Chapter 12
ARM Testimonials

Edward Ho, Tobias Jacobs, Stefan Meissner, Sonja Meyer, Miguel-Angel Monjas, and Alexander Salinas Segura

This chapter shows how the IoT ARM is perceived by the IoT community and how the ARM can be placed in relation to existing IoT related standards and research projects. The first sections of this chapter present reverse mappings of existing standards and platforms to the IoT ARM and the last section of this chapter shows a business case evaluation for an example use case in the healthcare domain.

E. Ho (✉)
University of St.Gallen, Dufourstrasse 40a, CH-9000 St.Gallen, Switzerland
e-mail: edward.ho@unisg.ch

T. Jacobs
NEC Laboratories Europe, Software & Services Research Division, NEC Europe Ltd., Kurfürsten-Anlage 36, 69115 Heidelberg, Germany
e-mail: tobias.jacobs@neclab.eu

S. Meissner
University of Surrey, Stag Hill, GU2 7XH Guildford, UK
e-mail: s.meissner@surrey.ac.uk

S. Meyer
SAP AG, Dietmar-Hopp-Allee 16, 69190 Walldorf, Germany
e-mail: sonj.meyer@sap.com

Miguel-Angel Monjas
Ericsson Spain, Madrid R&D Center, Technology and Innovation, Vía de los Poblados, 13, 28033 Madrid, Spain
e-mail: Miguel-Angel.Monjas@ericsson.com

A. Salinas Segura
University of Würzburg, Josef-Stangl-Platz 2, 97070 Würzburg, Germany
e-mail: alexander.salinas@uni-wuerzburg.de

A. Bassi et al. (eds.), *Enabling Things to Talk*, DOI 10.1007/978-3-642-40403-0_12,

12.1 Introduction to Reverse Mapping

In the course of its own project roadmap, our sister project – the *Internet of Things Initiative* (IoT-i) – has targeted three different (but connected) activities directly relating to the IoT-A architecture work as shown below:

1. To review and categorise existing reference models having a connection to the IoT field (or underlying disciplines, as IoT as such is more a technology umbrella). Example of the reference models reviewed by IoT-i are ETSI M2M, IETF Core, EPCglobal, Ucode and NFC to name just a few (IoT-A D1.2);
2. To put online a survey, the goal of which was to capture, people understanding and expectation, as far as reference models are concerned. This exercise was very important because people have generally different understanding about what are reference models, architectures and what they should consist of;
3. Finally, to come back on reference models introduced and summarised in previous versions of this deliverable and to do a reverse mapping exercise towards the IoT Reference Model. The goal of this exercise was to show that the reference model as defined by IoT-A is expressive enough in order to allow a modelling of those (pre- IoT-A) existing IoT reference models using the IoT-A one. In other words, if we would consider that IoT-A does not attempt to define what is an IoT system using sentences and words, but defining models where any IoT system (from the IoT understanding) shall fit, then all those existing reference models would be IoT systems reference models.

In this Section we aim at giving some details about this reverse mapping exercise applied to ETSI M2M, EPCglobal and uID. Some of the material in this Section comes directly from the IoT-A D1.5 deliverable (Carrez et al. 2013) (especially the UML figures and concept tables). In order to improve readability, we do not use direct citations, although the work presented in the following Section was performed by the IoT-A project and reported in their deliverable D1.5.

In addition to the standards that we have mentioned above, we also apply the IoT Architectural Reference Model to a concrete architecture, namely the architecture of the MUNICH (MUNICH 2010) project in order to validate the IoT ARM against a real system in contrast to an abstract standard. Furthermore we show a reverse mapping to the information model of the IoT-related research project BUTLER[1].

12.2 Reverse Mapping ETSI M2M

Within the IoT-A D1.5 deliverable, Sect. 3.1.1 discusses the ETSI M2M standard (ETSI TS 102 690). In this section we analyse the ETSI M2M standard. The acronym ETSI stands for *European Telecommunications Standards Institute*

[1] http://www.iot-butler.eu/

(ETSI), viz. the standardisation body responsible for this standard. The acronym M2M stands for Machine-to-Machine, which is a pointer to the application field this standard addresses, viz. machine-to-machine communications. Release 1 of this standard was published in October 2011 (ETSI TS 102 690), this discussion within IoT-A also takes the later update (ETSI TS 103 092) into account that was released in May 2012.

