine the correct *Publisher* to transfer the data to, based on the metadata provided by each *Thing*. This requirement for the *Owners* to permanently be able to access and manage their *Things* and keep the metadata up-to-date could significantly hamper the deployment of large scale WSNs due to increased operating costs. This is because the *Owners* of these networks very likely need to provide network access themselves in order to guarantee up-to-date metadata of the *Things* in the WSN. The additional metadata stored on *Things* is required because *Gateways* directly communicate with *Things* and bypass the corresponding *Publishers*. Without this metadata the *Gateway* cannot determine the receiver of the data, what kind of data they collect or if they are allowed to access the *Thing´s* data in the first place.

In conclusion, the following problems arise when the *Network IoT Architecture Perspective* and the *Organisational IoT Architecture Perspective* and the corresponding scenarios are considered in combination.

- Lack of privacy and anonymity
  With the ability to bypass the *Publisher*, which should act as a proxy and ensure privacy and anonymity, neither the privacy nor anonymity of an *Owner* of a *Thing* are guaranteed.
- Onerous *Thing* management
  With each *Thing* being able to be directly but not permanently addressed by *Gateways*, the metadata describing the *Thing* must be manageable at all times. However, this becomes an issue in “foreign” environments when communication cannot be guaranteed. Furthermore, this issue scales with the number of *Things* to manage.
- Increased requirements for *Things*
  The requirement to be able to store additional metadata, to host a simplistic authentication framework and to provide some means of remote management requires the *Things* to provide additional storage and computational capacities.
- Reliance on potentially untrusted *Gateways*
  Gateways currently do not have an organisational relation with *Publishers*. Thus, they can be considered as untrusted because it is not guaranteed that the *Gateway* conscientiously transmits the data to the respective *Publisher*. Likewise, a *Thing* cannot determine if a *Gateway* that tries to access it´s data or services acts on behalf of the *Thing´s* associated *Publisher*. This untrusted communication can lead to issues ranging from publishing false data, negative impacts in a *Thing´s* reputation all the way to the denial of services provided by a *Thing,* or publishing data which violates the privacy of the *Thing´s Owner*.

To solve the above-mentioned problems, *Owners* need a system to easily manage their *Things*. This system should be able to manage the restrictions, access rules and other settings of a *Thing*.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Furthermore, the system should be able to manage the registrations of each *Thing* with a *Publisher*. The guiding principles and development of this *Thing Management System* (TMS) are discussed in section 4.3.2 and section 4.3.3, respectively. After having discussed the *Thing Management System* as a new component for the *Holistic IoT Architecture Framework based on S2aaS*, it will be embedded into the architecture in section 0. However, the *Thing Management System* only solves the problems "*Lack of privacy and anonymity*" and "*Onerous Thing management"*. In order to tackle the problems "*Increased requirements for Things"* and *"Reliance on potentially untrusted Gateways"*, the relation between *Gateways*, *Publishers* and *Things* as well as the role of a *Gateway* itself must be revised (see Figure 22). For *Gateways* to become trusted entities in relation to *Publishers* and being able to access *Things* on behalf of a *Publisher* an authentication system between *Gateways* and *Publishers*, and between *Gateways* and *Things* must be designed (see section 4.3.4).

4.3.2 Utilising Principles of Identity Management for Thing Management in IoT

The previous section suggested the so-called Thing Management System as a tool for *Owners* to manage their *Things* and share their Things services with Publishers. By being responsible for these tasks, the Thing Management System is expected to ensure the privacy and anonymity of a Thing´s Owner while simultaneously simplifying the management of these Things. In order to be able to design and develop such a system guiding principles or a domain model need to either be developed or identified and transferred from other research or problem domains. As described in section 2.2, the suggestion and the development phase of the GDC are to be guided by the patterns *Theory Development* and *Problem Space Tools and Techniques* (Vaishnavi & Kuechler 2007). Based on the task, which is to develop a system to manage and share *Things*, and the nature of these *Things*, the principles of *Identity Management* shall be utilised to develop the *Thing Management System*.

Vaishnavi and Kuechler (2007) suggest that the pattern *Problem Space Tools and Techniques* can be applied when a research problem has been identified and the researcher wants to assess the problem space and utilise his general knowledge in order to identify tools and techniques that assist in the solution of the research problem. In this particular case the sharing of data (e.g. what kind of data) and providing access to services (e.g. specific users are granted or denied access), the maintaining of privacy and anonymity, and the management of shared data and access (e.g. granting and revoking access) make up the problem space. In essence, the problem space consists of issues regarding the management of *Things* by their corresponding *Owners*. However, as described in section 2.4, IDM deals with issuing credentials to users, identifying users with identifiers and granting access to services and data. As mentioned earlier, the nature of *Things* led to the conclusion that IDM provides suitable principles to guide the development of a system for managing *Things*.

In section 4.2.2, *Things* have been defined as components that provide unique sensing capabilities and have rules for accessing the services they provide. This definition of *Things* bears similarities with the

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Service Providers[13] described in the context of IDM in section 2.4. Users wanting to access a *Thing´s* services can authenticate themselves and are authorised to access the sensing or metadata of the *Thing*. While authentication and authorisation are not necessarily required for accessing a *Thing´s* services, these mechanisms are required when the service of a *Thing* must be compensated, as described in section 4.2.2. As soon as a *Thing* requires compensation for its services, it must be able to identify users and grant or deny access, depending on the compensation the respective user provides. In order to be able to identify a user and if he needs to, or already has provided compensation for the services, the *Thing* needs to issue credentials to its users. Hence, a *Thing* is "surrounded" by an *Identity Domain* and uniquely identifies it´s users *Identities* based on a set of Identifiers defined by the *Identity Domain* of the Thing. Based on this description, a *Thing* can be regarded as a *Service Provider* in the context of IDM.

However, with regard to the relationship between a *Thing* and its corresponding *Owner*, which has also been defined in section 4.2.2, a *Thing* can additionally be interpreted as an *Identity* of an *Owner.* The *Owner* of a *Thing* imposes his requirements regarding the compensation of services and the access restrictions onto the *Things* he owns. As described by Perera et al. (2014) i.a., *Owners* have full control over the *Things* they own and thus can decide if *Things* are published and which characteristics are made available by publishing a *Thing*. The characteristics of a *Thing* generally consist of descriptions of the services a *Thing* provides (e.g. sensor type, data scheme, information model, owner preferences, availability, etc.) and additional metadata (e.g. location) (Zachariah et al. 2015; Perera, Zaslavsky, Christen, et al. 2014). These characteristics, primarily denoting and designating the *Thing´s entity* and *identity* respectively (see Figure 3), could very well be used to infer characteristics of the *Thing´s Owner.* Thus, the *Identity* of a *Thing* also refers to its *Owner*, which is its *Entity* in terms of IDM. For example, when the metadata of a *Thing* contains location information, one could infer that the *Owner* of that *Thing* is or was active in that general area, e.g. because he had to deploy the device. Additionally, since an *Owner* imposes his own requirements and specifications onto his *Things,* e.g. by specifying access rules or defining what kind of compensation is required, he implicitly transfers characteristics denoting his own *Entity* to the *Thing´s* characteristics, which can later be used to designate the *Thing´s Identity*. Hence, a *Thing* can also be regarded an *Identity* referring to his *Owner* as its *Entity.* Figure 23 illustrates this relationship between *Things* and *Owners* in terms of IDM.

[13] It must be noted that the term *Service Provider* is ambiguous. Whenever a *Service Provider* in terms of IDM is meant, it is mentioned in the text. Otherwise the *Service Provider* defined in section 4.2.2 is meant.

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

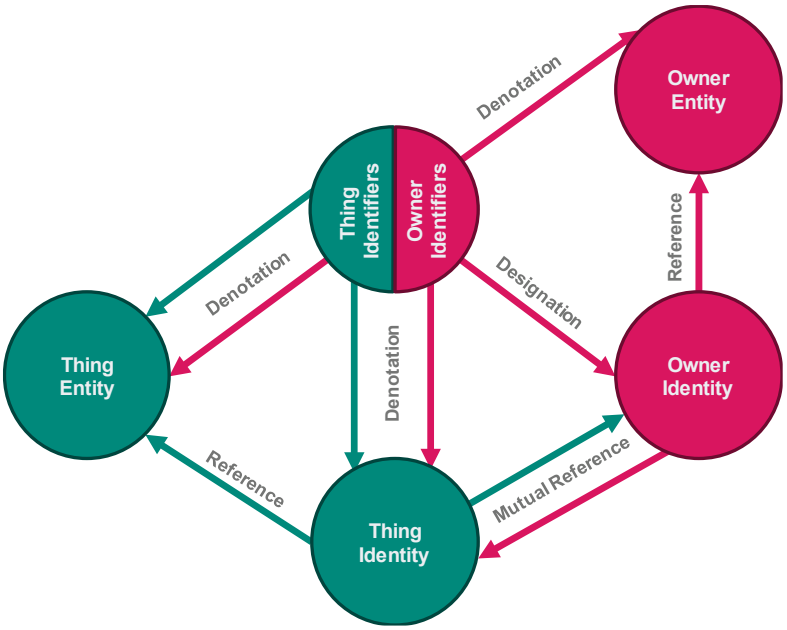


Figure 23: Relationship between *Things* and *Owners* in terms of IDM (own illustration)

In conclusion, in the domain of IDM, *Things* can be regarded as both *Identities* and *Service Providers* alike. A *Thing* can act as a *Service Provider*, requiring users to authenticate themselves and granting access to the services it provides. At the same time, in order to be able to be discovered, a *Thing* exposes its *Identity*, which consists of characteristics describing the *Thing*, its services explicitly and its *Owner* implicitly. Hence, a *Thing* must always be considered in combination with its respective *Owner*.

The insight that a *Thing* is an *Identity* will help significantly during the development of the *Thing Management System* because the "*Laws of Identity*" (Cameron 2005) (see section 2.4) can be used as requirements for the consecutive development steps of the *Axiom Based Design* which is applied in the following section. Furthermore, the *Personal Authentication* Device (see Figure 5) proposed by Jøsang and Pope (2005) can also be used to guide the development of the *Thing Management System* as it has been designed to manage and share multiple sets of credentials or *Identities* with *Service Providers*. Simultaneously, the insight that a *Thing* can also be considered as a *Service Provider* in terms of IDM will be helpful in section 4.3.4, where the roles and relations of *Gateways* are revised. With *Things* acting as *Service Providers*, *Gateways* can be interpreted as users who need to authenticate with a *Thing*. However, this requires a *Thing* to provide an authentication system, which increases the technical requirements for it (see section 4.3.1).

4.3.3 Development of the Thing Management System

Having discussed the need for a convenient, anonymity and privacy preserving system for managing *Things* in section 4.3.1 and having identified suitable existing concepts and technologies in section 4.3.2 which can be used to guide the development of such a *Thing Management System*, this section will now present the development of the *Thing Management System*. The development of this new component

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

for the *Holistic IoT Architecture Framework* to be designed in this thesis, will apply the principles of the *Axiomatic Design* method (see section 2.2).

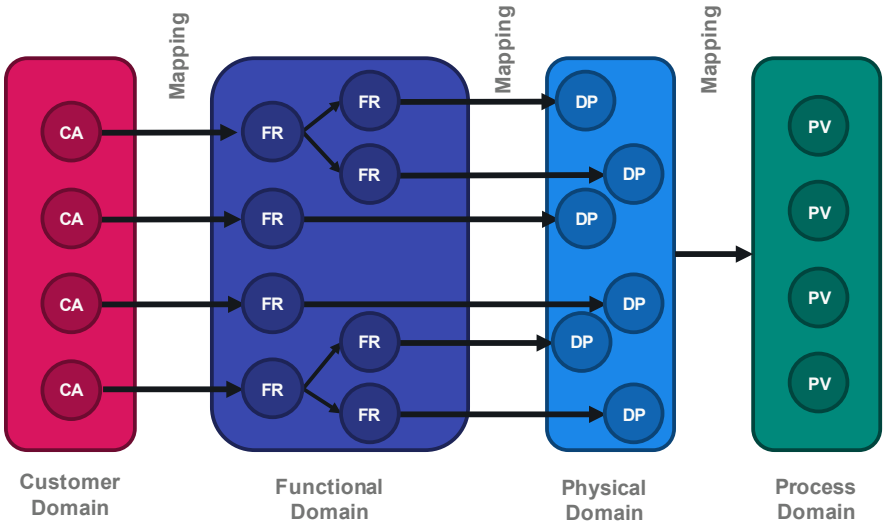


Figure 24: Relationship of domains, mapping and design space in axiomatic design (adapted from Suh & Do 2000)

The first step of the development process applying *Axiomatic Design* consists of mapping *Functional Requirements* or deriving them from the *Customer Domain* (see Figure 24). The set of *Customer Attributes* (CA) that will be used to derive the *Functional Requirements* (FR) are provided by Jøsang and Pope (2005) and Cameron (2005). The *Personal Authentication Device* (PAD) described by Jøsang and Pope (2005) (see Figure 5) will be used to retrieve requirements for the *Thing Management System*. The PAD is based on the idea that *Service Providers*, in terms of IDM, generally have access to systems that allow the automated management of *Identities*, while users do not use or have access to such systems. Jøsang and Pope (2005) discuss that the growing number of *Service Providers* a user can and will consume might lead to security and usability issues when users need to manage these identities manually (e.g. by memorising credentials for each *Service Provider)*. Consequently, the concept of *Federated Identity Management Models* was introduced which in theory only requires a single set of credentials for the users to memorise or manage. However, Jøsang and Pope (2005) argue that if users only needed to manage a single set of credentials this would imply some sort of global federated *Identity Domain*, which is unlikely to be feasible. This is because due to different requirements regarding the characteristics or *Identifiers* making up an *Identity* across different *Identity Domains (e.g.* different legal or security requirements) (Jøsang & Pope 2005). To address the issues of poor usability regarding the management of identities, Jøsang and Pope (2005) present the PAD, which is a device or service controlled by a single user that securely stores an arbitrary number of credentials which are linked to corresponding *Service Providers.* Consequently, a user only needs to remember a single set of credentials for authentication with his own PAD to be able to authenticate with every other *Service Provider* he uses. Jøsang and Pope (2005) suggest that this can create a so called “virtual single-sign-on”

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

environment, where a user is authenticated by single set of credentials across multiple *Service Providers*. The different sets of credentials for each *Service Provider* are handled by the PAD, thus the prefix "virtual" single-sign-on. Furthermore, the PAD can be backwards-compatible and can be implemented in every existing *Service Provider´s* authentication framework because it only manages the credentials or *Identifiers* and not the authentication (e.g. it can be interpreted as a database of a user´s credentials) (2005).

The descriptions provided by Jøsang and Pope are listed in Table 2. These were extracted from the description of the PAD provided by Jøsang and Pope (2005) and will be used as *Customer Attributes* from the *Customer Domain*. In addition to the requirements extracted from the descriptions of the PAD, the *"Laws of Identity"* described by Cameron (2005), which are briefly described in section 2.4, will be used as additional CAs for the *Customer Domain*. These seven "*Laws of Identity"* will be especially useful for generating requirements for the *Thing Management System* because the laws are addressing *Identity Management Systems* in general (Cameron 2005). With *Things* being both *Identity* and *Service Provider* (see section 4.3.2) and with the *Thing Management System* aiming to simplify the management of *Things*, the requirements for an *Identity Management System*, provided in the form of the *"Laws of Identity"*, can be also be applied to the *Thing Management System*. These requirements are also listed in Table 2.

The *Customer Attributes*, given in the form of the statements and descriptions provided by Jøsang and Pope (2005) as well as Cameron (2005), essentially express the customer needs and expectations that the complete design, which in this case is the TMS, must fulfil. These expressions and expectations are likely to be vague and unstructured (e.g. extracted from interviews or other informal specifications) and thus need further refinement and analysis in order to be able to map the CAs to *Functional Requirements* (FR). This refinement or mapping is guided by the *information axiom* and *independence axiom* provided by the *axiomatic design* approach. To illustrate this mapping between the *Customer Domain* and the *Functional Domain* (see Figure 24), Table 2 lists the CAs as well as some preliminary requirements and Table 3 lists the actual FRs and the corresponding mapping between CAs and FRs. The following paragraphs will further elaborate on the structure of the mentioned tables, the CAs, the preliminary requirements, the FRs and the notable exceptions of them.

The first column of Table 2 contains unique identifiers assigned to CAs extracted from either Jøsang and Pope (2005) or Cameron (2005) which are shown in the second column. The third column lists the preliminary set requirements which have been extracted from these CAs. A preliminary requirement in column three addresses one or more actors or components and defines an expected behaviour or functionality for these actors or components. In addition, each requirement is considered as either a "must-have", a "should-have" or a "could-have". Whereas a "must-have" requirement is critical for successfully fulfilling the expectations for a system and a "could-have" requirement merely provides supplementary functionalities that are not critical for the system (Bradner 1997; Clegg & Barker 1994). Furthermore, each requirement is assigned with a unique identifier derived from the corresponding CA´s identifier. This additional mapping between a single CA and the corresponding preliminary

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

requirements is done because a CA could incorporate one or more preliminary requirements (e.g. CA1, CA26, i.a.).

The preliminary set of requirements listed in Table 2, which has been directly derived or extracted from the statements or CAs, still contains duplicates and is generally unstructured. Furthermore, some preliminary requirements are either non-functional requirements (e.g. CA1.3, CA9.1 – CA9.3, i.a.) or must be further decomposed (e.g. CA3.1, CA8.1, i.a.) to satisfy the *independence* and *information axiom* described in section 2.2.

The preliminary requirement CA5.1, provided by Jøsang and Pope (2005, p.8), states that the TMS could be able to be deployed on a portable device. However, considering the intended application of the PAD or TMS, this requirement is not applicable. This is because Jøsang and Pope (2005) suggest that the PAD only stores the *Credentials* and *Identities* of a single user and that the PAD is only used when the user actually needs these *Credentials* (e.g. when he needs to authenticate with a *Service Provider)*. In contrast, the TMS will need to be able to react to inquiries for *Identity*-information at any time, even when the owner of the mobile device is not actively using a *Service Provider*. Consequently, the device on which the TMS is deployed must be always connected to the internet (e.g. due to CA16.1-3, CA25.3), which is unlikely for mobile devices (e.g. due to the lack of available cellular networks, increased power consumption, etc.). The *Customer Attribute* CA14, provided by Cameron (2005, p.6), states that the *Owner* using the TMS must be warned if he selects an *Identity Provider* that tracks internet behaviour. Additionally, CA29.2 states that the TMS must support different roles for its users, which directly map to different *Identities* (e.g. a user can have an employer-, a private- and a public role or *Identity*). These CAs and the corresponding preliminary requirements are not relevant for the development of the TMS. This is because Cameron (2005) assumes that *Owners* use the TMS to manage *Identities* and *Credentials* referring to themselves instead to their *Things*. However, the TMS acts as a *Service Provider* for *Owners and* not as an identity provider. The services offered by the TMS include the management of *Things* and their corresponding *Identities* which are then used or shared with other *Service Providers*. It is not intended that the Owners manage their *Identities* (themselves)[14]. The fact that the *TMS* acts as a *Service Provider* for *Owners* is also the reason why CA30.4, which states that the TMS should not be tied to a single *Identity Provider*, is not relevant for the development of the TMS.

[14] Although it has been stated in section 4.3.2 that a *Thing* can be considered as an *Identity* referring to an *Owner*, it must be noted that the TMS is only intended to manage *Identities* that transitively refer to the *Owner´s Entity*.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Table 2: *Customer Attributes* and a first set of preliminary requirements for the TMS (own listing)

CA Req. ID	Description	Requirement	
CA1	*"A solution, which seems quite obvious, is simply to let users store identifiers and credentials from different service providers in a single tamper resistant hardware device which could be a smart card or some other portable personal device." (Jøsang & Pope 2005, p.7)*	**CA1.1**	An *Owner* must be able to store *Identifiers* and *Credentials* in the TMS.
		CA1.2	The TMS must be able to handle different types of *Credentials* for different *Service Providers*.
		CA1.3	The TMS must be tamper-resistant.
		CA1.4	The TMS must be able to be deployed on portable devices.
CA2	*"Because its main purpose would be authentication, the device can be called a personal authentication device (PAD)." (Jøsang & Pope 2005, p.7)*	**CA2.1**	The TMS must only provide authentication and necessarily related services.
CA3	*"The user must authenticate himself to the PAD, e.g. with a PIN, before the PAD can be used for authentication purposes." (Jøsang & Pope 2005, p.7)*	**CA3.1**	The TMS must provide authentication mechanisms for *Owners*.
CA4	*"A more advanced solution could be to connect the PAD to the client platform via a communication channel such as bluetooth or wireless LAN, or to let the PAD communicate directly with the server through a secondary channel. This would allow the PAD to be fully integrated into the authentication process. This is described in more detail in Sec.6" (Jøsang & Pope 2005, p.8)*	**CA4.1**	The TMS must be able to communicate with different *Service Providers*.
		CA4.2	The TMS must support various communication protocols and technologies.
		CA4.3	The devices a TMS is deployed on should support LAN, IPv4, IPv6, Wi-Fi, BLE.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

CA5	*"The functionality of a PAD could be integrated into other devices such as a mobile phone or personal digital assistant (PDA) which many people carry already. Using a mobile phone would also allow advanced solutions such as registration and challenge-response authentication through a mobile secondary channel." (Jøsang & Pope 2005, p.8)*	**CA5.1**	The TMS could be able to be deployed on portable devices.
		CA5.2	The TMS must be able to communicate with *Service Providers* (e.g. to perform authentication / validation automatically).
CA6	*"With a PAD connected to the client platform, virtual SSO solutions are possible. This could be implemented by letting the PAD automatically authenticate itself on behalf of the user as long as the PAD is connected to the client platform." (Jøsang & Pope 2005, p.8)*	**CA6.1**	The TMS must be able to automatically authenticate with a *Service Provider*.
CA7	*"The PAD should be under the control of the user, and not under the control of the identifier providers, the credential issuers or the service providers." (Jøsang & Pope 2005, p.8)*	**CA7.1**	The TMS should be under the *Owner's* control (e.g. hosted on his resources, only he has access, etc.).
CA8	*"In order to gain full advantage of the PAD, it should be a general security device capable of handling many types of identities and credentials. Some level of standardisation, such as that described in the Personal Transaction Protocol [7], might be needed for that to be practical." (Jøsang & Pope 2005, p.8)*	**CA8.1**	The TMS should support as many different authentication protocols and technologies as possible.
		CA8.2	The TMS could use a standardised protocol for transmitting *Identity* related information, if available.
CA9	*"No one is as pivotal to the success of the identity metasystem as the individual who uses it. The system must first of all appeal by means of convenience and simplicity. But to endure, it must earn the users trust above all." (Cameron 2005, p.6)*	**CA9.1**	The TMS must be simple to use.
		CA9.2	The TMS must be convenient to use.
		CA9.3	*Owners* must trust the TMS.
CA10	*"The system must be designed to put the user in control - of what digital identities are used, and what information is released." (Cameron 2005, p.6)*	**CA10.1**	Owners must always give their consent before actions regarding their *Identities* are performed (e.g. sharing, publishing).
		CA10.2	Owners must decide which *Characteristics* or *Identifiers* are shared with other third parties.

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

CA11	*"The system must also protect the user against deception, verifying the identity of any parties who ask for information." (Cameron 2005, p.6)*	**CA11.1**	The TMS must verify the *Identity* of third parties inquiring the *Identity* a *Thing*.
CA12	*"Should the user decide to supply identity information, there must be no doubt that it goes to the right place." (Cameron 2005, p.6)*	**CA12.1**	The TMS must ensure that *Identities are* shared only with certified third parties.
CA13	*"And the system needs mechanisms to make the user aware of the purposes for which any information is being collected" (Cameron 2005, p.6)*	**CA13.1**	The TMS must inform the *Owner* about which *Identity* of which of his *Things* is used by which party for which purpose.
CA14	*"The system must inform the user when he or she has selected an identity provider able to track internet behaviour" (Cameron 2005, p.6)*	**CA14.1**	The TMS must inform the *Owner* when the *Owner* selected an Identity Provider that tracks his internet behaviour.
CA15	*"Further, it must reinforce the sense that the user is in control regardless of context, rather than arbitrarily altering its contract with the user." (Cameron 2005, p.6)*	**CA15.1**	An *Owner* must give his consent regardless of the context or consequences of allowing or denying sharing an *Identity* of one of his *Things*.
		CA15.2	An *Owner* must be asked for his consent regardless of the context of an inquiry in general.
CA16	*"The Law of User Control and Consent allows for the use of mechanisms whereby the metasystem remembers user decisions, and users may opt to have them applied automatically on subsequent occasions." (Cameron 2005, p.6)*	**CA16.1**	An *Owner* must be asked for his consent before every action taken by the TMS.
		CA16.2	The TMS can be able to store or remember an *Owner's* decision regarding specific actions.
		CA16.3	The TMS can be able to automatically apply decisions/ automatically act when the *Owner* previously has allowed it to.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

CA17	*"To mitigate risk, it is best to acquire information only on a "need to know" basis, and to retain it only on a "need to retain" basis. By following these practices, we can ensure the least possible damage in the event of a breach." (Cameron 2005, p.7)*	**CA17.1**	The TMS must only retain the minimal amount of *Identity*-information.
		CA17.2	The TMS must only acquire the minimal amount of *Identity*-information.
		CA17.3	The TMS must only use or provide *Identity*-information in scenarios that inevitably require the *Identity*-information.
CA18	*"At the same time, the value of identifying information decreases as the amount de- creases. A system built with the principles of information minimalism is therefore a less attractive target for identity theft, reducing risk even further" (Cameron 2005, p.7)*	**CA18.1**	The TMS must store as little *Identity*-information as possible.
CA19	*"The concept of "least identifying information" should be taken as meaning not only the fewest number of claims, but the information least likely to identify a given individual across multiple contexts. For example, if a scenario requires proof of being a certain age, then it is better to acquire and store the age category rather than the birth date." (Cameron 2005, p.7)*	**CA19.1**	Wherever possible, the TMS should select the *Characteristics or Identifiers* that satisfy the needs of the inquirer and that can least likely be used to denote the *Identity* of an *Owner*.
CA20	*"We can also express the Law of Minimal Disclosure this way: aggregation of identifying information also aggregates risk. To minimize risk, minimize aggregation." (Cameron 2005, p.7)*	**CA20.1**	The TMS must store the least amount of *Identity*-information as possible to provide the desired services.
CA21	*"Digital identity systems must be de- signed so the disclosure of identifying information is limited to parties* *having a necessary and justifiable place in a given identity relationship." (Cameron 2005, p.7)*	**CA21.1**	The TMS must verify if a party has a necessary / justifiable claim for accessing *Identity*-information.
CA22	*"The identity system must make its user aware of the party or parties with whom she is interacting while sharing information." (Cameron 2005, p.7)*	**CA22.1**	The TMS must provide *Identity*-information about other parties that want to access the *Identity*-information managed by the TMS.
CA23	*"The justification requirements apply both to the subject who is disclosing information and the relying party who depends on it." (Cameron 2005, p.7)*	**CA23.1**	The *Identities* used to access *Service Providers* must be justifiable / necessary. E.g. "Official" *Identities* must not be necessary when wanting to access a private service (e.g. family wiki).

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

CA24	*"We know from the law of control and con- sent that the system must be predictable and "translucent" in order to earn trust. But the user needs to understand who she is dealing with for other reasons, as we will see in law six (human integration)." (Cameron 2005, p.7)*	**CA24.1**	An *Owner* must always be informed about who (e.g. third parties) he is dealing with.
		CA24.2	The parties involved in an authentication process must always be presented / shown.
CA25	*"Every party to disclosure must provide the disclosing party with a policy statement about information use. This policy should govern what happens to disclosed information. One can view this policy as defining "delegated rights" issued by the disclosing party." (Cameron 2005, p.8)*	**CA25.1**	A third party wanting to access *Identity*-information must provide reasons / a policy that states what the information is used for.
		CA25.2	An *Owner* using the TMS must be asked for his consent when a party wants to access an *Identity* of one of his *Things*.
		CA25.3	A TMS should be able to automatically decide if *Identity*-information is shared or not, based on the policies if the *Owner* wishes so.
CA26	*"A universal identity system must support both "omnidirectional" identifiers for use by public entities and "unidirectional" identifiers for use by private entities, thus facilitating discovery while preventing unnecessary re- lease of correlation handles" (Cameron 2005, p.8)*	**CA26.1**	A TMS must allow the *Owner* to select omni- and uni-directional *Identifiers* or *Identities*.
		CA26.2	Omnidirectional *Identifiers* or *Identities* must be public, invariant and well know.
		CA26.3	Unidirectional *Identifiers* or *Identities* must remain private and only be used when the *Owner* wishes so.
		CA26.4	Unidirectional *Identifiers* or *Identities* shall be used in private communication contexts.
		CA26.5	Omnidirectional *Identifiers* or *Identities* must be discoverable and "emit" or act as beacon for *Identity* (of the respective *Entity*)
		CA26.6	Omnidirectional *Identifiers* or *Identities* shall be used in public communication / certification scenarios.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

CA27	*"A universal identity system must channel and enable the inter-working of multiple identity technologies run by multiple identity providers" (Cameron 2005, p.9)*	**CA27.1**	A TMS must support various kinds of representations of *Identities*.
		CA27.2	A TMS must be able to interface with various types of operators / *Service Providers*.
CA28	*"But in many cultures, employers and employees would not feel comfortable using government identifiers to log in at work. A government identifier might be used to con- vey taxation information; it might even be required when a person is first offered employment. But the context of employment is sufficiently autonomous that it warrants its own identity, free from daily observation via a government-run technology." (Cameron 2005, p.9)*	**CA28.1**	Depending on the context or scenario, an *Owner* must be able to choose which *Identity* he uses / provides to a *Service Provider.*
		CA28.2	*Identities* must have different characteristics and must be able to be assigned to different *Service Providers*.
CA29	*"A universal system must embrace differentiation, while recognizing that each of us is simultaneously - in different contexts - a citizen, an employee, a customer, a virtual personal." (Cameron 2005, p.9)*	**CA29.1**	A TMS must be able to distinguish between contexts (e.g. based on the *Service Provider* which inquires *Identity*-information).
		CA29.2	A TMS must support different roles for its *Owners*, which can be customers, employees, citizens, etc. at the same time.
CA30	*"This demonstrates, from yet another angle, that different identity systems must exist in a metasystem. It implies we need a simple encapsulating protocol (a way of agreeing on and transporting things). We also need a way to surface information through a unified user experience that allows individuals and organizations to select appropriate identity providers and features as they go about their daily activities." (Cameron 2005, p.9)*	**CA30.1**	A TMS must use a standardised communication protocol for transmitting *Identity*-information.
		CA30.2	A TMS must display / present *Identities* in a similar manner across different *Identity Domains* / application contexts.
		CA30.3	*Owners* must be able to select *Identity Providers* through a TMS.
		CA30.4	A TMS must not be tied to a specific *Identity Provider*.
		CA30.5	A TMS must be able to communicate with other TMS.

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

CA31	*"The universal identity metasystem must not be another monolith. It must be polycentric (federation implies this) and also polymorphic (existing in different forms). This will allow the identity ecology to emerge, evolve and self-organize." (Cameron 2005, p.10)*	**CA31.1**	A TMS embedded into a metasystem should be polycentric and polymorphic, thus a TMS should be able to communicate with different kinds of TMS.
CA32	*"Carl Ellison and his colleagues have coined the term „ceremony" to describe interactions that span a mixed network of human and cybernetic system components – the full channel from web server to human brain. A ceremony goes beyond cyber protocols to ensure the integrity of communication with the user.* *This concept calls for profoundly changing the user´s experience so it becomes predictable and unambiguous enough to allow for informed decisions."(Cameron 2005, p.10)*	**CA32.2**	A TMS must inform the *Owner* unambiguously regarding the *Identities* the *Owner* is using and dealing with while he accessing *Identity*-information / sharing *Identity*-information with *Service Providers*.
CA33	*"Since the identity system has to work on all platforms, it must be safe on all platforms. The properties that lead to its safety can't be based on obscurity or the fact that the underlying platform or software is unknown or has a small adoption."(Cameron 2005, p.10)*	**CA33.1**	A TMS should be platform agnostic.
		CA33.2	The *Identity*-information, transmitted from the digital system to the *Owner* must be reliable.
CA34	*"The unifying identity metasystem must guarantee its users a simple, consistent experience while enabling separation of contexts through multiple operators and technologies."(Cameron 2005, p.10)*	**CA34.1**	A TMS must be able to distinguish between application contexts (e.g. based on different *Service Providers)*.
		CA34.2	A TMS must support the usage of different *Identities* for different contexts.
CA35	*"To make this possible, we must "thingify" digital identities – make them into "things" the user can see on the desktop, add and delete, select and share. How usable would today´s computers be had we not invented icons and lists that consistently represent folders and documents? We must do the same with digital identities."(Cameron 2005, p.11)*	**CA35.1**	A TMS must present/ display an *Identity* in a way the *Owner* understands. An *Identity* should be an object with properties and applications / instances in specific contexts (e.g. browsing, personal, community, etc.).

The CAs and corresponding preliminary requirements listen in The first step of the development process applying *Axiomatic Design* consists of mapping *Functional Requirements* or deriving them from the *Customer Domain* (see Figure 24). The set of *Customer Attributes* (CA) that will be used to derive the *Functional Requirements* (FR) are provided by Jøsang and Pope (2005) and Cameron (2005). The *Personal Authentication Device* (PAD) described by Jøsang and Pope (2005) (see Figure 5) will be used

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

to retrieve requirements for the *Thing Management System*. The PAD is based on the idea that *Service Providers*, in terms of IDM, generally have access to systems that allow the automated management of *Identities*, while users do not use or have access to such systems. Jøsang and Pope (2005) discuss that the growing number of *Service Providers* a user can and will consume might lead to security and usability issues when users need to manage these identities manually (e.g. by memorising credentials for each *Service Provider)*. Consequently, the concept of *Federated Identity Management Models* was introduced which in theory only requires a single set of credentials for the users to memorise or manage. However, Jøsang and Pope (2005) argue that if users only needed to manage a single set of credentials this would imply some sort of global federated *Identity Domain*, which is unlikely to be feasible. This is because due to different requirements regarding the characteristics or *Identifiers* making up an *Identity* across different *Identity Domains (e.g.* different legal or security requirements) (Jøsang & Pope 2005). To address the issues of poor usability regarding the management of identities, Jøsang and Pope (2005) present the PAD, which is a device or service controlled by a single user that securely stores an arbitrary number of credentials which are linked to corresponding *Service Providers.* Consequently, a user only needs to remember a single set of credentials for authentication with his own PAD to be able to authenticate with every other *Service Provider* he uses. Jøsang and Pope (2005) suggest that this can create a so called "virtual single-sign-on" environment, where a user is authenticated by single set of credentials across multiple *Service Providers*. The different sets of credentials for each *Service Provider* are handled by the PAD, thus the prefix "virtual" single-sign-on. Furthermore, the PAD can be backwards-compatible and can be implemented in every existing *Service Provider´s* authentication framework because it only manages the credentials or *Identifiers* and not the authentication (e.g. it can be interpreted as a database of a user´s credentials) (2005).

The descriptions provided by Jøsang and Pope are listed in Table 2. These were extracted from the description of the PAD provided by Jøsang and Pope (2005) and will be used as *Customer Attributes* from the *Customer Domain*. In addition to the requirements extracted from the descriptions of the PAD, the *"Laws of Identity"* described by Cameron (2005), which are briefly described in section 2.4, will be used as additional CAs for the *Customer Domain*. These seven "*Laws of Identity"* will be especially useful for generating requirements for the *Thing Management System* because the laws are addressing *Identity Management Systems* in general (Cameron 2005). With *Things* being both *Identity* and *Service Provider* (see section 4.3.2) and with the *Thing Management System* aiming to simplify the management of *Things*, the requirements for an *Identity Management System*, provided in the form of the *"Laws of Identity"*, can be also be applied to the *Thing Management System*. These requirements are also listed in Table 2.

The *Customer Attributes*, given in the form of the statements and descriptions provided by Jøsang and Pope (2005) as well as Cameron (2005), essentially express the customer needs and expectations that the complete design, which in this case is the TMS, must fulfil. These expressions and expectations are likely to be vague and unstructured (e.g. extracted from interviews or other informal specifications) and thus need further refinement and analysis in order to be able to map the CAs to *Functional Requirements*

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

(FR). This refinement or mapping is guided by the *information axiom* and *independence axiom* provided by the *axiomatic design* approach. To illustrate this mapping between the *Customer Domain* and the *Functional Domain* (see Figure 24), Table 2 lists the CAs as well as some preliminary requirements and Table 3 lists the actual FRs and the corresponding mapping between CAs and FRs. The following paragraphs will further elaborate on the structure of the mentioned tables, the CAs, the preliminary requirements, the FRs and the notable exceptions of them.

The first column of Table 2 contains unique identifiers assigned to CAs extracted from either Jøsang and Pope (2005) or Cameron (2005) which are shown in the second column. The third column lists the preliminary set requirements which have been extracted from these CAs. A preliminary requirement in column three addresses one or more actors or components and defines an expected behaviour or functionality for these actors or components. In addition, each requirement is considered as either a "must-have", a "should-have" or a "could-have". Whereas a "must-have" requirement is critical for successfully fulfilling the expectations for a system and a "could-have" requirement merely provides supplementary functionalities that are not critical for the system (Bradner 1997; Clegg & Barker 1994). Furthermore, each requirement is assigned with a unique identifier derived from the corresponding CA´s identifier. This additional mapping between a single CA and the corresponding preliminary requirements is done because a CA could incorporate one or more preliminary requirements (e.g. CA1, CA26, i.a.).

The preliminary set of requirements listed in Table 2, which has been directly derived or extracted from the statements or CAs, still contains duplicates and is generally unstructured. Furthermore, some preliminary requirements are either non-functional requirements (e.g. CA1.3, CA9.1 – CA9.3, i.a.) or must be further decomposed (e.g. CA3.1, CA8.1, i.a.) to satisfy the *independence* and *information axiom* described in section 2.2.

The preliminary requirement CA5.1, provided by Jøsang and Pope (2005, p.8), states that the TMS could be able to be deployed on a portable device. However, considering the intended application of the PAD or TMS, this requirement is not applicable. This is because Jøsang and Pope (2005) suggest that the PAD only stores the *Credentials* and *Identities* of a single user and that the PAD is only used when the user actually needs these *Credentials* (e.g. when he needs to authenticate with a *Service Provider)*. In contrast, the TMS will need to be able to react to inquiries for *Identity*-information at any time, even when the owner of the mobile device is not actively using a *Service Provider*. Consequently, the device on which the TMS is deployed must be always connected to the internet (e.g. due to CA16.1-3, CA25.3), which is unlikely for mobile devices (e.g. due to the lack of available cellular networks, increased power consumption, etc.). The *Customer Attribute* CA14, provided by Cameron (2005, p.6), states that the *Owner* using the TMS must be warned if he selects an *Identity Provider* that tracks internet behaviour. Additionally, CA29.2 states that the TMS must support different roles for its users, which directly map to different *Identities* (e.g. a user can have an employer-, a private- and a public role or *Identity*). These CAs and the corresponding preliminary requirements are not relevant for the development of the TMS. This is because Cameron (2005) assumes that *Owners* use the TMS to manage *Identities* and *Credentials*

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

referring to themselves instead to their *Things*. However, the TMS acts as a *Service Provider* for *Owners and* not as an identity provider. The services offered by the TMS include the management of *Things* and their corresponding *Identities* which are then used or shared with other *Service Providers*. It is not intended that the Owners manage their *Identities* (themselves). The fact that the *TMS* acts as a *Service Provider* for *Owners* is also the reason why CA30.4, which states that the TMS should not be tied to a single *Identity Provider*, is not relevant for the development of the TMS.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Table 2 are mapped to *Functional Requirements*, which are listed in Table 3. The first column of Table 3 contains unique identifiers for each requirement while the second column contains the associated *Customer Attributes* from The first step of the development process applying *Axiomatic Design* consists of mapping *Functional Requirements* or deriving them from the *Customer Domain* (see Figure 24). The set of *Customer Attributes* (CA) that will be used to derive the *Functional Requirements* (FR) are provided by Jøsang and Pope (2005) and Cameron (2005). The *Personal Authentication Device* (PAD) described by Jøsang and Pope (2005) (see Figure 5) will be used to retrieve requirements for the *Thing Management System*. The PAD is based on the idea that *Service Providers*, in terms of IDM, generally have access to systems that allow the automated management of *Identities*, while users do not use or have access to such systems. Jøsang and Pope (2005) discuss that the growing number of *Service Providers* a user can and will consume might lead to security and usability issues when users need to manage these identities manually (e.g. by memorising credentials for each *Service Provider)*. Consequently, the concept of *Federated Identity Management Models* was introduced which in theory only requires a single set of credentials for the users to memorise or manage. However, Jøsang and Pope (2005) argue that if users only needed to manage a single set of credentials this would imply some sort of global federated *Identity Domain*, which is unlikely to be feasible. This is because due to different requirements regarding the characteristics or *Identifiers* making up an *Identity* across different *Identity Domains (e.g.* different legal or security requirements) (Jøsang & Pope 2005). To address the issues of poor usability regarding the management of identities, Jøsang and Pope (2005) present the PAD, which is a device or service controlled by a single user that securely stores an arbitrary number of credentials which are linked to corresponding *Service Providers.* Consequently, a user only needs to remember a single set of credentials for authentication with his own PAD to be able to authenticate with every other *Service Provider* he uses. Jøsang and Pope (2005) suggest that this can create a so called “virtual single-sign-on” environment, where a user is authenticated by single set of credentials across multiple *Service Providers*. The different sets of credentials for each *Service Provider* are handled by the PAD, thus the prefix “virtual” single-sign-on. Furthermore, the PAD can be backwards-compatible and can be implemented in every existing *Service Provider´s* authentication framework because it only manages the credentials or *Identifiers* and not the authentication (e.g. it can be interpreted as a database of a user´s credentials) (2005).