The purpose of the ETSI M2M functional architecture is to define a service-capability layer which serves as middleware between applications in the Internet and Devices or gateways residing in local-area networks. The current release is mainly concerned with secure and reliable data transport.

In what follows, we give a more detailed description of a possible reverse mapping of ETSI M2M to the IoT Domain Model and IoT Communication Model as well as their management information model and how it maps to our management model. We also have a brief look at the ETSI M2M security model and how it compares to our threat analysis.

12.2.1 Mapping to the IoT Domain Model

As everything above the ETSI M2M Service Capability Layer is considered an application, there is no explicit concept of a User in ETSI M2M. In particular, Human Users are out of scope, as the standard focuses on machine-to-machine communication. The role of an IoT-A User would typically be taken by ETSI network applications, in some cases also by ETSI gateway applications, because these applications use the information provided by sensing M2M Devices and control the actuation capabilities of Devices.

ETSI M2M defines Sensors and Actuators in a similar way as the IoT Domain Model. However, there is a subtle difference regarding the concept of a Device. While in IoT-A there is a "is-a" relationship between Sensor/Actuator and Device, ETSI M2M defines a Device to be a unit comprising Sensors and Actuators, as well as embedded processing and communication capabilities – so here Sensors and Actuators are part of Devices.

The ETSI M2M defines a Service Capability Layer with standardised interfaces. Since this layer includes similar functionalities to the IoT-A Service level (e.g. registration), it is reasonable to map these functionalities to IoT Services. There are also some differences between ETSI and IoT-A terminology. For example, the ETSI Services are not only exposed towards actors which IoT-A would consider as Users, but also towards (ETSI) applications residing on Devices. Additionally, the concept of IoT Resource (IoT-A) as a native software interface of Devices does not explicitly exist in ETSI M2M – although software components on legacy Devices could be considered as IoT Resources. Instead the term of Resources in ETSI is exclusively used to describe the RESTful interface exposed by the Service Capability Layer.

Table 12.1 Mapping ETSI M2M concepts to the IoT-A Domain Model

ETSI M2M	IoT Domain Model	Comments
Device	Device	Sensors and Actuators are hosted on Devices, they are not special cases of Devices
Sensor	Sensor	The Sensor in ETSI M2M is not a Device
Actuator	Actuator	The Actuator in ETSI M2M is not a Device
Network application	User	In ETSI M2M, there are no Human Users, but only applications that process the data coming from the "Device and Gateway Domain". This concept of an application as a User is reflected in IoT-A
Gateway application	User	In ETSI M2M, there are no Human Users, but only applications that process the data coming from the "Device and Gateway Domain". This concept of an application as a User is reflected in IoT-A
Service	Service	In ETSI M2M, Services are not defined as exposing Resources on Devices, but can interact with the Devices. A Resource concept as in IoT-A does not exist
Resource	Service	Resources in ETSI M2M are defined in analogy to RESTful Service Interfaces

The mapping of ETSI M2M concepts to the IoT Domain Model is shown in Table 12.1.

As the current ETSI M2M release is rather concerned with data transport than with real-world modelling, the (physical, virtual, augmented) entity concept is not defined in (ETSI TS 103 092).

12.2.2 Mapping to the Management FG

Management functionalities are an inherent part of both of IoT-A and ETSI M2M. Both architectures distribute and cluster the management functions into different packages or functional components.

In (ETSI TS 102 690), the following packages are defined for management:

- General Management (GEN): Allows retrieving general information of the M2M Device or gateway, and provides generic mechanism applicable to different specific management functions;
- Configuration Management (CFG): Allows configuration of the device capabilities and features for supporting M2M Services and applications, including activating/deactivating device hardware components or I/Os in the M2M Device or gateway;
- Diagnostic & Monitoring Management (D&M): Allows running specific diagnostic tests on a device and collecting the results or alerts from the M2M Device or gateway. This package is also called Fault and Performance Management;

- Software/Firmware Management (SFW): Allows installation/update/removal of application specific or SCL related software/firmware in M2M Device or gateway;
- Area Network Management (ANW): Allows M2M Gateway-specific configuration and M2M Area Network and Device management through a M2M gateway;
- SCL Management (SCL): Allows remote configuration and retrieval of M2M Device or gateway service capability layer parameters.