The descriptions provided by Jøsang and Pope are listed in Table 2. These were extracted from the description of the PAD provided by Jøsang and Pope (2005) and will be used as *Customer Attributes* from the *Customer Domain*. In addition to the requirements extracted from the descriptions of the PAD, the *“Laws of Identity”* described by Cameron (2005), which are briefly described in section 2.4, will be used as additional CAs for the *Customer Domain*. These seven “*Laws of Identity”* will be especially useful for generating requirements for the *Thing Management System* because the laws are addressing *Identity Management Systems* in general (Cameron 2005). With *Things* being both *Identity* and *Service Provider* (see section 4.3.2) and with the *Thing Management System* aiming to simplify the management of *Things*, the requirements for an *Identity Management System*, provided in the form of the *“Laws of*

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Identity", can be also be applied to the *Thing Management System*. These requirements are also listed in Table 2.

The *Customer Attributes*, given in the form of the statements and descriptions provided by Jøsang and Pope (2005) as well as Cameron (2005), essentially express the customer needs and expectations that the complete design, which in this case is the TMS, must fulfil. These expressions and expectations are likely to be vague and unstructured (e.g. extracted from interviews or other informal specifications) and thus need further refinement and analysis in order to be able to map the CAs to *Functional Requirements* (FR). This refinement or mapping is guided by the *information axiom* and *independence axiom* provided by the *axiomatic design* approach. To illustrate this mapping between the *Customer Domain* and the *Functional Domain* (see Figure 24), Table 2 lists the CAs as well as some preliminary requirements and Table 3 lists the actual FRs and the corresponding mapping between CAs and FRs. The following paragraphs will further elaborate on the structure of the mentioned tables, the CAs, the preliminary requirements, the FRs and the notable exceptions of them.

The first column of Table 2 contains unique identifiers assigned to CAs extracted from either Jøsang and Pope (2005) or Cameron (2005) which are shown in the second column. The third column lists the preliminary set requirements which have been extracted from these CAs. A preliminary requirement in column three addresses one or more actors or components and defines an expected behaviour or functionality for these actors or components. In addition, each requirement is considered as either a "must-have", a "should-have" or a "could-have". Whereas a "must-have" requirement is critical for successfully fulfilling the expectations for a system and a "could-have" requirement merely provides supplementary functionalities that are not critical for the system (Bradner 1997; Clegg & Barker 1994). Furthermore, each requirement is assigned with a unique identifier derived from the corresponding CA´s identifier. This additional mapping between a single CA and the corresponding preliminary requirements is done because a CA could incorporate one or more preliminary requirements (e.g. CA1, CA26, i.a.).

The preliminary set of requirements listed in Table 2, which has been directly derived or extracted from the statements or CAs, still contains duplicates and is generally unstructured. Furthermore, some preliminary requirements are either non-functional requirements (e.g. CA1.3, CA9.1 – CA9.3, i.a.) or must be further decomposed (e.g. CA3.1, CA8.1, i.a.) to satisfy the *independence* and *information axiom* described in section 2.2.

The preliminary requirement CA5.1, provided by Jøsang and Pope (2005, p.8), states that the TMS could be able to be deployed on a portable device. However, considering the intended application of the PAD or TMS, this requirement is not applicable. This is because Jøsang and Pope (2005) suggest that the PAD only stores the *Credentials* and *Identities* of a single user and that the PAD is only used when the user actually needs these *Credentials* (e.g. when he needs to authenticate with a *Service Provider)*. In contrast, the TMS will need to be able to react to inquiries for *Identity*-information at any time, even when the owner of the mobile device is not actively using a *Service Provider*. Consequently, the device on which the TMS is deployed must be always connected to the internet (e.g. due to CA16.1-3, CA25.3),

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

which is unlikely for mobile devices (e.g. due to the lack of available cellular networks, increased power consumption, etc.). The *Customer Attribute* CA14, provided by Cameron (2005, p.6), states that the *Owner* using the TMS must be warned if he selects an *Identity Provider* that tracks internet behaviour. Additionally, CA29.2 states that the TMS must support different roles for its users, which directly map to different *Identities* (e.g. a user can have an employer-, a private- and a public role or *Identity*). These CAs and the corresponding preliminary requirements are not relevant for the development of the TMS. This is because Cameron (2005) assumes that *Owners* use the TMS to manage *Identities* and *Credentials* referring to themselves instead to their *Things*. However, the TMS acts as a *Service Provider* for *Owners and* not as an identity provider. The services offered by the TMS include the management of *Things* and their corresponding *Identities* which are then used or shared with other *Service Providers*. It is not intended that the Owners manage their *Identities* (themselves). The fact that the *TMS* acts as a *Service Provider* for *Owners* is also the reason why CA30.4, which states that the TMS should not be tied to a single *Identity Provider*, is not relevant for the development of the TMS.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Table 2 or other related *Functional Requirements*. The third column determines the severity of the requirements, which are again grouped into “must-have”, “should-have” and “could-have” (Bradner 1997; Clegg & Barker 1994). The fourth and last column contains the description of the *Functional Requirement.*

The FRs listed in Table 3 are grouped into six modules that encapsulate similar functionalities. These modules are *Access and Authentication*, *Management*, *Communication and Publishing*, *Provisioning,* and *Thing and Identity Representation* (see Figure 25). Each module is responsible for a specific and decoupled set of tasks and responsibilities. The modules have been created by semantically grouping similar CAs and corresponding preliminary requirements together. For example, the CAs CA1.3, CA3.1, C9.1, CA9.2 and CA2.1 state that the TMS need some means of authentication module that is tamper resistant, easy and convenient to use. It must be noted that some *Customer Attributes*, e.g. CA1.3, CA9.1, CA9.2, CA28.1, CA28.2, i.a., can be assigned to multiple modules at the same time. This is because these CAs are either non-functional requirements (e.g. CA1.3, CA9.1, CA9.2) or that they are generic in such a way that they address multiple issues in multiple modules (e.g. CA28.1, CA28.2). The modules created by this grouping are described in the following paragraphs.

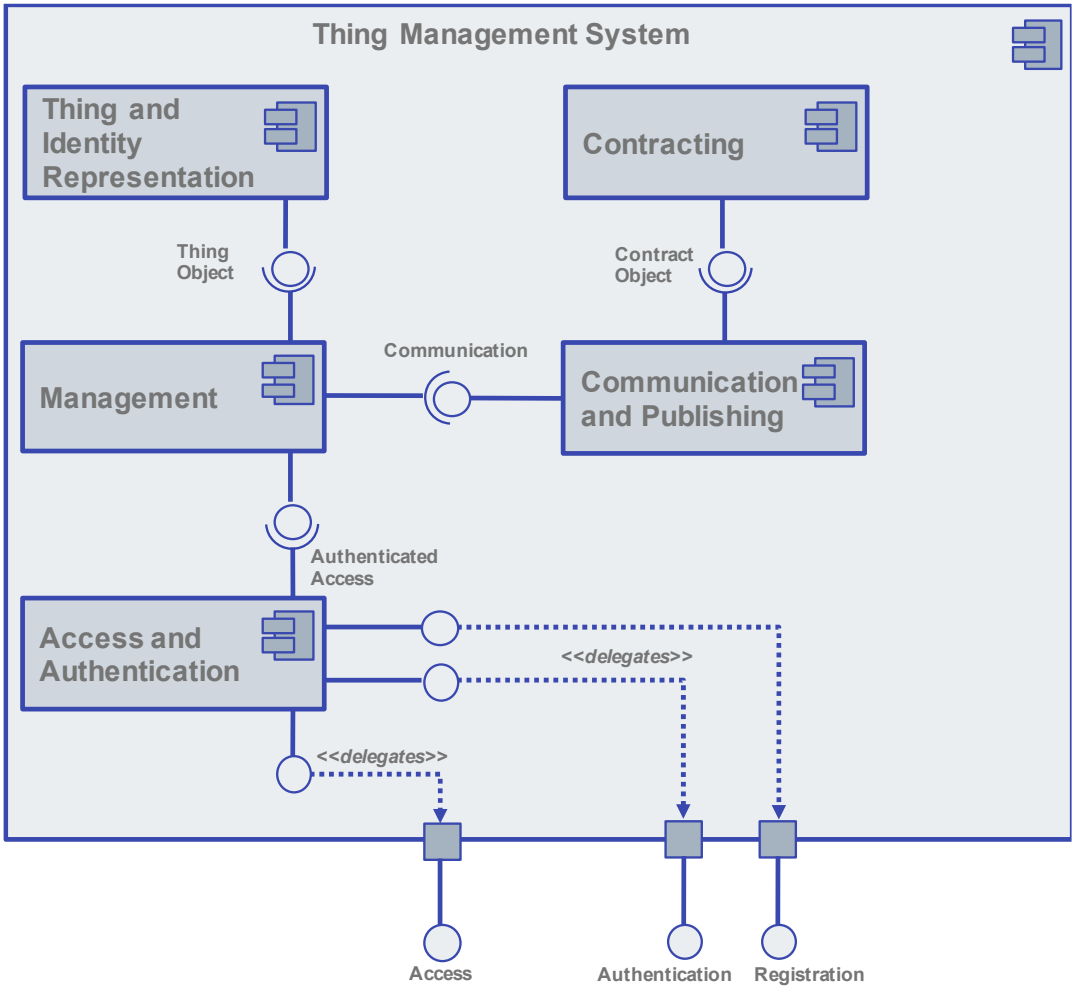


Figure 25: Component diagram for TMS based on FRs (own illustration)

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Access and Authentication

This module is responsible for allowing *Owners* to securely access the TMS by requiring them to authenticate themselves with an *Identity* consisting of *Characteristics* and *Credentials* defined by the *Identity Domain* of the TMS. Furthermore, this module provides functionalities to register new users or *Owners* with the TMS. During this process the *Credentials* and corresponding *Identity* referring to the user or *Owner* are created and securely stored. In order to ensure that only authenticated users have access to the TMS, the *Access and Authentication* module monitors every action and request that is handled by the TMS and validates that the respective issuers of the requests and actions are properly authenticated. Requests made by unauthenticated issuers are rejected. Issuers of request can be either *Owners*, *Things* or *Service Providers* (e.g. *Publishers)*.

Management

This module is responsible for storing and managing *Things* with their corresponding *Identities*. To fulfil its responsibilities, this module provides means of creating, listing, updating and deleting both *Things* and corresponding *Identities*. Users, which act as *Owners*, of the TMS must be able to create an arbitrary representation of a *Thing* and add *Identities* to this *Thing*. An *Identity* of a *Thing* must contain a reasonable amount of information to uniquely identify a *Thing* attached to the TMS. Furthermore, the *Owner* must be able to define access rules and restrictions for each *Thing* individually and the *Owner* must be able to specify if a single *Identifier* of an *Identity* is publicly available or if it must remain private. The *Management* module is also responsible for publishing *Identities* containing public *Identifiers* and allowing the *Owner* to assign *Identities* to *Service Providers* by forming contracts, which are provided by the *Contracting* as well as *Communication and Publishing* module (see section 4.2.2). Additionally, this module allows *Owners* to keep track which *Identities* are assigned to which *Service Provider*.

Communication and Publishing

The *Management* module uses the *Communication and Publishing* module when assigning *Identities* to *Service Providers*. Thus, this module´s main responsibility is to enable and control communication between the TMS and other third parties which are usually *Service Providers*. This module acts as an adapter to allow the TMS to communicate with a variety of different *Service Providers* via various communication technologies and protocols. The module abstracts the communication details (e.g. data format, communication protocol, etc.), hence the *Management* module is capable of assigning *Identities* to different types of *Service Providers* without issues. Furthermore, the *Communication and Publishing* module is responsible for handling inquiries for *Identity*-information and obtaining the *Owners* consent. Before any action is performed (e.g. publishing a *Thing´s Identity*), this module requests the consent of the *Owner*. The “request for consent” contains information regarding the *Identity* of the inquirer and the usage policy of the requests *Identity*-information and is presented to the *Owner*. Subsequently, the *Owner* can either accept or decline the inquiry and save this decision so that the TMS can automatically perform the same decision if the same inquirer asks for the same *Identity* with the same policy again.

Thing and Identity Representation

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

While the *Management* module is responsible for managing *Things*, this module is responsible for a uniform representation of *Things* across different contexts and for different *Service Providers*. Like the *Management* module, the *Thing and Identity representation* module acts as an adapter which abstracts the technical details (e.g. storing a *Thing´s* data, different *Identifiers* or *Characteristics*, data formats, etc.) of *Things* and provides a uniform representation of both *Things* and their corresponding *Identities*. Furthermore, this module is also responsible for uniformly representing the *Identities* of various third parties (e.g. *Service Providers)* during the "request for consent" handled by the *Communication and Publishing* module.

Provisioning

This module specifies how the TMS should be able to be provisioned. Due to the heterogenetic nature of IoT in general (see section 1.1), the TMS should be platform agnostic and should be able to be deployed in a variety of different environments. In order to accomplish this, the TMS could be designed as a web-service that provides a number of endpoints and services, which are represented by the modules *Access and Authentication*, *Management* and *Communication and Publishing*. Subsequently, the TMS could either be deployed on the infrastructure provided by an *Owner* (on premise) or be deployed as a *Service Provider* in the internet, where many *Owners* share use the TMS together. However, to ensure privacy and the respective *Owners* full control over the system, the TMS should properly encapsulate the data of each *Owner* (e.g. by providing multitenancy functionalities).

Contracting

Similar to the *Thing and Identity Representation* module, the *Contracting* module abstracts the details of assigning and forming contracts between the TMS and third parties (e.g. *Service Providers),* thus acting as an adapter. This module is used by the *Communication and Publishing* module during the process of assigning an *Identity* to a *Service Provider*. A contract contains an *Identity* as well as a set of access rules and restrictions which have been defined using the *Management* module. Essentially, this module is responsible converting the *Identity* and the set of access rules and restrictions into a transferable format that can be handled by the *Service Provider* with whom a contract is to be formed. Because the TMS must be able to communicate with a variety of different third parties, this module should support a variety of transferable formats for contracts.

Having described each module in general, Table 3 lists the detailed functional requirements for each module.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Table 3: *Functional Requirements* and *Customer Attribute* mapping for the TMS (own listing)

FR Req. ID		Related Req. ID	Severity	Description
FR1		CA1.3 CA3.1 CA9.1 CA9.2 CA2.1	Must	The TMS must only allow authenticated *Owners* to access the system.
	FR1.1	CA1.1	Must	The TMS must be able to store an *Owner´s Credentials*.
	FR1.2	CA9.3	Must	The TMS must provide an authentication mechanism and function for *Owners*, denying unauthenticated *Owners* access.
	FR1.3	CA17.1 CA18.1 CA19.1 CA20.1	Must	The TMS must store the smallest possible set of *Credentials* to authenticate an *Owner.* The *Credentials* must be selected in such a way that they cannot be used easily to infer the *Owners Identity*.
	FR1.4	CA1.3 CA3.1	Must	The TMS must provide a function to initially register an *Owner*.
	FR1.5	CA1.3 CA3.1	Must	The TMS must provide a function for authenticating with the system.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

FR2		FR5 CA1.2 CA8.1 CA9.1 CA9.2 CA17.2 CA19.1 CA28.1 CA28.2	Must	The TMS must be able to manage different types of *Things and Identities* for different *Service Providers*.
	FR2.1	CA17.2 CA18.1 CA19.1 CA28.1 CA28.2	Must	The *Identity*-information stored in the TMS must not require unnecessary information. It must be able to store a minimal amount of *Identity*-information to uniquely identify or assign an *Identity*.
	FR2.2	CA1.2		The TMS must provide a function to create, list, view, update and delete *Things*.
	FR2.3	CA1.2		The TMS must provide a function to create, list, view, update and delete *Identities for each Thing*.
	FR2.4	CA5.2 CA28.1 CA28.2 CA34.2 CA10.2		The TMS must be able to assign an *Identity* of a managed *Thing* to a *Service Provider*.
	FR2.5	CA26.1 CA28.1 CA28.2		Each *Characteristic* of a *Thing´s Identity* must be defined as either *public* or *private*.
	FR2.6	CA26.1 CA26.3		A private *Characteristic* must only be used in private communication contexts.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

	FR2.7	CA26.1 CA26.2		A public *Characteristic* can be used on public communication contexts.
	FR2.8	CA26.5 CA26.6		An *Identity* consisting of at least one public *Characteristic* must be discoverable by *Service Providers*.
	FR2.9	FR2.8		A TMS must expose information regarding *Identities* consisting of *public Characteristics*.
	FR2.10	FR2.4 CA13.1		An *Owner* must be able to see which *Identities* of which of *Thing* are used or assigned to which *Service Provider.*
FR3		CA4.1 CA1.3 CA30.5	Must	The TMS must be able to securely communicate with different *Service Providers* and other TMS.
	FR3.1	CA4.2 CA4.3 CA27.2 CA31.1		The TMS must support various communication technologies and protocols.
	FR3.2	FR3.1 CA31.1		The TMS´s must provide a modularised communication that allows polymorphic communication with other systems.
	FR3.3	CA6.1 FR3.2 CA9.1 CA9.2		The TMS must be able to automatically authenticate with a *Service Provider*.
	FR3.4	FR3.3 FR3.2 FR4.1 CA8.1 FR2		The TMS should provide as many different authentication protocols as possible.

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

	FR3.5	FR3.3 FR3.2 FR3.1 CA8.2 CA30.1		The TMS should support a standardised communication protocol, if one is available.
	FR3.6	FR3.3 CA9.3 CA10.1 CA15.1 CA15.2 CA16.1 CA24.1		Before any action is performed by the TMS, the *Owner* must be consented.
	FR3.7	FR3.6 CA16.2 CA16.3 CA25.3		The TMS can automatically authenticate with other systems, if the *Owner* wishes so.
	FR3.8	FR3.7 CA16.1 CA16.2 CA16.3		The *Owner* must be able to decide if and with which other systems the TMS can automatically communicate and which *Identity* is used. The decisions must be manageable and revocable.
	FR3.9	FR3.6 CA9.3 CA24.1 CA21.1 CA11.1 CA12.1 CA22.1 CA33.2		Each system or third party that communicates with the TMS must reliably authenticate itself.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

	FR3.10	FR3.9 CA24.2 CA32.2 CA30.2 CA35.1		The TMS must show/ provide the *Identity* of any third party requesting to inquire *Identity*-information from the TMS to the *Owner*.
	FR3.11	FR.10 CA23.1 CA17.3		A third party inquiring *Identity*-information from a TMS must state the reasons why the information is required.
	FR3.12	CA25.1		A third party inquiring *Identity*-information from a TMS must state what the *Identity*-information is to be used for. These usage principles must be provided in a form of a policy.
	FR3.13	FR3.9 FR3.10		The *Owner* must be able to accept or reject an inquiry from a third party.
	FR3.14	FR3.7 FR3.8 CA25.3		The TMS can automatically respond and share *Identity*-information with authenticated third parties if the *Owner* wishes so.
	FR3.15	FR2.6 FR.4.14 CA29.1 CA34.1		The TMS must be able to decide if *Identity*-information is shared based on the authenticated third party, the justifications for the inquiry and the *Owners* previous decisions.
FR4		CA30.2 CA35.1 CA27.1		A TMS must display *Things* and their corresponding *Identities* in a similar manner across different contexts.
	FR4.1	FR2.3 FR4		A *Thing* must be able to have one or more *Identities*.
	FR4.2	FR4		An *Identity* must consist of an arbitrary number of *Characteristics/ Identifiers*.

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

	FR4.3	FR2.6 FR2.7 FR2.8		The *Characteristics* of each *Identity* corresponding to a *Thing* managed by the TMS must be manageable.
	FR4.4	FR4		The *Identity* of a *Thing* and any other third party must be displayed in a similar manner.
FR5		CA33.1 CA7.1		The TMS should be platform agnostic and either be hosted by or fully under control of the *Owner*.
	FR5.1	CA4.1 CA7.1 CA33.1		The TMS could be provided as a web application and either be hosted on the *Owner´s* resources or by a third party.
	FR5.2	CA4.1		The TMS could be accessible via mobile devices.
FR6		FR2.4		Upon assigning *Thing´s Identities* to *Service Providers*, contracts or agreements must be formed that mediate the subsequent usage or access to the corresponding *Identity´s* resources and services.
	FR6.1			An agreement between the TMS and a *Service Provider* regarding a *Thing´s Identity* must consist of an *Identity* and a set of rules and restrictions.
	FR6.2	FR6.1		The rules and restrictions of an agreement must be able to govern the usage of the corresponding *Thing´s* services (e.g. time based access, required compensations, etc.).

After having identified the *Customer* Attributes (see The first step of the development process applying *Axiomatic Design* consists of mapping *Functional Requirements* or deriving them from the *Customer Domain* (see Figure 24). The set of *Customer Attributes* (CA) that will be used to derive the *Functional Requirements* (FR) are provided by Jøsang and Pope (2005) and Cameron (2005). The *Personal Authentication Device* (PAD) described by Jøsang and Pope (2005) (see Figure 5) will be used to retrieve requirements for the *Thing Management System*. The PAD is based on the idea that *Service Providers*, in terms of IDM, generally have access to systems that allow the automated management of *Identities*, while users do not use or have access to such systems. Jøsang and Pope (2005) discuss that the growing number of *Service Providers* a user can and will consume might lead to security and usability issues when

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

users need to manage these identities manually (e.g. by memorising credentials for each *Service Provider)*. Consequently, the concept of *Federated Identity Management Models* was introduced which in theory only requires a single set of credentials for the users to memorise or manage. However, Jøsang and Pope (2005) argue that if users only needed to manage a single set of credentials this would imply some sort of global federated *Identity Domain*, which is unlikely to be feasible. This is because due to different requirements regarding the characteristics or *Identifiers* making up an *Identity* across different *Identity Domains (e.g.* different legal or security requirements) (Jøsang & Pope 2005). To address the issues of poor usability regarding the management of identities, Jøsang and Pope (2005) present the PAD, which is a device or service controlled by a single user that securely stores an arbitrary number of credentials which are linked to corresponding *Service Providers.* Consequently, a user only needs to remember a single set of credentials for authentication with his own PAD to be able to authenticate with every other *Service Provider* he uses. Jøsang and Pope (2005) suggest that this can create a so called "virtual single-sign-on" environment, where a user is authenticated by single set of credentials across multiple *Service Providers*. The different sets of credentials for each *Service Provider* are handled by the PAD, thus the prefix "virtual" single-sign-on. Furthermore, the PAD can be backwards-compatible and can be implemented in every existing *Service Provider´s* authentication framework because it only manages the credentials or *Identifiers* and not the authentication (e.g. it can be interpreted as a database of a user´s credentials) (2005).

The descriptions provided by Jøsang and Pope are listed in Table 2. These were extracted from the description of the PAD provided by Jøsang and Pope (2005) and will be used as *Customer Attributes* from the *Customer Domain*. In addition to the requirements extracted from the descriptions of the PAD, the *"Laws of Identity"* described by Cameron (2005), which are briefly described in section 2.4, will be used as additional CAs for the *Customer Domain*. These seven "*Laws of Identity"* will be especially useful for generating requirements for the *Thing Management System* because the laws are addressing *Identity Management Systems* in general (Cameron 2005). With *Things* being both *Identity* and *Service Provider* (see section 4.3.2) and with the *Thing Management System* aiming to simplify the management of *Things*, the requirements for an *Identity Management System*, provided in the form of the *"Laws of Identity"*, can be also be applied to the *Thing Management System*. These requirements are also listed in Table 2.

The *Customer Attributes*, given in the form of the statements and descriptions provided by Jøsang and Pope (2005) as well as Cameron (2005), essentially express the customer needs and expectations that the complete design, which in this case is the TMS, must fulfil. These expressions and expectations are likely to be vague and unstructured (e.g. extracted from interviews or other informal specifications) and thus need further refinement and analysis in order to be able to map the CAs to *Functional Requirements* (FR). This refinement or mapping is guided by the *information axiom* and *independence axiom* provided by the *axiomatic design* approach. To illustrate this mapping between the *Customer Domain* and the *Functional Domain* (see Figure 24), Table 2 lists the CAs as well as some preliminary requirements and Table 3 lists the actual FRs and the corresponding mapping between CAs and FRs. The following

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

paragraphs will further elaborate on the structure of the mentioned tables, the CAs, the preliminary requirements, the FRs and the notable exceptions of them.

The first column of Table 2 contains unique identifiers assigned to CAs extracted from either Jøsang and Pope (2005) or Cameron (2005) which are shown in the second column. The third column lists the preliminary set requirements which have been extracted from these CAs. A preliminary requirement in column three addresses one or more actors or components and defines an expected behaviour or functionality for these actors or components. In addition, each requirement is considered as either a "must-have", a "should-have" or a "could-have". Whereas a "must-have" requirement is critical for successfully fulfilling the expectations for a system and a "could-have" requirement merely provides supplementary functionalities that are not critical for the system (Bradner 1997; Clegg & Barker 1994). Furthermore, each requirement is assigned with a unique identifier derived from the corresponding CA´s identifier. This additional mapping between a single CA and the corresponding preliminary requirements is done because a CA could incorporate one or more preliminary requirements (e.g. CA1, CA26, i.a.).

The preliminary set of requirements listed in Table 2, which has been directly derived or extracted from the statements or CAs, still contains duplicates and is generally unstructured. Furthermore, some preliminary requirements are either non-functional requirements (e.g. CA1.3, CA9.1 – CA9.3, i.a.) or must be further decomposed (e.g. CA3.1, CA8.1, i.a.) to satisfy the *independence* and *information axiom* described in section 2.2.

The preliminary requirement CA5.1, provided by Jøsang and Pope (2005, p.8), states that the TMS could be able to be deployed on a portable device. However, considering the intended application of the PAD or TMS, this requirement is not applicable. This is because Jøsang and Pope (2005) suggest that the PAD only stores the *Credentials* and *Identities* of a single user and that the PAD is only used when the user actually needs these *Credentials* (e.g. when he needs to authenticate with a *Service Provider)*. In contrast, the TMS will need to be able to react to inquiries for *Identity*-information at any time, even when the owner of the mobile device is not actively using a *Service Provider*. Consequently, the device on which the TMS is deployed must be always connected to the internet (e.g. due to CA16.1-3, CA25.3), which is unlikely for mobile devices (e.g. due to the lack of available cellular networks, increased power consumption, etc.). The *Customer Attribute* CA14, provided by Cameron (2005, p.6), states that the *Owner* using the TMS must be warned if he selects an *Identity Provider* that tracks internet behaviour. Additionally, CA29.2 states that the TMS must support different roles for its users, which directly map to different *Identities* (e.g. a user can have an employer-, a private- and a public role or *Identity*). These CAs and the corresponding preliminary requirements are not relevant for the development of the TMS. This is because Cameron (2005) assumes that *Owners* use the TMS to manage *Identities* and *Credentials* referring to themselves instead to their *Things*. However, the TMS acts as a *Service Provider* for *Owners and* not as an identity provider. The services offered by the TMS include the management of *Things* and their corresponding *Identities* which are then used or shared with other *Service Providers*. It is not intended that the Owners manage their *Identities* (themselves). The fact that the *TMS* acts as a *Service*

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Provider for *Owners* is also the reason why CA30.4, which states that the TMS should not be tied to a single *Identity Provider*, is not relevant for the development of the TMS.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Table 2) and having derived *Functional Requirements* from the *Customer Domain* (see Table 3), the *Axiomatic Design* approach requires the assignment of *Design Parameters (DP)* to the *Functional Requirements, which* must satisfy the respective requirements. For the design to be considered a "good design" in terms of the *Axiom Based Design*, the assigned DPs should to fulfil the *Independence Axiom*. This means the corresponding *Design Matrix* should be either a triangular matrix or a diagonal matrix (Suh & Do 2000). A triangular design matrix refers to a *decoupled* design, whereas a diagonal matrix refers to an *uncoupled* design (see section 2.2). When considering the previously defined modules as *Design Parameters* that should fulfil the corresponding *Functional Requirements* (e.g. the module *Management* should fulfil FR2) and considering the interdependencies between each of the modules described earlier, the *Design Matrix* shown in Figure 26 can be created. Note that the numbering of the FRs listed in Table 3 are based on the order in which they were mentioned in the documents they were extracted from. Hence, the numbering of the FRs bears no meaning for the *Design Matrix*. Suh (2000) states that a triangular *Design Matrix* denotes a decoupled design and fulfils the *Independence Axiom* when design sequence is correct. The rows of *Design Matrix* for the TMS shown in Figure 26 contains the FRs from Table 3 and columns are considered as mapped to the *Design Parameters*. According to the principles of the *Axiom Based Design*, each FR should have an assigned DP that fulfils only the FR. If a one-to-one mapping between FRs and DPs cannot be achieved, the design is either coupled or decoupled (Suh & Do 2000). During the description of the modules, which are considered as the *Design Parameters*, the interdependencies between the modules have been mentioned. These interdependencies are also contained in the *Design Matrix*. The matrix is filled with either a zero, which denotes no interdependency or with a non-zero value which indicates any kind of dependency (Suh & Do 2000). For example, the *Design Matrix* for the TMS denotes no dependency between the module *Provisioning* and any other module, while the module *Communication and Publishing* depends on all other modules. To fulfil the *Independence Axiom* when having a triangular *Design Matrix*, which denotes a decoupled design, the correct design sequence must be met.

	DP1	DP2	DP3	DP4	DP5	DP6
FR5	**X**	**X**	**X**	**X**	**X**	**X**
FR1	0	**X**	0	0	0	**X**
FR4	0	0	**X**	**X**	**X**	**X**
FR2	0	0	0	**X**	0	**X**
FR6	0	0	0	0	**X**	**X**
FR3	0	0	0	0	0	**X**

Figure 26: High level design matrix for the TMS (own illustration)

The numbering of the *Design Parameters* (see Figure 26) already denotes the design sequence. In order to fulfil the *Independence Axiom,* the following design sequence must be used. The first module to be further developed is the *Provisioning* module because it has no dependency to another module. The

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

next step in the sequence can either be the Access and *Authentication* module or the *Thing and Identity Representation* module, as both only rely on the *Provisioning* module. The third step of the sequence can either contain the Management module or the *Contracting* module. The last step of the design sequence consists of the *Communication and Publishing* module. Thus, the *Design Parameters* are assigned to the modules as follows.

- **DP1** implements the *Provisioning* module (FR5 ≡ DP1)
- **DP2** implements the *Authentication and Access* module (FR1 ≡ DP2)
- **DP3** implements the *Thing and Identity Representation* module (FR4 ≡ DP3)
- **DP4** implements the *Management* module (FR2 ≡ DP4)
- **DP5** implements the *Contracting* module (FR6 ≡ DP5)
- **DP6** implements the *Communication and Publishing* module (FR3 ≡ DP6)

With the functional requirements corresponding to the modules defined earlier still being fairly complex, the *Axiom Based Design* approach demands that these requirements are further decomposed, that for each of the new requirements a new *Design Parameter* is created, that a new more detailed *Design Matrix* is created and this matrix is again evaluated according to the *Independence Axiom* (Suh & Do 2000). However, as described in section 2.4, this thesis aims to develop a high-level architecture framework which focusses on the components external relations and dependencies as well as their individual responsibilities. Thus, the detailed inner workings of each component, which would be described and specified by further decomposing the *Functional Requirements* listed in Table 3, are not relevant for developing the high-level architecture framework.

In conclusion, the identification of a further component supporting *Owners* in managing their *Things* and the subsequent development of the *Thing Management System* that fulfils the derived requirements in this section complete the answer to RQ1.1 and RQ1.3. Thus, having answered RQ1.2 in section 4.2.1 as well as the initial parts of RQ1.1 and RQ1.3 in section 4.2.2, RO1 is achieved in total. Additionally, by transferring and mapping the concepts of *Identity Management* to architectural components of IoT using the DSR pattern *Problem Space Tools and Techniques* components both RQ2.1 and RQ2.2 have been answered and RO2 has been achieved.

4.3.4 Revising and discussing the Gateway´s Roles and Relations

Considering the *IoT Architecture Perspectives* which have been exemplarily visualised in Figure 20 and Figure 21 and considering the insights from section 4.3.2 that a *Thing* does not only act as a *Service Provider* but can also be interpreted as an *Identity*, the special role of the *Gateway* component becomes apparent. From the *Network IoT Architecture Perspective´s* point of view, the *Gateway´*s role responsibility is to provide a communication channel between a *Thing* and a *Publisher*. In this view, *Things* are willing to accept connections and communicate with anyone. Considering the *Organisational IoT Architecture Perspective,* the *Gateway* is considered as a business entity that collects and sells sensing information on its own behalf. Besides occasionally selling the sensed data to a *Publisher*, the

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Gateway has no organisational ties with the *Publisher* at all. Both, the *Network IoT Architectures Perspective* and the *Organisational IoT Architecture Perspective* assume that a *Thing* willingly communicates with any other component. However, considering section 4.3.2, where it was discussed that a *Thing* can also be interpreted as an *Identity* referring to its *Owner*, one must re-evaluate the assumption that a *Thing* willingly communicates with any other component. With *Things* being "organisationally" registered to a *Publisher* who then acts as a proxy between the *Thing* and *Service Providers*, it becomes apparent that a *Gateway* should also be related to a *Publisher* when the intention is to establish a communication between a *Thing* and a *Publisher*. In order to maintain the *Publisher´s* role as a proxy and thus the only way of accessing a *Thing´s* services, a *Gateway* must be able to act on behalf of the *Publisher* when establishing a network connecting with a *Thing*. This intended role of the *Gateway* is similar to the *Mobile Sensing Terminal Operator* suggested by Mizouni and El Barachi (2013) which has been described in section 4.2.1. With *Gateways* acting on behalf of *Publishers* to establish a communication with *Things*, a system for allowing *Gateways* to authenticate themselves as representatives of *Publisher*s must be devised (see Figure 27).

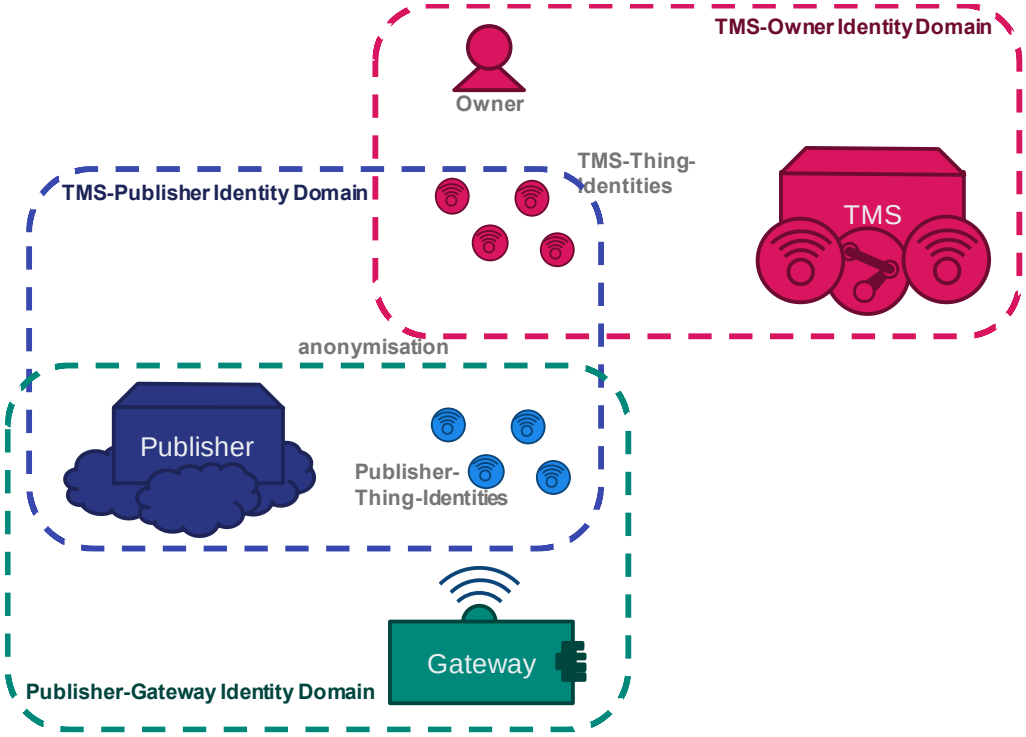


Figure 27: Revised roles and responsibilities of *Gateways* for the *Holistic IoT Architecture Framework* (own illustration)

In this system, the TMS, which has been described in the previous section, manages many *Identities* of *Things*. The specific *Characteristics* of these *Identities* are defined by the *Thing-Identity-Domain* of the TMS. This domain contains all *Identities* of all *Things* the TMS manages. These *Identities* may be able to directly refer to the *Owner* of its corresponding *Thing* and are shared or assigned to a *Publisher*. A *Publisher* maintains many *Things*, or their respective *Identities*, which are owned by many different *Owners.* The *Publisher* additionally has many registered *Gateways* who act on the behalf of the

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Publisher. The *Identities* of the *Gateways* are defined by the *Gateway-Identity-Domain* of the *Publisher*. This domain specifies which *Characteristics* are required for a *Gateway* to register itself with a *Publisher*. Furthermore, the *Publisher* maintains another *Thing-Identity-Domain* which contains the *Identities* of the *Things* the *Publisher* manages. The *Identities* of this domain are shared with the authenticated *Gateways* of the *Publisher*. However, it must be noted that the *Identities* of the *Thing-Identity-Domain* of the *Publisher* are not the same *Identities* that are shared with the *Publisher* via the *Thing-Identity-Domain* of the TMS. The *Publisher´s "Thing-Identities"* are mapped to the *"Thing-Identities"* of the TMS, which is illustrated in Figure 27. This mapping between the *Identities* that are shared with *Gateways* and the *Identities* that have been shared with the *Publisher* allows the *Publisher* to maintain its role as a proxy. Only the *Publisher* knows which *Identity* that is shared with a *Gateway* refers to a certain *Owner*, thus the *Publisher* is able maintain the privacy of the *Owners* of the *Things* he manages.

However, this system additionally needs to consider the network-relation between a *Gateway* and a *Thing*. The fact that a *Gateway* acts on behalf of a *Publisher* needs to be incorporated into the "network-level" relation between a *Gateway* and a *Thing*. The *Gateway* needs to authenticate itself as a representative of the associated *Publisher* of a *Thing* with which it intends to establish a communication channel. For this purpose, it could be possible for the *Publisher* to issue an authentication-token to all its *Gateways*. A *Gateway* can then provide this token to the *Thing* it intends to communicate with. The *Thing* must then validate the token and grant or deny access accordingly. However, as described in section 4.3.2, this would require a *Thing* to provide an authentication system. This system must either be able to validate the token on its own or communicate with its *Publisher* to let it validate the token. The first option is similar to the *credential-focussed* identity management approach and the latter is similar to the *relationship-focussed* approach to identity management (see section 2.4). However, both options are equally problematic. The first option requires the token "valid on its own" (e.g. like a passport, as described in section 2.4), which is unlikely to be feasible because it will become increasingly difficult to validate a token when the number of *Things*, *Publishers* or *Gateways* grows. The second option would require the *Thing* to be able to directly communicate with its associated *Publisher*, which is similar to the credit card example described in section 2.4, which is also not applicable. This is because if the *Thing* would be able to directly communicate with its *Publisher* there is no need for a *Gateway* in the first place.