In a similar fashion, Sect. 8.2.2 of this document identifies different Functional Components used for management functionalities. These include:

- **Configuration**: Initialising the system configuration. Gathering and storing configurations from FCs and Devices, tracking configuration changes;
- **Fault**: The goal of the Fault FC is to identify, isolate, correct and log faults that occur in the IoT system;
- **Member**: This FC is responsible for the management of the membership and associated information of any relevant entity (FG, FC, VE, IoT Service, Device, Application, and User) to an IoT system;
- **Reporting**: The Reporting FC can be seen as an overlay for the other Management FCs. It distils information provided by them. One of many conceivable reporting goals is to determine the efficiency of the current system;
- **State**: The State FC monitors and predicts state of the IoT system. For a ready diagnostic of the system, as required by Fault FC, the past, current and predicted (future) state of the system are provided.

When mapping these different management components, it becomes obvious once again that the focus of ETSI M2M is narrower in terms of its scope and therefore it is more detailed in the definition of its management capabilities and does not include all of the functionality defined by IoT-A. For instance, there is no equivalent to State FC in terms of its temporal distribution and the related billing capabilities. This aspect is not really central, as it is not contradictory and could be built upon the D&M package. In general however there is a strong overlap, as D&M roughly relates to the Reporting FC, CFG closely resembles Configuration, and both D&M and Fault deal with monitoring functionalities. Error and fault handling as such is handled specifically in the Fault FC, whereas D&M also handles performance management. On the other hand, and in line with the general focus of ETSI-M2M, the different aspects of the Configuration FC are handled in more specific packages in ETSI M2M, such as SCL, ANW, and SFW which each deal with specific functionalities that are subsumed under Configuration in IoT-A. As the different architectures naturally have different levels of abstraction, it is not surprising to not have a 1:1 relationship between the two architectures, but a mapping can be performed easily in both directions.

12.2.3 Mapping to the IoT Communication Model

The ETSI M2M standard defines a Service Capability Layer in order to enable seamless, secure, and reliable end-to-end communication in M2M networks. The ETSI Service Capability Layer can therefore be mapped to the end-to-end layer of the IoT Communication Model (see Sect. 7.6). (ETSI) applications, communicating via the Service Capability Layer, would accordingly be associated to the IoT-A Data Layer (see Sect. 7.6.2), although they do not only exchange data, but also control and management information.

A Network and ID group (see Sect. 7.6.2) is not in the focus of ETSI M2M, and the current bindings to HTTP and CoAP do not assume such a layer. However, in cases where the Service Capability Layer enables a direct connection of mobile Device applications to network applications, an ID layer that describes the Device independently from its network location could assist the Service Capability which provides seamless connectivity. The three communication layers at the bottom of Fig. 7.17 can be considered as identical in ETSI M2M and IoT-A.

From the point of view of ETSI M2M, all actors making use of the Service Capability Layer are applications. The model distinguishes between device applications, gateway applications, and network applications. ETSI M2M also considers so-called legacy Devices; these are Devices that have no own Service Capability Layer and therefore need to be integrated via a gateway application into the M2M system (M2M system is a term implicitly used in ETSI M2M to refer to the overall architecture).

The IoT-A term IoT Device is used more or less in the same way in ETSI M2M, but the concept of an IoT Application does not directly exist in ETSI M2M – mainly because the concept of an application is more broadly defined in ETSI M2M.

12.2.4 Mapping to the Security Model

One of the purposes of the ETSI M2M Service Capability Layer is to address all security requirements of M2M communication. The standard defines a key hierarchy of three levels. The ETSI M2M Root Key is used for mutual authentication between device or gateway nodes and the M2M Service Provider. It is also used for deriving and agreeing on the key of the next layer of the hierarchy – the ETSI M2M Connection Key which is used for every service connection procedure. Finally, the ETSI M2M Application Key is used for securing sessions between specific applications. This largely maps to the IoT-A Key exchange and management functionality in IoT-A with respect to key management is not yet explicitly defined in this document.