In conclusion, *Gateways* need to have a relation to the *Publisher* to be able to access *Things* on behalf of a *Publisher* who then can maintain the privacy and anonymity of the *Owner* corresponding to the *Things*. However, the authentication issue between *Gateways* and *Things* remains to be solved.

4.4 Holistic IoT Architecture Framework based on S2aaS

Based on the *IoT Architecture Perspectives* identified in section 4.2.1 and the *IoT Architecture Components* described in section 4.2.2 the lack of a component in the overall architecture has been identified. The TMS, which has been developed in section 4.3.3, is a novel component which will be integrated into the *Holistic IoT Architecture Framework* in this section.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Table 4 provides an overview of all components the *Holistic IoT Architecture Framework* consists of. The fist column denotes the component, while the second column provides a description of the respective component. The third column of Table 4 contains the relations a component has with other components of the architecture. Furthermore, each relation is described from both, the *Network IoT Architecture Perspective* and the *Organisational IoT Architecture Perspective*.

Table 4: Overview of the components of the Holistic IoT Architecture Framework (own listing)

Component	Description	Relations	
Consumer	• Is exclusively interested in obtaining data and information as well as offloading tasks it cannot perform with its own resources to other components. • Will provide compensation or incentives for using services like consuming data and information or offloading tasks to other components.	Service Provider	• **Network:** Can directly access a *Service Provider`s* services via the internet (e.g. by accessing the web application provided by the *Service Provider*). • **Organisational**: Provides compensation when accessing services (e.g. issuing *Sensing Tasks)* provided by the *Service Provider*.
		Publisher	• **Network:** Can issue *Sensing Tasks* by directly accessing the *Publishers* API endpoint via the internet. • **Organisational**: Provides compensation when accessing services (e.g. API endpoint of the *Publisher*).
Thing	• Provides unique services in form of either *sensing* or *actuating* capabilities. • Mode of use (e.g. when it can be accessed, what compensations are demanded, etc.) is predetermined by the corresponding *Owner*. • Is generally restricted regarding energy consumption, computational power and data storage. • Needs to rely on opportunistic communication channels if deployed in *foreign* environment. • The *Characteristics* of a *Thing* (e.g. metadata like	Owner	• **Network:** If deployed in an environment that is fully controlled by the *Owner*, the *Owner* can establish a direct connection with the *Thing*, optionally via the TMS. If deployed in a foreign environment, a direct connection between *Owner* and a *Thing* cannot be established. • **Organisational**: All aspects and properties of the *Thing* are managed by the *Owner*.
		Publisher	• **Network:** If deployed in an environment that is fully controlled by the *Owner* of the *Thing*, the *Publisher* can establish a direct connection with the *Thing*. If deployed in a foreign environment, the *Publisher* needs to rely on *Gateways* to establish an opportunistic connection. • **Organisational**: The *Thing* is “registered” to one or more *Publishers*, which act as a proxy between the *Thing* and other third parties and represent the interests of the *Thing´s Owner*. The interests of the *Owner* are held down in a “contract”, which is formed during the registration of the *Thing* with the *Publisher*.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

	location data, or sensing data in general) denote its *Owner*.	TMS	• **Network**: If deployed in an environment that is fully controlled by the *Owner* of the *Thing*, the TMS can establish a direct connection with the *Thing*. If deployed in a foreign environment, the TMS cannot establish a direct connection at any time, the TMS must rather rely on an opportunistic connection that is manually provided by the *Owner* (e.g. to register a new *Thing* with the TMS). • **Organisational**: A *Thing* is initially registered with the TMS (e.g. network addresses and metadata are initially stored). From then on, the TMS handles all *Sensing Tasks* issued by the corresponding *Publishers*.
		Gateway	• **Network:** If deployed in a foreign environment, the *Thing* can establish a connection with a *Publisher* via a *Gateway*. The *Gateway* can directly establish a connection with the *Thing* and the corresponding *Publisher(s)*. • **Organisational**: The *Thing* does not have an organisational tie with a *Gateway*.
Owner	• The *Owner* is the main stakeholder of a *Thing* and imposes his requirements and interests onto the *Things* he owns. • The *Things* owned by the *Owner* refer to himself and are considered as the *Owner´s Identities* because each *Characteristic* of a *Thing* can potentially denote a property of the *Owner* (e.g. the location of a *Thing* could denote the address of the *Owner*).	Thing	• **Network**: The *Owner* can establish a connection via the TMS, depending on the environment the *Thing* is deployed in. • **Organisational**: The *Owner* manages the *Characteristics* of a *Thing* via the TMS.
		TMS	• **Network**: The *Owner* can directly access the web application provided by the TMS. The connection is either established by using the "home-network" of the *Owner* (when the TMS is hosted on premise) or via the internet. • **Organisational**: The *Owner* manages his *Things* via the TMS. However, the *Owner* has no direct organisational relation with the TMS.
		Publisher	• **Network***:* The *Owner* does not establish a network connection with the *Publisher*, except for an indirect connection during the registration of a *Thing* with a *Publisher*, however this connection is handled by the TMS. • **Organisational**: The *Publisher* acts on behalf of the *Owner* when advertising the Services offered by the *Things* of the *Owner*. The *Owner* may provide some compensation for using the *Publisher* as representative or proxy.
TMS	• The TMS manages the *Things* and their corresponding *Identities* on behalf of its *Owner*. • *Owners* consume the management services provided by the TMS.	Thing	• **Network:** Depending on the environment the *Thing* is deployed in (controlled or foreign), the TMS can establish a direct connection or needs to rely on a connection that is manually created by the *Owner* of the *Thing*. • **Organisational**: The TMS manages the *Characteristics* of each of its *Things* on behalf of the *Owner*.

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

	• The TMS can automatically react to *Sensing Tasks* based on the policies the *Owner* has defined. • The TMS provides *Identities* for *Things* and assists *Owners* in assigning these *Identities* to *Publishers.*	Publisher	• **Network**: The TMS can establish a permanent and direct connection with a *Publisher* via the internet. • **Organisational**: The TMS receives the *Sensing Tasks* forwarded by the *Publisher*. It initially forms contracts regarding an individual *Thing* or a set of *Things* with a *Publisher*.
		Owner	• **Network**: The TMS can establish a permanent and direct connection with the *Owner* via the internet or the local home network. • **Organisational**: The *Owner* consumes the management services of the TMS.
Publisher	• Maintains a database of *Things* along with descriptive metadata. • Exposes and advertises the services of the *Things* it manages. • Accepts incoming *Sensing Tasks* and forwards these tasks, to *Owners* of *Things* that could fulfil the respective tasks and whose *Thing´s* restrictions and access rules do not contradict with the respective *Sensing Task* (e.g. an *Owner* might forbid tasks requiring non-anonymised location data). • Conceals the real *Identity* of the *Owners* of the *Things* it manages. • Wants to be compensated for the proxy-services it provides.	TMS	• **Network:** The *Publisher* can establish a permanent and direct connection with the TMS via the internet. • **Organisational**: The *Publisher* forwards the *Sensing Tasks* it received to each TMS corresponding to a suitable *Thing* that could fulfil the *Sensing Task*.
		Owner	• **Network:** The *Publisher* does not establish a network connection with the *Owner*, except for an indirect connection during the registration of a *Thing* with the *Publisher*, however this connection is handled by the TMS. • **Organisational**: The *Publisher* heeds the policies which have been defined by the Owner during the registration of a *Thing* with the *Publisher*. Hence, the *Publisher* represents the *Owners* interests and demands compensation for its services.
		Thing	• **Network**: Depending on the environment the *Thing* is deployed in (controlled or foreign), the *Publisher* can either establish a direct connection via the internet or an opportunistic connection via a *Gateway*. • **Organisational**: The *Publisher* advertises the services of a *Thing;* thus, *Things* can be interpreted as the products a *Publisher* intends to sell on behalf of the *Owner*.
		Gateway	• **Network:** The *Publisher* can establish a direct connection with the *Gateway* via the internet. • **Organisational**: The *Publisher* uses *Gateways* to establish opportunistic network connections with *Things* which are deployed in foreign environments. These *Gateways* act on behalf of the *Publisher* and are compensated for their services.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

		Service Provider	• **Network**: The *Publisher* can establish a direct connection with *Service Providers* via the internet. • **Organisational**: The *Publisher* offers the services of the *Things* it acts on behalf of and demands compensation for these services on behalf of the corresponding *Owners* of the *Things*.
		Consumer	• **Network:** The *Publisher* can establish a direct connection with *Consumers* via the internet. • **Organisational**: The *Publisher* offers the services of the *Things* it acts on behalf of and demands compensation for these services on behalf of the corresponding *Owners* of the *Things*.
Service Provider	• The *Service Provider* offers value added services and demands compensation for these services. • Services range from easy access to a large number of *Things* to transforming and reasoning over sensed data and displaying the data appropriately.	Publisher	• **Network:** The *Service Provider* can establish a direct connection with the *Publishers* via the internet. • **Organisational**: The *Service Provider* issues *Sensing Tasks* to the *Publishers* it knows and provides compensation for the services of the *Publishers*.
		Consumer	• **Network:** The *Service Provider* can establish a direct connection with *Consumers* via the internet. • **Organisational**: Provides services and demands compensation.

The components and relations of the *Holistic IoT Architecture* are further illustrated in Figure 28. In essence, the architecture framework developed during this thesis is closely oriented towards the original S2aaS architecture presented by Perera et al. (2014) and Sheng et al. (2013). The basic concepts of all major components, the *Publisher*, the *Owner and Thing, the Service Provider* and the *Consumer* can be easily mapped to the architectures presented by Perera et al. (2014) and Sheng et al. (2013). However, it has been discussed in section 4.2.1 that both Perera et al. (2014) as well as Sheng et al. (2013) can be assigned to the *Organisational IoT Architecture Perspective*, in which specific possibly bothersome aspects of IoT could have been eluded because they were not the focus of the respective architecture proposal. By distinguishing between different *IoT Architecture Perspectives* and using them to analyse a variety of IoT architecture proposals, the need for a novel component and issues regarding the network connectivity between *Things* and *Publishers* have been identified. The *Thing Management System*, which is the new component, supports *Owners* in administering their *Things* and is located between *Things*, *Owners* and *Publishers* in the *Holistic IoT Architecture Framework (see* Figure 28*)*. The connectivity issues have been discussed in section 4.3.4 and are superficially incorporated into the architecture illustrated in Figure 28.

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

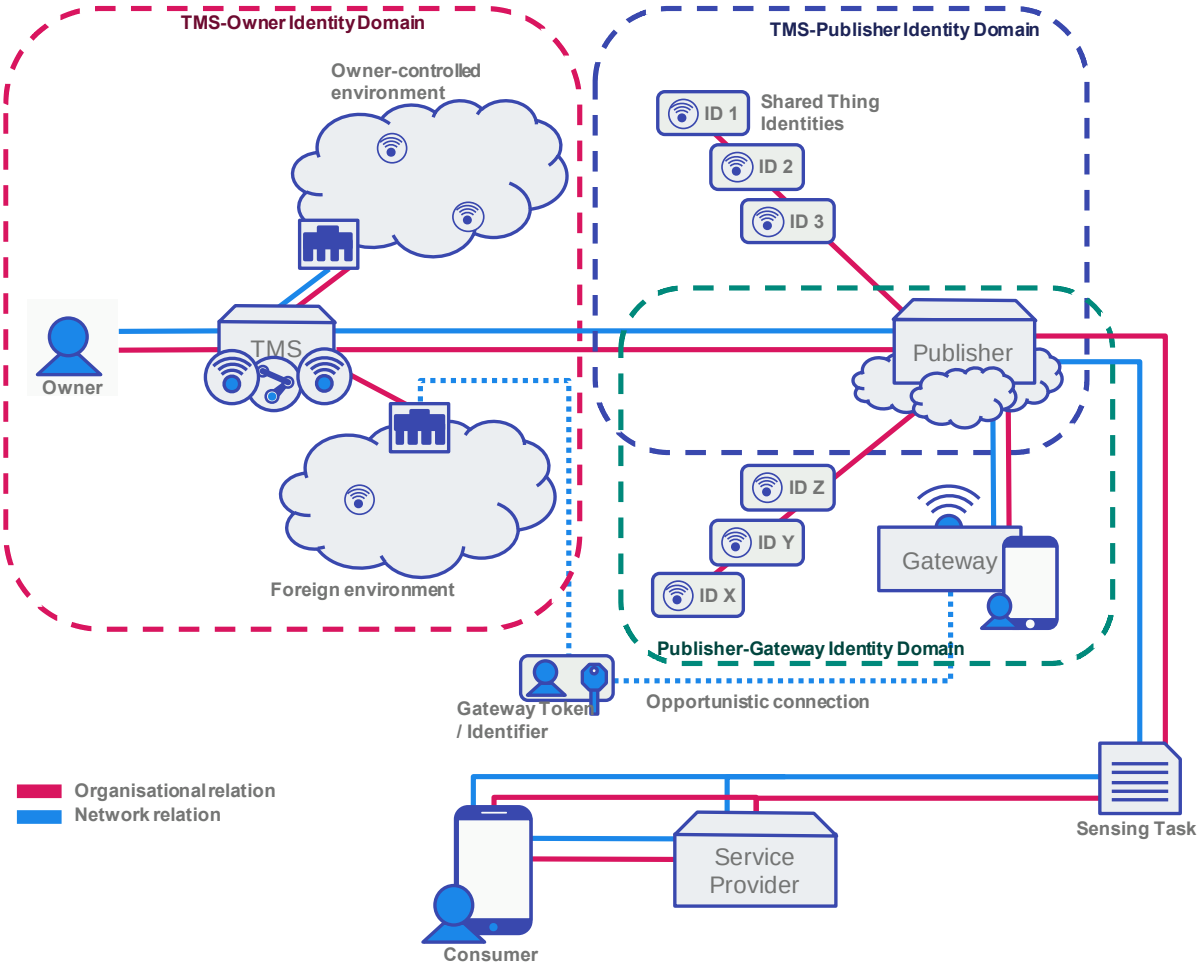


Figure 28: *Holistic IoT Architecture Framework* (own illustration)

This section concludes the development of the *Holistic IoT Architecture Framework* and specifically answers and achieves RQ3.1 and RO3 respectively. The following chapter will cover the development of a prototype using the *Holistic IoT Architecture Framework.*

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

5 Implementation of the Prototype

Based on the developed *Holistic IoT Architecture Framework* (see Figure 28) and the completed corresponding *development* step of the GDC this chapter addresses the *evaluation* step of the GDC. As stated in section 2.2, this chapter utilises the *demonstration pattern* (Vaishnavi & Kuechler 2007). According to this pattern, the *Holistic IoT Architecture Framework* is validated through implementation work, which is demonstrated in this chapter. In the course of this, the issues raised in section 1.1 are covered.

The implementation of each component of the *Holistic IoT Architecture Framework* aims to demonstrate that the architecture provides sufficient specifications of roles and responsibilities, requirements and descriptions of relations so that each component can be implemented separately. Additionally, each component must be able to communicate with its related components according to the architecture framework. However, due to the fact that the architecture framework merely defines the semantics[15] of inter-component communication, the actual syntax used for the communication in this prototype can be simplified. Thus, the prototype only aims to demonstrate a specific situation, which uses only a single type of *Things*, adopting a mock-up communication syntax. Nevertheless, the prototype is designed to be extensible to the effect that it supports multiple different communication channels and data formats.

The following paragraphs will briefly elaborate the most important technologies, frameworks and implementation approaches used throughout the implementation of the prototype.

Ruby on Rails

The components of the *Holistic IoT Architecture* have to offer and consume various services over the internet and are generally considered as web applications (see section 4.2.2 and 4.3.3). To implement these applications, *Ruby on Rails*[16] (RoR) was selected as the server-side web application framework. The RoR framework was selected because it is especially suitable for rapid prototyping due to its two predominant and guiding principles, which are deeply embedded in the architecture and development process of RoR applications. The first principle, “don’t repeat yourself (DRY)”, is a common principle of software design and states that every piece of information must have an unambiguous and singular purpose or representation within a software system and should be reusable. The second, more important principle of RoR is “convention over configuration”. The RoR framework predefines a set of conventions which are deemed to be “best practice” (e.g. naming conventions, file- and data-structure).

[15] The *semantics* of inter-component communication refer to the information exchanged between the components (e.g. a contract between *Thing Management System* and *Publisher*), without specifying any requirements regarding the *syntax* of the communication (e.g. data format, what data a contract consists of, etc.).

[16] https://rubyonrails.org/

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

When adhering to these conventions, applications can be developed rather quickly while remaining scalable and extensible.

PostgreSQL

Considering the overall heterogeneity of IoT regarding technologies, data formats as well as communication protocols and technologies and the sheer scale of IoT in general yield several requirements for the selection of a storage engine. Firstly, to be able to represent various types of *Things*, each with different data structures, the storage engine should provide means of storing data without requiring a schema. For example, the metadata provided by each *Thing* can be in a different format and may very well contain different key-value pairs. In order to remain flexible and scalable, this data should be stored directly in the storage engine without transforming or parsing it beforehand. Secondly, the storage engine itself should be scalable. In essence, this means that the storage engine should be able to store and handle very large amounts of data, should be able to retrieve data relatively fast and should be able to be deployed as a distributed system. However, for the implementation of the prototype, the second requirement will not be considered for the selection of a storage engine[17]. Consequently, PostgreSQL[18] was selected as the storage engine for the prototype for the following reasons. Firstly, it provides means of directly storing, indexing and retrieving schemaless data despite being a relational database. Secondly, it is supported by RoR´s built *object-relational-mapper* which further simplifies the development process.

The subsequent sections of this chapter will briefly present the *use cases*, *data model*, *interfaces* of each component of the implemented *Holistic IoT Architecture Framework*. Additionally, each section will discuss the insights, which have been gained during the development process of each component. These insights are then further discussed and refined in the last section of this chapter.

[17] It must be noted that the selected storage engine, PostgreSQL, might face scalability issues when the amount of data stored significantly exceeds reasonable amounts deemed appropriate for a prototype (e.g. PostgreSQL may experience performance issues when exceeding 1-2TB of data, whereas the average representation of a *Thing* of the prototype implementation ranges from 50B to 200B, which means that performance issues of PostgreSQL may arise when exceeding 5×10^9 *Things*).

[18] https://www.postgresql.org/

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

5.1 Thing Management System

The implementation of the *Thing Management System* is guided by the functional requirements and modules defined in section 4.3.3. Based on these requirements, the use cases illustrated in Figure 29 are supported by the implemented prototype component.

Users, which are the *Owners* in terms of the *Holistic IoT Architecture Framework*, can register themselves with the TMS. During this process, users define their *Credentials* which consist of an e-mail and a password. These credentials are encrypted and stored in the database. As soon as a user is authenticated, he can access the *Management* module of the TMS. This module allows the *user* to create or import *Things*. The prototype representation of a *Thing* consists of a name, which is defined by the *user* and *metadata* providing descriptive information of a *Thing* (see Figure 30). This data may contain the address (e.g. IP) of the *Thing* and details for accessing it (e.g. what protocols to use, which data format, etc.). This data is stored and handled as a schemaless JSON[19] by the TMS. The user is able to create, update and delete each of his own *Things*. As stated in section 4.3.3, the *Provisioning* module of the TMS distinguishes between on premise deployments and deployments as *Service Providers*. Thus, multiple users can register with the TMS and are able to manage *Things* simultaneously. This allows both the deployment as a *Service Provider* and as an on premise application with only a single *user*. However, the data of each *user* and the corresponding *Things* are stored in the same database and on the same server, which might not fulfil advanced data privacy and security requirements.