Most of the communication security and Service security aspects of the IoT-A security model are implicitly addressed in ETSI M2M – although the terminology of IoT-A is not explicitly used. The ETSI M2M standard describes a range of variants that depend on the security characteristics of the underlying network layers and on the relationships between the M2M service provider and the network operator. For example, if these stakeholders are identical, key provisioning can be significantly simplified. One issue clearly not addressed in ETSI M2M are trust models.

12.2.5 Threat Analysis Mapping

(ETSI TR 103 167) deals with a threat analysis related to the ETSI M2M standard. In a similar way as the risk analysis provided in this document in Section. [Chapter 6 Sect. 6.8] ETSI M2M defines those threats that are most relevant for the standard, and discusses respective countermeasures. Here, the different focus of ETSI M2M in terms of network security becomes obvious again, because most of the threats identified by ETSI M2M deal with keys or message exchange. That means that the scope of IoT-A is broader, as it also includes, for instance, Human Users that do not behave correctly. Consequently, IoT-A refers to a general risk analysis that includes by definition non-malicious behaviour that still imposes a risk on the system. As the scope of IoT-A is broader, not all the risks identified within IoT-A are applicable to ETSI M2M, but the threats of ETSI M2M map well to the risks identified within Sect. 6.8. This is shown in Table 12.2 below.

As we can see in Table 12.2, there is a slight difference between both models regarding the consequence or the cause of a risk, as ETSI M2M has a stronger focus on what actions are actually applied in order to impose a risk on the system, whereas IoT-A focuses more on the consequences of these actions. Nevertheless, there is a good mapping between the two models. The granularity of ETSI M2M is naturally higher, as it focuses on a more narrow class of threats.

12.2.6 Conclusion

If we consider that the aim of the ETSI M2M standard is to provide an M2M architecture with a generic set of capabilities for M2M Services and to provide a framework for developing Services independently of the underlying network, it becomes clear that the scope of IoT-A is much broader, taking the entire Internet of Things domain into account, esp. by explicitly modelling entities and also providing a much more fine-grained set of relationships between the different

Table 12.2 Mapping ETSI M2M threat analysis to the IoT-A risk analysis

ETSI M2M	IoT-A
Threat 1: Discovery of long-term service-layer keys stored in M2M devices or M2M gateways	Attacker gains knowledge of sensitive exchanged data Disclosure of identities and cryptographic material
Threat 2: Deletion of long-term service-layer keys stored in M2M devices or M2M gateways	Disruption of a global Service
Threat 3: Replacement of long-term service-layer keys stored in M2M devices or M2M gateways	Disruption of a global Service
Threat 4: Discovery of long-term service-layer keys stored in the SCs of the M2M core	Attacker gains knowledge of sensitive exchanged data Disclosure of identities and cryptographic material
Threat 5: Deletion of long-term service-layer keys stored in the SCs of an M2M core	Disruption of a global Service
Threat 6: Discovery of long-term service-layer keys stored in MSBF or MAS	Attacker gains knowledge of sensitive exchanged data Disclosure of identities and cryptographic material
Threat 7: Deletion of long-term service-layer keys stored in the MSBF/MAS	Attacker gains knowledge of sensitive exchanged data Disclosure of identities and cryptographic material
Threat 8: Discover keys by eavesdropping on communications between entities	Attacker gains knowledge of sensitive exchanged data Disclosure of identities and cryptographic material
Threat 9: Modification of data stored in the M2M service capabilities	Alteration of the return value upon service invocation Attacker alters leaf-device content so that a user will eventually be redirected to a malicious content Attacker alter sensor device so that monitoring of a Physical Entity fails
Threat 10: Provisioning of non-legitimate keys	Disruption of a global Service
Threat 11: Unauthorised or corrupted application and service-layer software in M2M	Attacker impersonates infrastructure Services, compromising IoT functionalities and/or other dependent infrastructure services
Threat 12: Subverting the M2M device/gateway integrity-checking procedures	Alteration of the invocation of a Service
Threat 13: Unauthorised or corrupted software in M2M core	Attacker impersonates infrastructure Services, compromising IoT functionalities and/or other dependent infrastructure services
Threat 14: Subverting the integrity-checking procedures in the M2M core	Alteration of the invocation of a Service
Threat 15: General eavesdropping on M2M service-layer messaging between entities	Attacker gains knowledge of sensitive exchanged data

(continued)

Table 12.2 (continued)

ETSI M2M	IoT-A
Threat 16: Alteration of M2M service-layer messaging between entities	Alteration of the invocation of a Service
Threat 17: Replay of M2M service-layer messaging between entities	Compromised intermediary devices alter traversing data
	Alteration of the invocation of a Service
Threat 18: Breach of privacy due to inter-application communications	User is involved in transactions with a malicious peer
	Attacker gains knowledge of user private parameters
Threat 19: Breach of privacy due to attacks on M2M device/gateway service capabilities	User is involved in transactions with a malicious peer
	Attacker gains knowledge of user private parameters
Threat 20: Discovery of M2M long-term service-layer keys from knowledge of access-network keys	Attacker gains knowledge of sensitive exchanged data
	Disclosure of identities and cryptographic material
Threat 21: Transfer of module containing access-network keys and/or M2M long-term keys to a different terminal/device/gateway	Attacker gains knowledge of sensitive exchanged data
	Disclosure of identities and cryptographic material

kinds of devices, resources and services. While ETSI M2M makes different assumptions, especially in terms of security and communication, the basic concepts are somewhat compatible, at least on an abstract level of discussion. The major difference is that IoT-A is based on the assumption that the IoT Device space can be divided into the two main categories of constrained networks (NTU) and unconstrained networks (NTC), and the security measurements mainly need to address the boundaries between them, whereas ETSI focusses so far on the M2M Service Layer and its interfaces (ETSI TR 103 167) and not on the M2M Area Network Layer, so that IoT-A has a more network centred view of security than ETSI M2M. That being said, the functionalities discussed in Sect. 6.8 largely represent in (ETSI TR 103 167; Sect. 10.2), so that a mapping is feasible on the same abstraction level as the IoT Domain Model can be mapped to the ETSI M2M Service Capability Layer.

12.3 Reverse Mapping EPCglobal

The EPCglobal high-level architecture was introduced briefly in D1.5 deliverable of the IoT-i project (Haller 2012) Figure 12.1 gives a simplified view of the EPCglobal system architecture, taken from [EPCglobal]. It is worth noting that the tag itself is not represented as this figure ends up (at the bottom) with the air interface.