For each *Thing,* a *user* can create several *Identities*. Each *Identity* inherits the *metadata* of its corresponding *Thing*. Furthermore, a *user* then can define the visibility of each inherited *Characteristic* of an *Identity*. When defining a *Characteristic* as “private”, it will not be shared with other components. A *Characteristic* defined as “public” will be shared with other components (e.g. *Publishers*). *Identities* are published by assigning them to *Contracts* and submitting these *Contracts* to a *Publisher*. A *Contract* consists of many *Identities* and *metadata*. The *metadata* is again stored and handled as a schemaless JSON and may be used to model access rules, restrictions or the compensations demanded for accessing the services provided by a *Thing*. Additionally, a *Contract* is assigned to a single *Publisher*. Upon submitting a *Contract* to a known *Publisher*, the *Contract* can either be “accepted” or “rejected” by the *Publisher*. When the *Publisher* accepts the *Contract*, which means that he is willing to act as a proxy between the *Identities* of the *Contract* and other components (e.g. *Service Providers)*, the TMS allows to receive *Sensing Requests* for *Identities* contained in the *Contract*. As discussed in section 4.3.3, the TMS must always retrieve the *users’* consent before performing an action. Thus, upon receiving a *Sensing Request*, the *user* can decide to “accept” or “reject” the *Sensing Request.* When the request is accepted, the corresponding *Publisher* will access the *Things´* specified in the *Sensing Request*.

[19] http://www.json.org/json-de.html

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

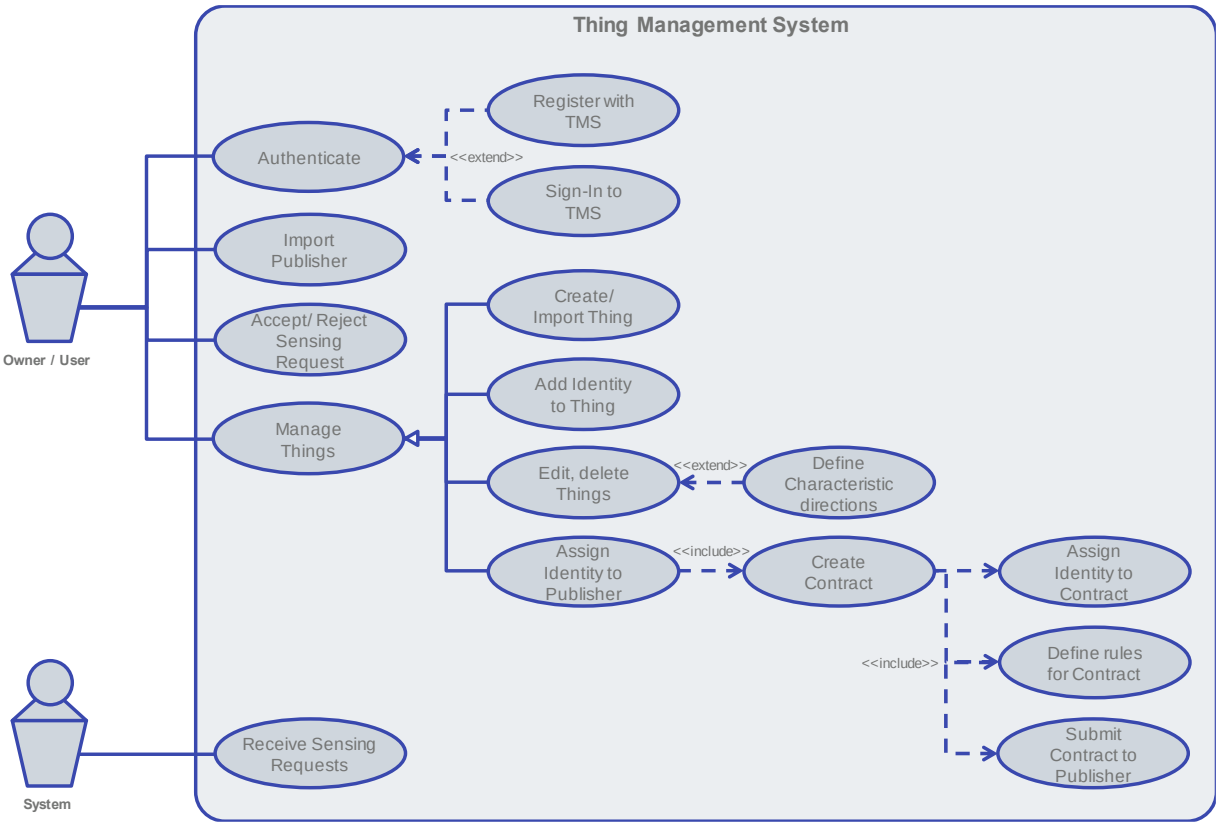


Figure 29: Use case diagram of the *Thing Management System* (own illustration)

To communicate with other components, the prototype implementation of the TMS uses the following RESTful APIs.

- **Contract-API**
  The *Contract-API* sends the *Contract* data to a registered *Publisher* via a HTTP POST request. The payload of the request consists of the JSON representation of the *Contract* along with the related *Identities*.
- **Sensing Request-API**
  The *Sensing Request-API* receives *Sensing Requests* via HTTP POST requests. Upon receiving a valid JSON payload, which consists of the *Contract* the *Sensing Request* relates to along with an additional policy, which specifies the *Sensing's Request* purpose, compensation and incentive mechanism.
  Additionally, the *Sensing Request-API* sends an *acceptance* or *rejection* to the corresponding *Publisher* via a HTTP POST request.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

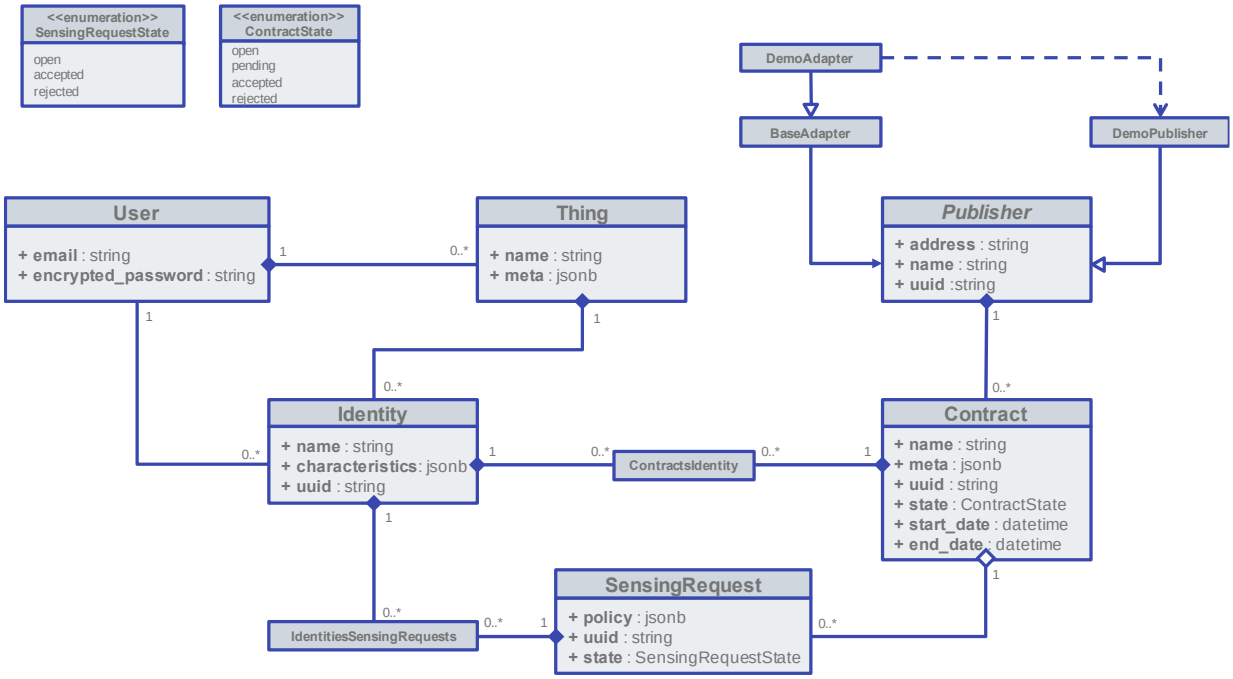


Figure 30 Class diagram of the *Thing Management System* (own illustration)

During the implementation of the *Thing Management System* the challenge of uniformly representing *Things* became more apparent. Considering that different *Things* may very well provide and require different interfaces, different metadata and different addressing mechanisms, a considerable amount of effort must be put into addressing this heterogeneity issue. The first step to approach this issue is to utilise schemaless databases for storing the *Things* data. The second, more challenging, step is to develop *adapters*[20] that provide the necessary behaviour based on the metadata of a *Thing*. However, the effort required to provide an *adapter* for each kind of *Thing* scales with the amount of different types of *Things*, which might be too high. However, this problem could either be solved by relying on the vendors of *Things* to provide the necessary adapters or by following the *Semantic Oriented Vision of IoT*. As discussed in section 3.1, this vision of IoT suggest to utilise semantic technologies which could be used to automatically generate the necessary adapters based on the *Things* metadata.

[20] This approach is similar to the *adapter pattern* presented by Gamma et al. (1995).

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

5.2 Publisher

The implementation of the *Publisher* is guided by the descriptions and specifications provided in section 4.2.2. Based on these requirements, the use cases illustrated in Figure 31 are supported by the implemented prototype component.

As discussed in section 4.2.2, the main purpose of the *Publisher* is to mediate the communication between *Owners*, *Things* and *Service Providers*. To illustrate this function, the implemented *Publisher* component only supports a specific set of use cases that are only accessible via a RESTful API. The *Publisher* has no additional stakeholders beside the *Publisher´s* "*system*" itself. The *system* can receive *contracts*, which essentially consist of metadata, rules and *Identities.* Upon receiving a *contract*, the *system* may accept or reject it. Both, the *contract's* metadata and rules as well as the *Identity´s Characteristics* are stored and handled as a schemaless JSON (see Figure 32).

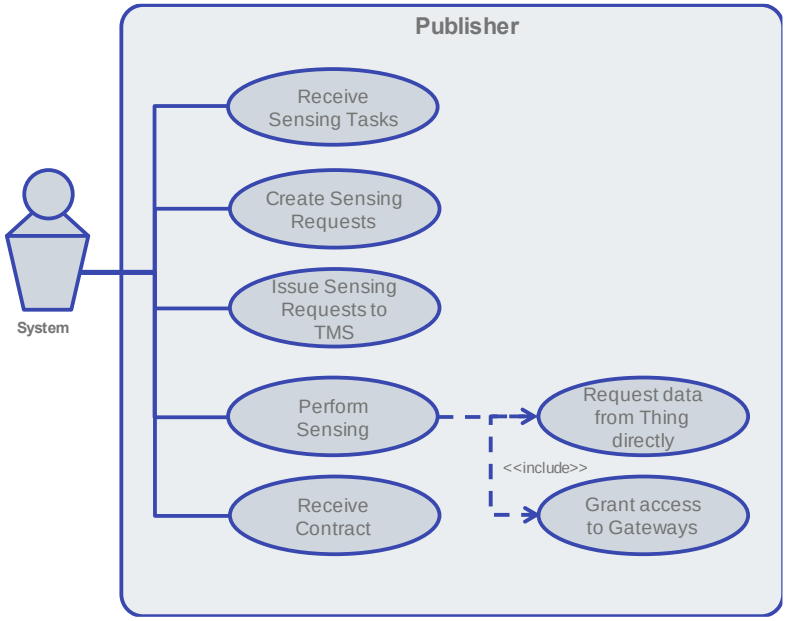


Figure 31: Use case diagram of the *Publisher* (own illustration)

The *system* can receive *Sensing Tasks* and create *Sensing Requests* based on the criteria provided by the *Sensing Tasks*. The criteria for creating and issuing a *Sensing Request* are extracted from the respective *Sensing Task´s* metadata, which is also stored and handled as a schemaless JSON. The prototype implementation of this feature uses the *metadata* of the *Sensing Task* and matches it with the *Identities* the *Publisher* manages. The matching mechanism of the prototype uses a query-string which is divided and compared with each value of an *Identity´s metadata*. For example, if the query-string of the *Sensing Task* contains a certain keyword and any value of the *metadata* of an *Identity* contains the same keyword, the *Sensing Task* matches the *Identity*. However, this matching mechanism is rather simplistic and only for demonstration purposes. Subsequently, the *system* can issue the *Sensing Requests* which have been created based on the received *Sensing Tasks* to the corresponding TMS. The information

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

regarding the address of a TMS, the data format and communication protocols to be used is stored in the corresponding *Contract* of the *Identity* which matches the *Sensing Task´*s query (see Figure 32).

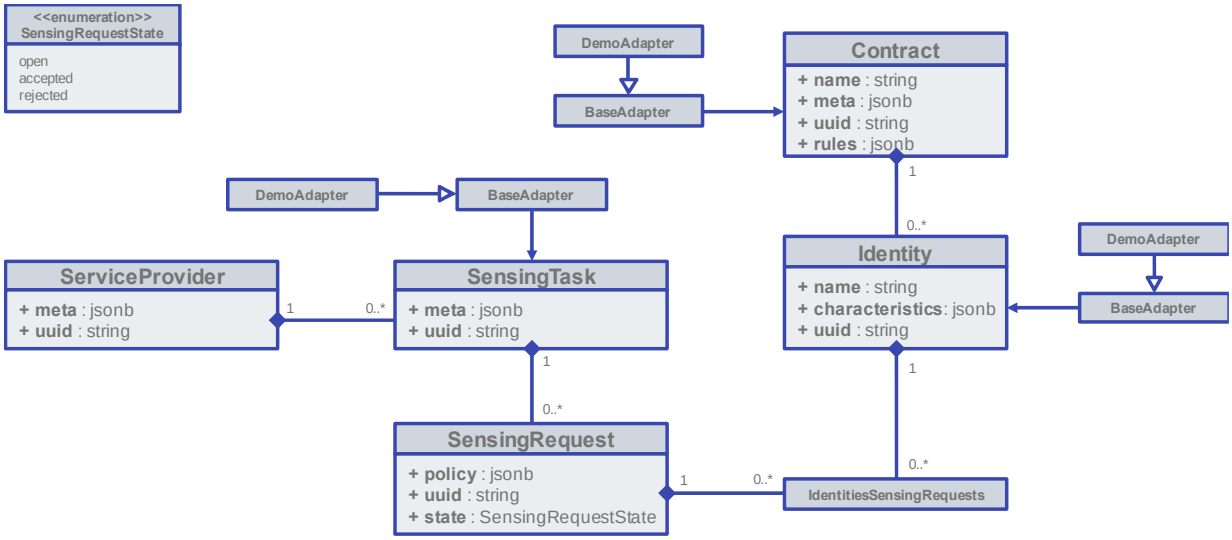


Figure 32: Class diagram of the *Publisher* (own illustration)

As soon as a *Sensing Request* has been accepted by the TMS (see section 5.1), the *system* can perform sensing by either directly accessing the services of the *Things* corresponding to the *Identities* a *Sensing Request* consists of or by granting *Gateways* access to the *Things*. This behaviour maps to the different kinds of environments *Things* might be deployed in (see section 4.3.1 and Figure 22 in particular).

In order to be able to mediate the communication between *Things*, *Owners* and *Service Providers*, the *Publisher* provides several RESTful APIs.

- **Sensing Task-API**
  The *Sensing Task-API* receives *Sensing Tasks* via HTTP POST requests. Upon receiving a valid JSON payload, which consists of the metadata describing the *Sensing Task*, a *Sensing Task* is being created and associated *Sensing Requests* are being generated.
  Additionally, the *Sensing Task-API* sends the data collected by *Gateways* or by the *Publisher* itself to the *Service Provider* corresponding to the *Sensing Task* via HTTP POST requests.
- **Contract-API**
  The *Contract-API* creates *Contracts* received via HTTP POST requests. The payload of a valid request contains the *Contract's* metadata and rules as well as a set of *Identities* which relate to the *Contract.*
- **Sensing Request-API**
  This API sends *Sensing Requests* via HTTP POST requests to a TMS. The payload of a request consists of the associated *Sensing Task´s* metadata, the *Contract* the *Sensing Request* relates to as well as a set of *Identities*.
  Additionally, the *Sensing Request-API* receives the acceptance or rejection of a *Sensing Request* via HTTP POST requests.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

- **Identity-API**
  This Identity-API collects the sensing data from *Things*. The type of request depends on the specified metadata of the *Thing* or the corresponding *Identity* managed by the *Publisher*. The prototype API uses HTTP GET requests to obtain the data from *Things*.

During the implementation of the *Publisher* component, challenges regarding the management and dynamic creation of suitable adapters to cope with the heterogeneity of *Things*, which also has been described in the previous section, became apparent. Additionally, the task of matching a *Sensing Task* to fitting *Identities* based on arbitrary rules, policies and specifications is a challenging task. As mentioned, the prototype implementation utilises a simplistic, demonstrative string matching mechanism. However, this mechanism should be further developed by using more sophisticated search mechanisms such as Apache Lucene[21] and Elasticsearch[22] which provide sophisticated search and matching functionalities that are able to cope with the heterogeneity of *Things*, their data representations and *Sensing Tasks* with their various requirements.

5.3 Service Provider

The implementation of the *Service Provider* is guided by the descriptions and specifications provided in section 4.2.2. Based on these requirements, the use cases illustrated in Figure 31 are supported by the implemented prototype component.

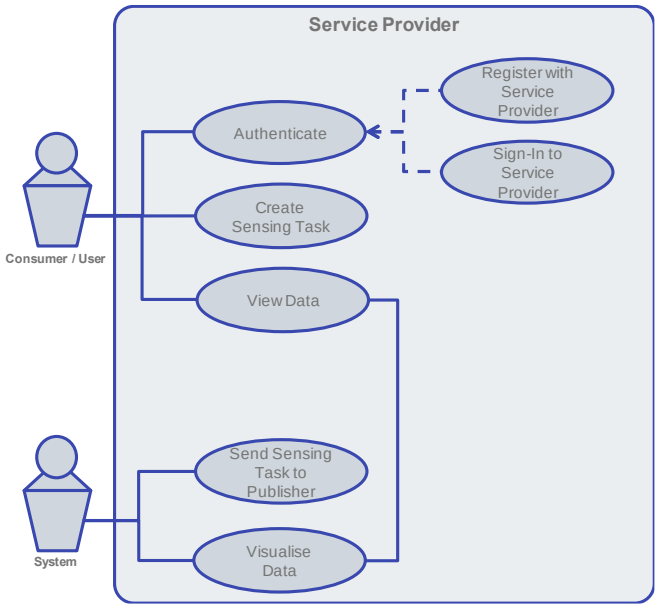


Figure 33: Use case diagram of the *Service Provider* (own illustration)

[21] https://lucene.apache.org/core/

[22] https://www.elastic.co/de/products/elasticsearch

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

The *Service Provider* is mainly used by *users* who act as *Consumers* in terms of the *Holistic IoT Architecture Framework*. *Users* can register themselves with the *Service Provider*. During this process, users define their *Credentials,* which consist of an e-mail and a password. These credentials are encrypted and stored in the database. As soon as a *user* is authenticated, he can create *Sensing Tasks* and view the data of the tasks he issued. A *Sensing Task* in this prototype implementation of a *Service Provider* consists of *metadata* and *result-data*. The *metadata* contains a query-string along with additional fields, e.g. for describing usage policies or compensations provided. The *result-data* contains the data that is sensed by the *Things* corresponding to the *Sensing Task*. The *metadata* as well as the *result-data* is stored and handled as a schemaless JSON. When a *Sensing Task* has been created, the *system* can send the task to known *Publishers* via a RESTful API (see Figure 34). Likewise, the *system* can receive the data associated to *Sensing Tasks* and visualise the received data for the user.

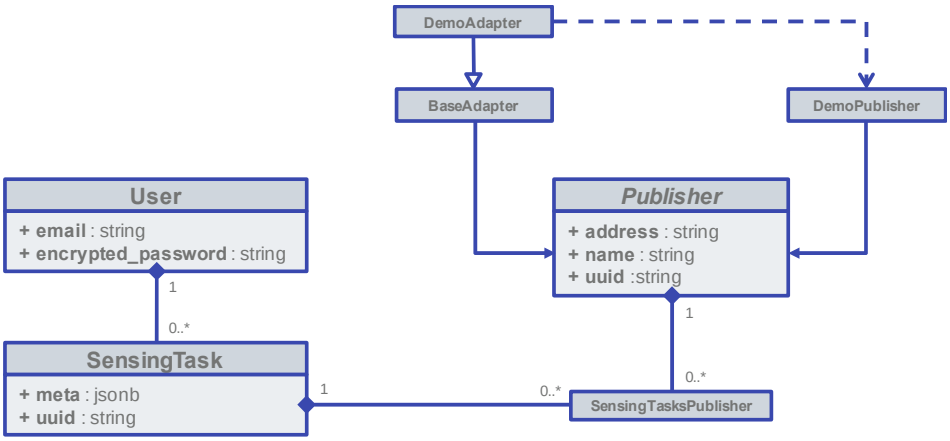


Figure 34: Class diagram of the *Service Provider* (own illustration)

Besides offering a web application for the *user* to interact with, the *Service Provider* uses a RESTful API to communicate with *Publishers*.