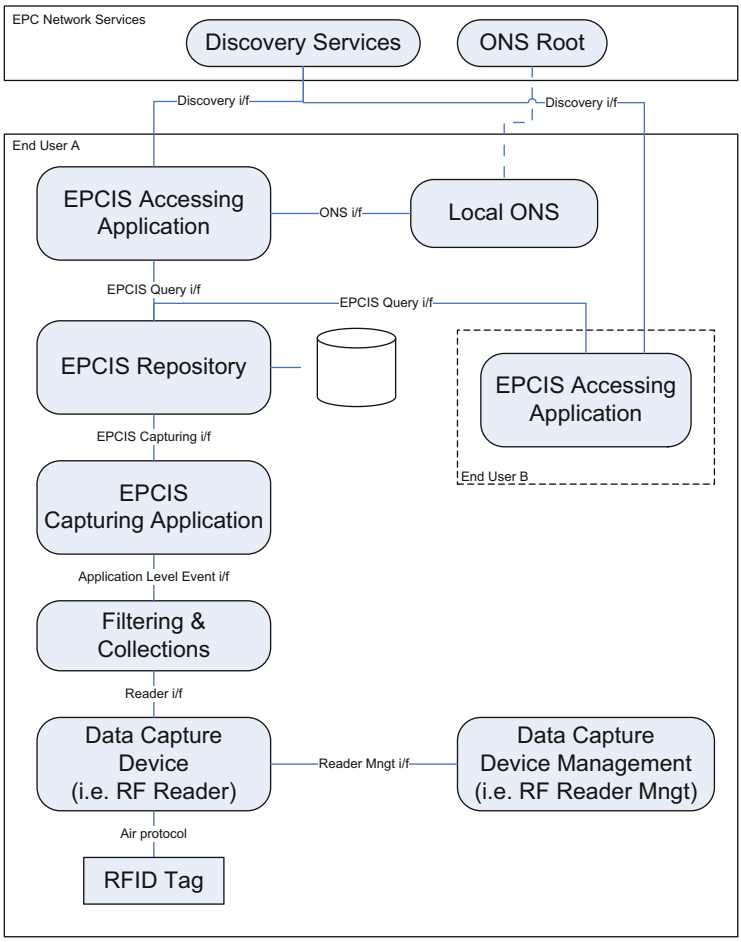

Fig. 12.1 EPCglobal system architecture (simplified)

12.3.1 Mapping to the Domain Model

In the EPCglobal architecture, the unique identifier associated with a physical object is the *Electronic Product Code* (EPC). It is defined by the EPCglobal Tag Data Standard, which defines its structure and encoding rules. Uniqueness of encoding structure (in order to avoid name collisions) is ensured by the use of a central Registration Authority.

The EPC Network Services in Fig. 12.1 are under the responsibility of the EPCglobal central authority and they are responsible for respectively providing discovery service to EPCglobal parties (end-users). The *Object Naming Service* (ONS) root management is also under the responsibility if the central authority

since it is the one allocating the EPC blocks. Local ONS are under the responsibility of the EPC manager (one per registered end-user).

After getting the address of an *Electronic Product Code Information Service* (EPCIS) responsible for the EPC of interest, an EPCIS Accessing Application will use the EPCIS query interface (i/f) to query additional information about an EPC (like class level/instance level or transactional data about a particular EPC). EPCIS query interface uses both push and pull mode, which means that it can be also used to receive notifications of observations concerning a particular EPC.

EPCIS Repository is the functional block, located at the "end-user A" side, deals with storage of information (of any nature) it wants to share with other parties (e.g. end-user B) of the EPC Global network. Of course all interfaces have to be implemented following the EPCglobal standards, however a certain level of freedom is left to "end-users" as for how those block shall be implemented.

The ONS block is a simple look-up Service that will map an EPC to the address of a designated EPCIS Service by which information about the EPC can be found.

The Filtering & Collection functional block is responsible for collecting raw tag data following policies defined by the EPCIS Capturing Application box. Example of such policy is: gathering all EPC of a certain class that have been read on a certain date, location and time interval.

The EPCIS Capturing Application supervises the operation at the lower level of the model and provides business context by coordinating with other components involved in a given business process. Again, a lot of freedom is left to the end-user for implementing this box, as far as the Application Level Event (ALE i/f) and capturing i/f are implemented according to the EPCglobal standards.

To finish up with the lower level, the Data Capture Device box (Tag Reader) is the one observing events relating to RFID Tags. The corresponding Reader i/f provides those events to the Filtering & Collection box.

The purpose of the reverse mapping is to check if the EPCglobal architecture is compatible with the IoT Reference Model.

The EPCglobal architecture illustrated in Fig. 12.1 is not exactly an EPCglobal domain model (as we understand IoT Domain Model in IoT-A), but rather a high-level diagram of a concrete architecture. Because the two models are not exactly similar in nature (i.e. IoT Domain Model is clearly at the "concept" level while the EPCglobal is a high level system architecture description) the reverse mapping of the EPCglobal architecture towards the IoT Domain Model is not a straightforward or simple process.