- **Sensing Task-API**
  This API sends *Sensing Tasks* via HTTP POST requests to a *Publisher*. The payload of a request consists of the *metadata* of the *Sensing Task.*
  Additionally, this API receives sensing data related to a *Sensing Task* and stores the data in the database.

The prototype implementation of the *Service Provider* component yielded insights similar to the ones gained during the implementation of the *Publisher* component. Again, heterogeneity remains an issue and providing adapters for each different kind of representations of *policies* or *incentive mechanisms* requires considerable effort. Additionally, parsing and transforming “high-level” user queries (e.g. air pollution data in a specific city) into formats that can be transferred to *Publishers* for further analysis is a difficult task. As suggested during the development of the *Publisher*, technologies like Apache Lucene or Elasticsearch, which provide sophisticated search mechanism, may prove to be helpful.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

5.4 Evaluation Results

The implementation of the *Holistic IoT Architecture Framework* as a prototype demonstrated that the conceptual design of the framework including the theoretical specifications and descriptions of each of its components is technically realisable. The implementation work evaluates and shows the general feasibility of the overall architecture. While addressing the technical details of each components deeper insights into the operation and performance could be gained. However, looking at the technical details and inner workings of each component in detail and evaluating the architecture holistically at all levels of analysis may go beyond the scope of the thesis described in section 2.4. In this section, it is stated that the thesis aims to develop a *high-level* architecture framework, which does not emphasise the inner workings of components. Nonetheless, the evaluation of the implementation and especially technical details yielded evaluation results that may affect the high-level aspects of the *Holistic IoT Architecture Framework*. Thus, considering technical details and inner workings of components during the evaluation is justified. The following paragraphs elaborate the evaluation results of the *Holistic IoT Architecture Frameworks´* prototype implementation.

Component Addressing System

Regarding Figures Figure 30, Figure 32 and Figure 34, it becomes apparent that each contains a representation of either the *Publisher* component or the *Service Provider* component. This is due to the fact that each component requires information regarding other components it is intended to talk to. For example, the TMS prototype requires users to import *Publishers*. The *Publisher* stored in the database of the TMS contains information that describes the APIs the *Publisher* provides (see section 5.2 for the APIs provided by *Publishers*). As described in section 5.1, the TMS sends *Contracts* to *Publishers* via HTTP POST requests, which requires an address to which the request is sent to. Depending on the API, additional information might be required to transmit the information (e.g. when authentication is required). Users of the TMS are required to provide this information for the TMS to be able to send *Contracts* to *Publishers*. The same principles apply for both the *Publisher* and the *Service Provider*. However, manually importing the required information for each *Publisher* or *Service Provider* might not be applicable. The reason for this is twofold. Firstly, when the number of components grows, the management of the configuration information might become increasingly time consuming and complex. Secondly, when individual instances of components cease to exist (e.g. a *Service Provider* discontinues his services) or new instances are created (e.g. a new *Publisher* services enters the eco system), the changes do not reach each component. For example, a TMS might try to submit *contracts* to a *Publisher* which doesn't exist anymore. Consequently, this would require the users of each TMS to manually check if the Publisher information is still valid. Again, the same applies for the *Publisher* and *Service Provider* component.

To address this issue a system similar to the *Discovery Server* proposed by Chang et al. (2015) might be incorporated into the overall architecture. The *Component Addressing System* would provide information regarding available *Publishers* and *Service Providers*. This information may range from configuration data (e.g. API descriptions) to reputational classifications (e.g. monitoring and publishing

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

the reputation of a *Publisher*) or certification information (e.g. certifying the identity of a *Publisher* imported to the TMS).

Heterogeneity

The multitude of different types of *Things*, different data formats and representations remains an issue. However, with the *Holistic IoT Architecture Framework* defining the responsibilities, roles and relationships for each component, the starting point for addressing this issue is clearly marked out. As stated in the previous sections, this issue can be solved by providing a sufficient variety of adapters responsible for translating different data formats or communicating via different interfaces. This could be achieved by either implementing an adapter for each kind of *Thing* or by utilising the principles and technologies suggested by the *Semantic Oriented Vision of IoT* discussed in section 3.1 to automatically generate adapters or interfaces.

This chapter finalises the application of the GDC. The discussion of suitable technologies for implementing the prototype in the beginning of this chapter addressed RO4 and the corresponding research questions RQ4.1 and RQ4.2.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

6 Summary and Conclusion

This chapter summarises the key findings of this thesis in section 6.1 by answering each research question that was posed in section 1.2. Subsequently the research contribution (see section 6.2) and the limitations of this thesis (see section 6.3) are briefly discussed. This thesis concludes with hints for future work in section 6.4.

6.1 Research Questions

In this section, each of the research questions that guided the research conducted in this thesis (see section 1.2) is answered individually. For each research question, a summarising answer is provided along with the respective reference to the corresponding section(s) where the research question is elaborated on and answered in detail.

RO1 Identify and evaluate components of S2aaS.

RQ1.1 Which components of S2aaS are addressed in the pertinent IoT architecture literature?

Based on the *IoT Architecture Perspectives* developed in section 4.2.1, the following components have been identified in the literature and been described in detail in section 4.2.2. Across the pertinent literature the components *Consumer*, *Thing*, *Owner* and *Service Provider* are mentioned either implicitly or explicitly. The *Consumer* component is essentially interested in sensing data and is willing to provide compensation (see page 53). The *Thing* component, giving IoT its name, provides unique sensing and actuating services and is considered the bridge between the digital and physical world (see page 53). *Things* are owned by *Owners*, who essentially govern their *Things* by defining access rules or requiring compensation for their *Things*` services (see page 54). *Service Providers* provide value added services, which are employed by *Consumers*. In order to provide these value-added services, *Service Providers* rely on the sensing data of *Things* (see page *55*). Depending on the respective *IoT Architecture Perspective*, the components *Publisher* and *Gateway* are considered in the literature or not. *Gateways* aim to establish communication between the individual components, especially between *Things* and other components (see page 55). *Publishers* intend to represent the interests of *Owners* (e.g. privacy), thus they act as proxy between *Things*, *Owners* and *Service Providers* (see page 56). Additionally, with *Owners* potentially owning many different *Things*, the need for an additional component, the *Thing Management* System, has been identified in section 4.3.1. Essentially, the TMS allows *Owners* to manage a multitude of different *Things*, including the selection of data to be shared or published. The details of the TMS are discussed in section 4.3.3.

RQ1.2 Which perspectives on the components of S2aaS are to be considered?

During the analysis of IoT architecture proposals in the literature, two new perspectives on IoT architectures became apparent. The *Network IoT Architecture Perspective* focusses on establishing communication between individual components of an IoT architecture in general (see section 4.2.1, page 45) and is related to the *Gateway* component described earlier. The *Organisational IoT*

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Architecture Perspective concentrates on the description and definition of the organisational relationships between components of an IoT architecture (see section 4.2.1, page 48). This perspective is related to the *Publisher* component which was identified as part of the previous research question answered in section 4.2.2.

RQ1.3 What common requirements for S2aaS components can be defined?

By analysing the pertinent literature in section 4.2.2 requirements for each of the components *Consumer, Thing, Owner, Service Provider, Gateway* and *Publisher* have been worked out. The requirements extracted from the literature are illustrated in Table 1. In general, each component is interpreted as an independent software system which exposes specific functionalities via communication endpoints. The detailed requirements for each component are guided by the respective business level interests of the components stakeholders. For example, *Service Providers* offer value added services (e.g. simplifying searching for and visualising sensing data) and consequently need to be able to support various incentive mechanisms. Furthermore, the requirements for the novel component of the *Thing Management System* have been described in section 4.3.3 and illustrated in Table 2 as well as Table 3.

RO2 Map architectural components of S2aaS to existing IoT services, systems and concepts.

RQ2.1 How can existing services, systems and concepts be mapped to components of S2aaS?

This thesis applied the DSR pattern *Problem Space Tools and Techniques* presented by Vaishnavi and Kuechler (2007) to identify existing services, systems and concepts that could be mapped to components of S2aaS. With the research domain of IoT being relatively new and unstructured, the pattern suggested by Vaishnavi and Kuechler (2007) is especially considered useful. This pattern utilises researchers´ general knowledge to identify promising tools and techniques to solve a given research problem. Therefore, it is employed in this thesis to test novel solutions of existing research problems of IoT.

RQ2.2 Which existing services, systems and concepts can be mapped to components of S2aaS?

Based on the *Problem Space Tools and Techniques* pattern identified as part of the previous research question, the concepts and techniques of *Identity Management* have been identified as a suitable baseline or guiding principle to develop the novel component of the *Thing Management System* in section 4.3. By interpreting *Things* as *Identities* referring to their corresponding *Owners* as *Entities*, the principles and concepts of *Identity Management* can be transferred to IoT and used to design a system for managing *Things*.

RO3 Propose detailed specifications for the components of IoT architecture framework.

RQ3.1 What are the specifications for each component?

Having identified a common set of IoT architecture components (*Consumer, Thing, Owner, Service Provider*) as well as two additional components which have been identified by using the novel *IoT Architecture Perspectives* (*Publisher* and *Gateway*) and additionally having developed the *Thing Management System* as a completely new component, the detailed specifications of each of these

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

components are listed in Table 4. Following the intended high-level approach for the *Holistic IoT Architecture Framework* described in section 2.4, the specifications for each component focus on external relations with other components as well as the roles and responsibilities of each component.,

RO4 Find technologies supporting the implementation of the proposed architecture framework.

RQ4.1 Which criteria are important for the selection of technologies that support the implementation of the proposed architecture framework?

Based on the theoretical development and specifications of the components of the Holistic IoT Architecture as part of the previous research questions, it was evaluated if these specifications and descriptions are viable to guide the implementation of a prototype. In order to be able to develop a prototype, several choices regarding the selection of technologies had to be made. The discussion carried out in the beginning of chapter 5 shows that the sufficient support of heterogeneous data is a crucial criterion for selecting appropriate technologies supporting the implementation of such a framework.

RQ4.2 Which technologies are suitable to implement components for the proposed architecture?

Considering the need to support the representation, storage and handling of heterogeneous or schemaless data, the data storage technology for the prototype was selected accordingly. As discussed in the beginning of chapter 5, PostgreSQL was selected as the storage engine for the prototype because it provides sufficient capabilities to handle schemaless data and is simple to integrate into the other technologies of the prototype such as Ruby on Rails. However, it must be noted that while the selection of this storage engine is appropriate for a prototype like this, a large-scale implementation of the *Holistic IoT Architecture* should use technologies that scale better than PostgreSQL.

6.2 Research Contribution

In the course of this thesis, various aspects of IoT architectures proposed in the pertinent literature were interpreted and used to develop a *Holistic IoT Architecture Framework* that covers these various aspects as a whole. During the development of this framework two novel perspectives on IoT architectures in general were identified. These perspectives, the *Network*- and the *Organisational IoT Architecture Perspective*, helped to identify and specify architecture components which are mentioned in the literature, either implicitly or explicitly. In addition, the perspectives could be used as a supplemental tool to classify the focus of existing IoT architecture proposals in the literature. The components *Consumer, Thing, Owner, Service Provider, Publisher,* and *Gateway,* which were identified with the help of these perspectives, have been thoroughly described. Furthermore, by considering the novel IoT architecture perspectives and comparing the differences between the associated architectures, the need of an additional component became apparent. This conclusion was further encouraged by the realisation that *Things* can be interpreted as *Identities* in terms of *Identity Management.* Based on this insight that *Things* are *Identities* which refer to their respective *Owners* and the discrepancies between the *Organisational IoT Architecture Perspective* and the *Network IoT Architecture Perspective*, a novel component was developed which addresses the need to allow *Owners* to manage their *Things*. The

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Thing Management System developed throughout this thesis, which allows *Owners* to share and publish their *Identities* (*or Things),* was subsequently embedded into the overall architecture.

Overall, the *Holistic IoT Architecture Framework* developed in this thesis provides high level specifications and descriptions of IoT architecture components, their relations, roles and responsibilities. By applying this architecture, the expected behaviour and tasks of the IoT architecture components are clearly defined on a high level of abstraction. Based on the understanding of the role of each IoT architecture component and what the semantics of the communication between the individual components are, the implementation of a component is a straightforward process, which has been demonstrated in this thesis as well.

6.3 Limitations

Considering that the evaluation conducted in section 5.4 revealed that dealing with the technical details on a low level of abstraction can potentially have an impact on the architecture in general, the mere focus on a high level of abstraction might be a limitation of this thesis. In section 5.4, the need for some kind of *Component Addressing System* became apparent while evaluating the implementation of prototype of the *Holistic IoT Architecture Framework.*

As the evaluation in section 5.4 has shown, the evaluation on a lower level of abstraction can yield results that have an impact on higher levels. Thus, one limitation of this thesis is the focus on the high-level aspects of an IoT architecture. Additionally, only a subset of IoT architecture proposals has been considered in the analysis (see section 2.3), which inevitably suggests that some possibly relevant aspects addressed by other proposals are not covered by the *IoT architecture framework* developed in this thesis.

Another limitation of this thesis is mentioned in section 4.3.4. The architecture requires that *Gateways* may act on behalf of *Publishers* to gather data from *Things* which have been deployed in foreign environments and thus need to rely on opportunistic and non-permanent communication. However, as has been discussed in section 4.3.4, *Gateways* need to authenticate with *Things* and prove that they act on behalf of the associated *Publisher* of a *Thing* and have indeed the permission to access the *Thing´s* services. The main issue with this connection between *Gateways*, *Publisher* and *Things* is the lack of a suitable authentication mechanism. The mechanism must either provide *Gateways* with credentials that are “valid on their own” and which can be independently validated by a *Thing* or the *Thing* must be able to directly communicate with its assigned *Publisher* to check the relationship between the requesting *Gateway* and corresponding *Publisher*. As discussed in section 4.3.4, both options are equally problematic. Using credentials which are “valid on their own” (e.g. sufficiently encrypted tokens, etc.) is impractical because managing the tokens becomes increasingly difficult with the number of *Things*, *Gateways* and *Publishers* growing. For example, it will be very difficult to revoke access to *Things* when not having access to every *Thing* and to invalidate specific credentials so that access is denied. While

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

this issue has been discussed, a solution was not presented in this thesis. Hence, it is the main limitation of this thesis.

6.4 Future Work

As discussed in the previous section, communication and propagation of access rights and a suitable authentication mechanism between *Gateways*, *Publishers* and *Things* remain issues to be solved in the future. Solving these issues is especially important in order to be able to deploy IoT applications and sensor networks on a large scale. It is inevitable that *Things* are deployed in foreign environments and thus need to rely on opportunistic communication.

Furthermore, with Gartner (2014) forecasting a black market worth five billion US$ by 2020, the need for mechanisms and systems which can assess the quality and authenticity of sensing data becomes apparent. The *Holistic IoT Architecture* defines *Consumers* as components that are interested in sensing data and are willing to provide some means of compensation. The intention to provide compensation for data implies that sensing data has value that needs to be defined. The value of data could be coupled to the reputation of the *Owners*, *Things*, *Publisher* and *Gateways* involved in obtaining this data. Thus, a system must be developed to determine and manage the reputation of all actors involved in gathering sensing data. A system for assessing the reputation of an actor could be integrated into each component of the *Holistic IoT Architecture Framework*.

However, for S2aaS to be successful, a more general issue regarding IoT and sensing data must be addressed. S2aaS is designed to offer access to sensing data as a service, which describes various dimensions of any environment at anytime and anywhere. Furthermore, S2aaS suggests that this service, which is provided by a plethora of actors, needs to be compensated. For this purpose, various incentive mechanisms have been developed. The problem is, however, that research regarding S2aaS stops at this point. Potential use and value of sensing data, which justifies the compensation for gathering this data in the first place, is often disregarded. *Consumers* need methods to assess what kind of data and which amount of data is required in order to be able to derive information and knowledge from this data.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

References

Abdelwahab, S. et al., 2016. Cloud of Things for Sensing-as-a-Service: Architecture, Algorithms, and Use Case. *IEEE Internet of Things Journal*, 3(6), pp.1099–1112. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7457602.

Abdelwahab, S. et al., 2015. Cloud of Things for Sensing as a Service: Sensing Resource Discovery and Virtualization. In *2015 IEEE Global Communications Conference (GLOBECOM)*. IEEE, pp. 1–7. Available at: http://ieeexplore.ieee.org/document/7417252/.

Abdelwahab, S. et al., 2014. Enabling Smart Cloud Services Through Remote Sensing: An Internet of Everything Enabler. *IEEE Internet of Things Journal*, 1(3), pp.276–288. Available at: http://ieeexplore.ieee.org/document/6817547/.

Ahmad, A. et al., 2016. Model-Based Testing as a Service for IoT Platforms. In *Leveraging Applications of Formal Methods, Verification and Validation: Discussion, Dissemination, Applications*. Cham: Springer International Publishing, pp. 727–742. Available at: http://link.springer.com/10.1007/978-3-540-88479-8.

ARM Limited, 2017. CoAP. Available at: http://coap.technology/ [Accessed January 28, 2017].

Ashton, K., 2009. That "Internet of Things" Thing. *RFID Journal*. Available at: http://www.itrco.jp/libraries/RFIDjournal-That Internet of Things Thing.pdf [Accessed February 15, 2017].

Atzori, L., Iera, A. & Morabito, G., 2010. The Internet of Things: A survey. *Computer Networks*, 54(15), pp.2787–2805. Available at: http://dx.doi.org/10.1016/j.comnet.2010.05.010.

Auto-ID Lab, 2017. Auto-ID Labs. Available at: http://autoidlabs.org/ [Accessed January 14, 2017].

Barnaghi, P. et al., 2012. Semantics for the Internet of Things: early progress and back to the future. *International Journal on Semantic Web and Information Systems*, 8(1), pp.1–21. Available at: http://services.igi-global.com/resolvedoi/resolve.aspx?doi=10.4018/jswis.2012010101.

Bassi, A. et al., 2013. *Enabling Things to Talk* A. Bassi et al., eds., Berlin, Heidelberg: Springer Berlin Heidelberg. Available at: http://doi.wiley.com/10.1002/9781118600146.ch1.

Bhargav-Spantzely, A. et al., 2006. User centricity: A Taxonomy and Open Issues. In *Proceedings of the second ACM workshop on Digital identity management - DIM '06*. New York, New York, USA: ACM Press, p. 1. Available at: http://portal.acm.org/citation.cfm?doid=1179529.1179531.

Botterman, M., 2009. *Internet of Things: an early reality of the Future Internet*, Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Internet+of+Things+:+an+early+reality+of+the+Future+Internet#0.

Bradner, S., 1997. Key words for use in RFCs to Indicate Requirement Levels. , pp.1–3. Available at: https://www.ietf.org/rfc/rfc2119.txt.

Burke, J. et al., 2006. Participatory sensing. In *ACM SenSys 2006*. Boulder, Colorado, USA. Available at: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.122.3024.

Cameron, K., 2005. The Laws of Identity. , p.13. Available at: http://www.identityblog.com/stories/2005/05/13/TheLawsOfIdentity.pdf [Accessed September 2, 2016].

Campbell, A.T. et al., 2008. The Rise of People-Centric Sensing. *Internet Computing*, 12(4), pp.12–21.

Chii Chang, Srirama, S.N. & Liyanage, M., 2015. A Service-Oriented Mobile Cloud Middleware Framework for Provisioning Mobile Sensing as a Service. In *2015 IEEE 21st International Conference on Parallel and Distributed Systems (ICPADS)*. Melbourne, Australia: IEEE, pp. 124–131.

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Available at: http://ieeexplore.ieee.org/document/7384287/.

Clegg, D. & Barker, R., 1994. *Case Method Fast-Track: A Rad Approach*, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc.

daCosta, F., 2013. *Rethinking the Internet of Things*, Berkeley, CA: Apress. Available at: http://link.springer.com/10.1007/978-1-4302-5741-7_8.

Efremov, S. et al., 2015. Cloud IoT Platforms: A Solid Foundation for the Future Web or a Temporary Workaround? In S. Balandin, S. Andreev, & Y. Koucheryavy, eds. *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 47–55. Available at: http://link.springer.com/10.1007/978-3-319-23126-6_5.