So in the following we use the EPCglobal system architecture in order to extract the EPCglobal concepts and then build an EPCglobal domain model taking a basis the generic IoT DM (meaning we try linking the EPCglobal concepts using the IoT

Table 12.3 Mapping EPCglobal concepts to the IoT Domain Model

EPCglobal concept	IoT ref. model concept	Comments
Entity	Physical entity	Is the Physical object been tracked by the EPCglobal system
End-user	User	The user managing and using the EPCIS, and reading the EPC
Partner user	User	The user willing to access EPC information for their own business
Physical entity	Physical entity (special case of)	Corresponds to physical objects like parcels, objects etc.. . .
Location	Physical entity (special case of)	Places, room, lift,. . .
RFID tag	Tag	The physical tag embedding the EPC
Tag reader	Device/Sensor	
Reader interface	Service	
EPC manager	Service	Is granted a portion of the naming space and assigns EPC to products
EPCIS accessing application	User	Located at end-user side that is willing to access EPC related information
EPCIS service	Service	Service that encompasses interfaces for data exchange (through the EPCIS Query Interface e.g.) and specification of Data (EPCIS data standard)
EPCIS query interface	Service	Interface exposed by the EPCIS and accessed by the EPCIS Accessing Application
EPCIS capture interface	Service	
EPCIS repository	Service/Resource	Exposes the EPCIS Query Interface. Stores info about EPCs events. . .The actual functionality of storing (e.g. in a data base) could/should be modelled as a Resource whereas the component that exposes the interface would be a Service. Of course that could be implemented tightly coupled as one software component
EPC record	Virtual entity	Consists of all info related to EPC (stored in EPC Data Base)
EPC data base	Network resource	
EPCIS capturing application	Service	Exposes the EPCIS capture interface
Filtering & collection	Service/Resource	Exposes the filtering and collection interface. Collects tag reads over time intervals constrained by events definition by the EPCIS Capturing Application. Filtering functionality may be modelled as a Resource, whereas exposing the interface as a Service

DM relationships, and try mapping the EPCglobal concepts to the IoT-A concepts) Nevertheless, we still executed to reverse mapping by building an EPCglobal domain model taking a basis the generic IoT DM (meaning we linked the