Emmerich, W., Aoyama, M. & Sventek, J., 2008. The impact of research on the development of middleware technology. *ACM Trans Softw Eng Methodol*, 17(4), pp.1–48. Available at: http://discovery.ucl.ac.uk/46712/.

Fenn, J. & LeHong, H., 2011. *Hype Cycle for Emerging Technologies*, Stamford. Available at: http://www.gartner.com/newsroom/id/2819918%5Cnhttps://www.google.be/url?sa=t&rct=j&q=&esrc=s&source=web&cd=9&cad=rja&uact=8&ved=0ahUKEwjxzq6mnr3JAhUGPQ8KHR-WBGgQFghRMAg&url=http://isites.harvard.edu/fs/docs/icb.topic1360759.files/hype_cycle_for_emerging_t.

Ferna et al., 2015. *Ubiquitous Computing and Ambient Intelligence. Sensing, Processing, and Using Environmental Information* J. M. García-Chamizo, G. Fortino, & S. F. Ochoa, eds., Cham: Springer International Publishing. Available at: http://link.springer.com/10.1007/978-3-642-35377-2.

Fleisch, E., 2010. What is the Internet of Things? An Economic Perspective (white paper). In Zurich, Switzerland: Auto-ID Labs, pp. 1–27.

Fremantle, P. et al., 2014. Federated Identity and Access Management for the Internet of Things. In *2014 International Workshop on Secure Internet of Things*. IEEE, pp. 10–17. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-84926431695&partnerID=40&md5=2726c8ea5ad90df99e17f7e8099ca815.

Gamma, E. et al., 1995. *Design Patterns Elements of Reusable Object-Oriented Software*, Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc. Available at: http://www.cs.up.ac.za/cs/aboake/sws780/references/patternstoarchitecture/Gamma-DesignPatternsIntro.pdf.

Gartner, 2016a. Gartner Says By 2020, More Than Half of Major New Business Processes and Systems Will Incorporate Some Element of the Internet of Things. *Gartner.com*. Available at: http://www.gartner.com/newsroom/id/3185623 [Accessed February 10, 2017].

Gartner, 2013. Hype Cycle for the Internet of Things, 2013. , (July), pp.1–4. Available at: http://www.gartner.com/newsroom/id/2575515%0ASTAMFORD, [Accessed September 2, 2016].

Gartner, 2014. Hype Cycle for the Internet of Things, 2014. , (July), pp.1–4. Available at: http://www.gartner.com/newsroom/id/2819918 [Accessed September 2, 2016].

Gartner, 2016b. Hype Cycle for the Internet of Things, 2016. , pp.16–18. Available at: http://www.gartner.com/newsroom/id/3412017 STAMFORD, [Accessed September 2, 2016].

Gero, J.S., 2000. Research Methods for Design Science Research: Computational and Cognitive Approaches. *Proceedings of ANZAScA*, (November 1999). Available at: http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.86.7198&rep=rep1&type=pdf.

Geschickter, C., 2015. *A CIO ' s Guide to Realizing the Business Value of IoT , Part 2 : Scope , Deployment Options and Pilot*, Stamford.

 © 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Gregor, S. & Hevner, A.R., 2013. POSITIONING AND PRESENTING DESIGN SCIENCE Types of Knowledge in Design Science Research. *MIS Quarterly*, 37(2), pp.337–355.

Ha, T., Lee, S. & Kim, N., 2015. Development of a User-Oriented IoT Middleware Architecture Based on Users' Context Data. In N. Streitz & P. Markopoulos, eds. *Distributed, Ambient, and Pervasive Interactions*. Lecture Notes in Computer Science. Cham: Springer International Publishing, pp. 287–295. Available at: http://www.scopus.com/inward/record.url?eid=2-s2.0-84901599558&partnerID=tZOtx3y1.

Hui, J. & Corporation, A.R., 2009. *6LoWPAN: Incorporating IEEE 802.15.4 into the IP architecture*, Berkeley.

IBM, 2016a. Create IBM Watson IoT Platform Organization. Available at: https://developer.ibm.com/recipes/tutorials/how-to-register-devices-in-ibm-iot-foundation/ [Accessed July 11, 2016].

IBM, 2016b. Track-n-Trace. Available at: https://developer.ibm.com/recipes/tutorials/track-n-trace-7/ [Accessed July 11, 2016].

INFSO D.4 Networked Enterprise et al., 2008. *Internet of Things in 2020 A ROADMAP FOR THE FUTURE*, Berlin. Available at: http://www.smart-systems-integration.org/public/documents/publications/Internet-of-Things_in_2020_EC-EPoSS_Workshop_Report_2008_v3.pdf.

International Telecommunication Union, 2005. *The Internet of Things*, Geneva.

IoT Analytics, 2015. *IoT Platforms The central backbone for the Internet of Things*, Available at: http://www.iot-analytics.com.

Ishaq, I. et al., 2013. *IETF Standardization in the Field of the Internet of Things (IoT): A Survey*, Available at: http://www.mdpi.com/2224-2708/2/2/235/htm.

Jahankhani, H. et al., 2010. *Handbook of electronic security and digital forensics* 1st ed., Singapore: World Scientific. Available at: https://books.google.de/books?hl=de&lr=&id=ZgpV6Rvw2FoC&oi=fnd&pg=PA279&dq=digital+identity+management&ots=QWFHl-srib&sig=ZHhLmNipX0wxKYGzt2TPTKJdeeQ#v=onepage&q=digital identity management&f=false.

Jo, M. et al., 2015. Device-to-device-based heterogeneous radio access network architecture for mobile cloud computing. *IEEE Wireless Communications*, 22(3), pp.50–58. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=7143326.

Johnson, F.A. et al., 2009. *The EPCglobal Architecture Framework*, Brussels, Belgium.

Jøsang, A. & Pope, S., 2005. User centric identity management. In *AusCERT Asia Pacific Information Technology Security Conference*. pp. 1–13. Available at: http://web.mac.com/skjpope/downloads/files/usercentric.pdf.

Katasonov, A. et al., 2008. Smart Semantic Middleware for the Internet of Things. *Icinco-Icso*, 1(August), pp.169–178. Available at: http://www.mit.jyu.fi/ai/papers/ICINCO-2008.pdf.

Khan, R. et al., 2012. Future internet: The internet of things architecture, possible applications and key challenges. In *Proceedings - 10th International Conference on Frontiers of Information Technology, FIT 2012*. pp. 257–260.

Kim, J. et al., 2014. M2M service platforms: Survey, issues, and enabling technologies. *IEEE Communications Surveys and Tutorials*, 16(1), pp.61–76.

Krause, A. et al., 2008. Toward Community Sensing. In *2008 International Conference on Information Processing in Sensor Networks (ipsn 2008)*. IEEE, pp. 481–492. Available at: http://ieeexplore.ieee.org/document/4505497/.

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Kuechler, Vaishnavi & Petter, 2005. The Aggregate General Design Cycle as a Perspective on the Evolution of Computing Communities of Interest. *Computing Letters*, 1(3), pp.123–128. Available at: http://booksandjournals.brillonline.com/content/journals/10.1163/1574040054861221.

Lee, I. & Lee, K., 2015. The Internet of Things (IoT): Applications, investments, and challenges for enterprises. *Business Horizons*, 58(4), pp.431–440. Available at: http://dx.doi.org/10.1016/j.bushor.2015.03.008.

Linden, A. & Fenn, J., 2003. *Understanding Gartner's hype cycles*, Stamford. Available at: http://www.ask-force.org/web/Discourse/Linden-HypeCycle-2003.pdf.

Madakam, S., Ramaswamy, R. & Tripathi, S., 2015. Internet of Things (IoT): A Literature Review. *Journal of Computer and Communications*, 3(5), pp.164–173. Available at: http://www.scirp.org/journal/PaperInformation.aspx?PaperID=56616&#abstract.

Mashal, I. et al., 2015. Choices for interaction with things on Internet and underlying issues. *Ad Hoc Networks*, 28(November), pp.68–90. Available at: http://linkinghub.elsevier.com/retrieve/pii/S1570870514003138.

Mashal, I., Alsaryrah, O. & Chung, T.-Y., 2016. Testing and evaluating recommendation algorithms in internet of things. *Journal of Ambient Intelligence and Humanized Computing*, 7(6), pp.889–900. Available at: http://link.springer.com/10.1007/s12652-016-0357-4.

Mazhelis, O. et al., 2013. *Internet-of-Things Market , Value Networks , and Business Models : State of the Art Report*, JYVÄSKYLÄ.

Miao Wu et al., 2010. Research on the architecture of Internet of Things. In *2010 3rd International Conference on Advanced Computer Theory and Engineering(ICACTE)*. IEEE, pp. V5-484-V5-487. Available at: http://ieeexplore.ieee.org/document/5579493/.

Mineraud, J. et al., 2016. A gap analysis of Internet-of-Things platforms. *Computer Communications*, 89–90(1), pp.5–16. Available at: http://linkinghub.elsevier.com/retrieve/pii/S0140366416300731.

Mizouni, R. & El Barachi, M., 2013. Mobile phone sensing as a service: Business model and use cases. *International Conference on Next Generation Mobile Applications, Services, and Technologies*, pp.116–121.

Moreno-Vozmediano, R., Montero, R.S. & Llorente, I.M., 2013. Key challenges in cloud computing: Enabling the future internet of services. *IEEE Internet Computing*, 17(4), pp.18–25.

Mosser, S. et al., 2012. SENSAPP as a Reference Platform to Support Cloud Experiments: From the Internet of Things to the Internet of Services. In *2012 14th International Symposium on Symbolic and Numeric Algorithms for Scientific Computing*. IEEE, pp. 400–406. Available at: http://ieeexplore.ieee.org/document/6481058/.

Naur, P. & Randell, B., 1968. Software Engineering: Report of a conference sponsored by the NATO Science Committee. In *NATO Software Engineering Conference*. p. 231. Available at: http://homepages.cs.ncl.ac.uk/brian.randell/NATO/index.html.

Al Nuaimi, K. et al., 2012. Web-based wireless sensor networks. In *Proceedings of the 6th International Conference on Ubiquitous Information Management and Communication - ICUIMC '12*. New York, New York, USA: ACM Press, p. 1. Available at: http://dl.acm.org/citation.cfm?doid=2184751.2184881.

Park, G.-J., 2007. Axiomatic Design. In *Analytic Methods for Design Practice*. London: Springer London, pp. 17–105. Available at: http://link.springer.com/10.1007/978-1-84628-473-1.

Perera, C. et al., 2013. Context-Aware Sensor Search, Selection and Ranking Model for Internet of Things Middleware. In *2013 IEEE 14th International Conference on Mobile Data Management*. IEEE, pp. 314–322. Available at: http://ieeexplore.ieee.org/document/6569153/.

 © 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Perera, C. et al., 2015. Energy-Efficient Location and Activity-Aware On-Demand Mobile Distributed Sensing Platform for Sensing as a Service in IoT Clouds. *IEEE Transactions on Computational Social Systems*, 2(4), pp.171–181. Available at: http://arxiv.org/abs/1601.00428.

Perera, C., Jayaraman, P.P., et al., 2014. MOSDEN: An Internet of Things Middleware for Resource Constrained Mobile Devices. In *2014 47th Hawaii International Conference on System Sciences*. IEEE, pp. 1053–1062. Available at: http://ieeexplore.ieee.org/document/6758734/.

Perera, C., Zaslavsky, A., Christen, P., et al., 2014. Sensing as a service model for smart cities supported by Internet of Things. *Transactions on Emerging Telecommunications Technologies*, 25(1), pp.81–93. Available at: http://doi.wiley.com/10.1002/ett.2704.

Perera, C., Zaslavsky, A., Liu, C.H., et al., 2014. Sensor Search Techniques for Sensing as a Service Architecture for the Internet of Things. *IEEE Sensors Journal*, 14(2), pp.406–420. Available at: http://ieeexplore.ieee.org/document/6605518/.

Petrolo, R. et al., 2017. The design of the gateway for the Cloud of Things. *Annals of Telecommunications*, 72(1–2), pp.31–40. Available at: http://link.springer.com/10.1007/s12243-016-0521-z.

Pettey, C. & Vandermeulen, R., 2012. *Hype Cycle for the Internet of Things, 2012*, Stamford.

Rothensee, M., 2008. User Acceptance of the Intelligent Fridge: Empirical Results from a Simulation. In *The Internet of Things*. Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 123–139. Available at: http://portal.acm.org/citation.cfm?id=1793071&CFID=1793840&CFTOKEN=33112955.

Sarma, A.C. & Girão, J., 2009. Identities in the Future Internet of Things. *Wireless Personal Communications*, 49(3), pp.353–363. Available at: http://link.springer.com/10.1007/s11277-009-9697-0.

Serdaroglu, K.C. & Baydere, S., 2016. WiSEGATE: Wireless Sensor Network Gateway framework for internet of things. *Wireless Networks*, 22(5), pp.1475–1491. Available at: http://link.springer.com/10.1007/s11276-015-1046-5.

Sheng, X. et al., 2012. Sensing as a service: A cloud computing system for mobile phone sensing. In *2012 IEEE Sensors*. IEEE, pp. 1–4. Available at: http://ieeexplore.ieee.org/lpdocs/epic03/wrapper.htm?arnumber=6411516.

Sheng, X. et al., 2013. Sensing as a Service: Challenges, Solutions and Future Directions. *IEEE Sensors Journal*, 13(10), pp.3733–3741. Available at: http://ieeexplore.ieee.org/document/6515322/.

Solis, C. & Wang, X., 2011. A Study of the Characteristics of Behaviour Driven Development. In *2011 37th EUROMICRO Conference on Software Engineering and Advanced Applications*. IEEE, pp. 383–387. Available at: http://ieeexplore.ieee.org/document/6068372/.

Sone, M., 2001. Household consumable item automatic replenishment system including intelligent refrigerator. Available at: https://www.google.com/patents/US6204763.

Stanford-Clark, A. & Nipper, A., 2017. MQTT. Available at: http://mqtt.org/ [Accessed January 28, 2017].

Sterling, B., 2005. *Shaping Things*, Londong, England: The MIT Press. Available at: http://www.amazon.com/Shaping-Things-Mediaworks-Pamphlets-Sterling/dp/0262693267.

Suh, N.P. & Do, S.-H., 2000. Axiomatic Design of Software Systems. *CIRP Annals - Manufacturing Technology*, 49(1), pp.95–100. Available at: http://www.axiomaticdesign.com/technology/ADSChapter5.html.

Takeda, H. et al., 1990. Modeling Design Processes. *AI Magazine*, 11(4), pp.37–48.

Tarkoma, S. & Katasonov, A., 2011. *Internet of Things Strategic Research Agenda*, Helsinki, Finnland.

The National Intelligence Council, 2008. Disruptive Civil Technologies Six Technologies With Potential

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Impacts on US Interests Out to 2025. *National Intelligence Council*, 59(April), p.48. Available at: http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Disruptive+Civil+Technologies+Six+Technologies+With+Potential+Impacts+on+%7BUS%7D+Interests+Out+to+2025#1.

Tima, I., Simperl, E. & Hench, G., 2009. A joint roadmap for Semantic technologies and the Internet of Things. In *Proceedings of the Third STI Roadmapping Workshop*. Crete, Greece. Available at: http://link.springer.com/10.1007/s11277-011-0288-5.

Vaishnavi, V. & Kuechler, B., 2004. Design Science Research in Information Systems Overview of Design Science Research. *Ais*, p.45. Available at: http://www.desrist.org/design-research-in-information-systems/.

Vaishnavi, V. & Kuechler, W., 2007. *Design Science Research Methods and Patterns*, Boca Raton, USA: Auerbach Publications. Available at: http://www.crcnetbase.com/doi/book/10.1201/9781420059335.

Vazquez, J.I. & Lopez-de-ipina, D., 2008. Social Devices : Autonomous Artifacts That Communicate on the Internet. In pp. 308–324.

Velosa, A., Schulte, R.W. & Lheureux, B.J., 2015. Hype Cycle for the Internet of Things, 2015. , (July), pp.1–69. Available at: http://www.gartner.com/document/3098434?ref=solrAll&refval=161158590&qid=40f7a175899f78812787998fff35a614.

Wortmann, F. & Flüchter, K., 2015. Internet of Things. *Business & Information Systems Engineering*, 57(3), pp.221–224. Available at: http://link.springer.com/10.1007/s12599-015-0383-3.

Yang, D. et al., 2012. Crowdsourcing to smartphones. In *Proceedings of the 18th annual international conference on Mobile computing and networking - Mobicom '12*. New York, New York, USA: ACM Press, p. 173. Available at: http://dl.acm.org/citation.cfm?doid=2348543.2348567.

Yasrab, R. & Gu, N., 2016. Multi-cloud PaaS Architecture (MCPA): A Solution to Cloud Lock-In. In *2016 3rd International Conference on Information Science and Control Engineering (ICISCE)*. IEEE, pp. 473–477. Available at: http://ieeexplore.ieee.org/document/7726205/.

Zachariah, T. et al., 2015. The Internet of Things Has a Gateway Problem. In *Proceedings of the 16th International Workshop on Mobile Computing Systems and Applications - HotMobile '15*. New York, New York, USA: ACM Press, pp. 27–32. Available at: http://web.eecs.umich.edu/~prabal/pubs/papers/zachariah15gateway.pdf.

Zaslavsky, A., Perera, C. & Georgakopoulos, D., 2013. Sensing as a Service and Big Data. *Proceedings of the International Conference on Advances in Cloud Computing (ACC-2012)*, pp.21–29. Available at: http://arxiv.org/abs/1301.0159.

Zorzi, M. et al., 2010. From today's INTRAnet of things to a future INTERnet of things: a wireless- and mobility-related view. *IEEE Wireless Communications*, 17(6), pp.44–51. Available at: http://ieeexplore.ieee.org/document/5675777/.

 © 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Appendix

Appendix 1: Source code of the prototype

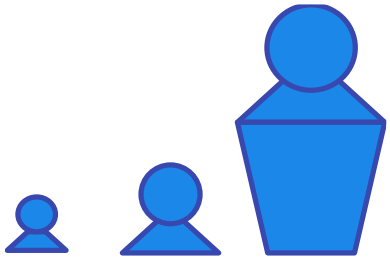
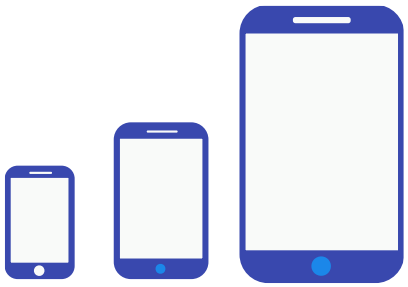
The source code of the prototype developed in chapter 5 is available at the following addresses. Each component is implemented as a separate web application based on Ruby on Rails and hosted in its own repository.

Thing Management System	https://gitlab.uni-koblenz.de/msc/msc-tms
Service Provider	https://gitlab.uni-koblenz.de/msc/msc-service-provider
Publisher	https://gitlab.uni-koblenz.de/msc/msc-publisher
Thing	https://gitlab.uni-koblenz.de/msc/msc-demo_thing

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

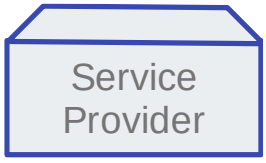
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

Appendix 2: Overview of the icons used in the illustrations

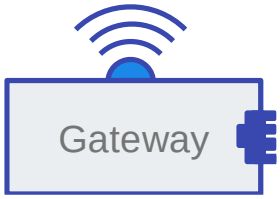
	• User of a system or service • Stakeholder of a system or service
	• User with a mobile device
	• Mobile device • Smartphone
	• Identity of a user
	• Credentials of a user
	• Identity of a *Thing*
	• *Thing*

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group
IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»

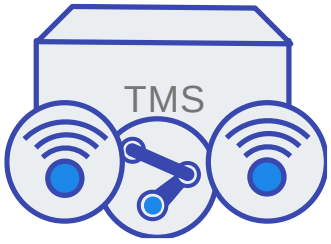
- *Publisher* of the *Holistic IoT Architecture Framework*



- *Service Provider* of the *Holistic IoT Architecture Framework*



- *Gateway* of the *Holistic IoT Architecture Framework*



- *Thing Management System* of the *Holistic IoT Architecture Framework*

© 2017 University Koblenz-Landau, Enterprise Information Management Research Group

IOT IRAN
مرکز تحقیقات اینترنت اشیا
«نمـــایه‌ســـازی علـــمی»