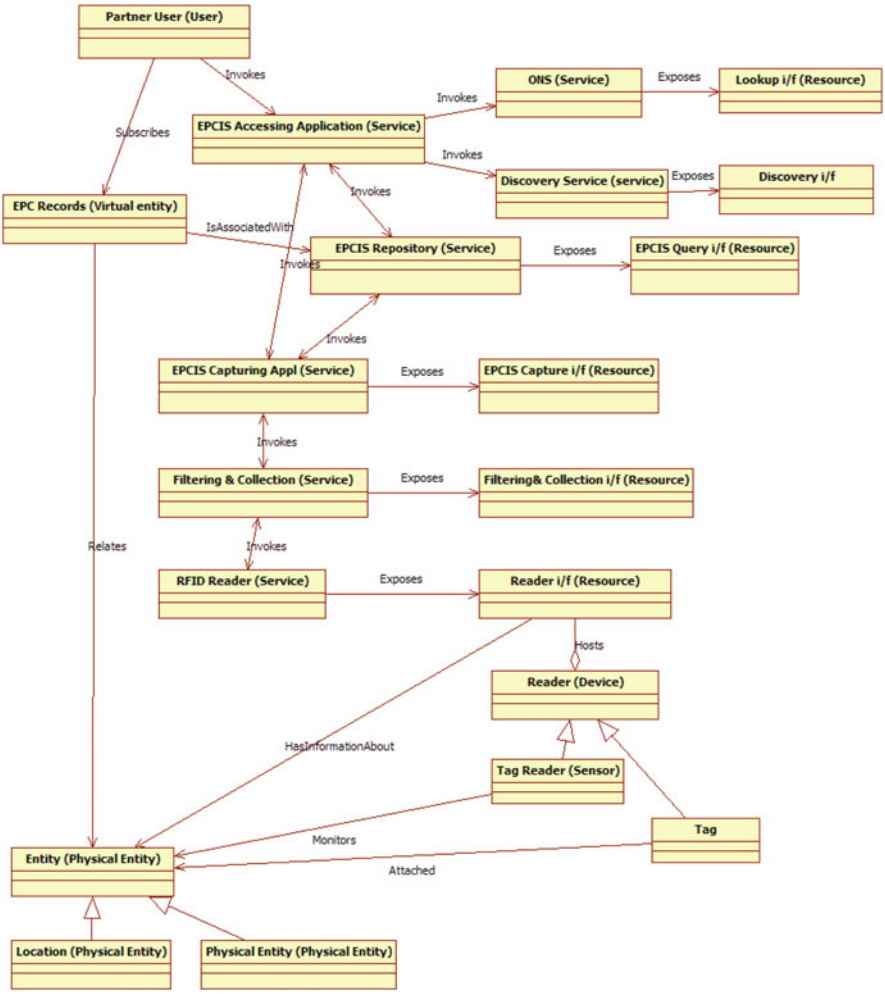

Fig. 12.2 EPCglobal domain model fit into the IoT Domain Model

EPCglobal concepts using the IoT DM relationships, and mapped the EPCglobal concepts to the IoT-A concepts).

First we identified a list of concepts that can be extracted from the EPCglobal system architecture and mapped them to the corresponding IoT DM concepts. This mapping is illustrated in Table 12.3.

Then according to the IoT Domain Model, the kind of concepts it handles and how those concepts are connected through relationships, the following (see Fig. 12.2) and consistent UML EPCglobal domain model could be extracted. As it fits the IoT Domain Model framework it can be argued that EPCglobal fits the IoT

Domain Model and that EPCglobal is truly an IoT system from the IoT-A definition point of view.

However, during this reverse mapping exercise IoT-A raised few comments:

1. Difficulties to model interfaces in general, as interfaces are not part of the IoT Domain Model in IoT-A. But it can also be argued that "interface" is purely a software concept which makes great sense in an architecture but making no sense at the concept level (i.e. in a domain model). Again this can be due to the fact that they (IoT-A) tried to fit somehow a system architecture into a domain model;
2. EPCglobal does not emphasise the need for Augmented Entities. They are therefore not part of the model;
3. Difficulties to model that a User can be responsible for managing a Tag (therefore End-user has not been included in the model);
4. There is a need for introducing end-users formally in the model with roles. It must be possible to express the fact that end-users with management role can associate information to a tag for instance.
5. It should be possible to express the fact that User can discover Services, that Services can discover Resources, that Resources can discover Resources (to be discussed which combinations make most sense);
6. Some links between IoT Domain Model and IoT Information Model should be explicitly described within the IoT Domain Model, like "*-description publishing"
7. Discovery and publishing are important concepts in IoT they should be very visible in the IoT Domain model as said in 7/ and 8/
8. We don't show here the reverse mapping to the IoT Information model, but it was pretty clear that the IoT Information Model is a meta-model that cannot really be used to model the class structure of the EPCglobal data handled at the different levels in the architecture (e.g. at Tag level, reader level, Filtering & Collection etc.. . .), in particular the IoT Information Model does not consider events (and EPCglobal is intensively using the notion of event). We reckon that most likely this is not the role of the IoT Information Model to model in a fine-grained way the class structure of a software system, especially when the class structure is clearly not IoT-specific.

12.3.2 Mapping to Information Model

As far as information is concerned, the main input in the EPCglobal reference architecture is the description of the EPC Information Service and the description of data the end user can share through the EPCIS interface.

EPCIS data within a so-called EPCIS record can be divided in several categories as follows[2] (see also Table 12.4):

[2] Excerpt from the EPCglobal Architecture document.

Table 12.4 Mapping of the EPCglobal information model to the IoT Information Model

EPCglobal concept	IoT ref. model concept	Comments
RFID tag	Device/Tag	Virtual entity representing RFID tag associated with the Physical Entity
EPC	Virtual entity	Electronic product code. It is encoded on the RFID tag
EPCIS event	Value	Might be just a wrapping of IPCIS data in the form of an event...
EPCIS data	Value	Is the data associated with the Physical Object and therefore contained in the EPCIS Virtual Entity
EPC record	Virtual entity	Consists of all info related to Physical Object identified by EPC (stored in EPCIS Data Base), i.e. IPCIS Data
EPCIS static data	Value	Contains class level Data and Instance level Data
EPCIS transactional data	Value	Relates to observations (instances, quantity within a class)

- **Static Data**: class level and instance level data, which do not change over time during the physical object life span
  - **Class Level Data**: there remain identical to any object which is an instance of that class
  - **Instance-Level Data**: the data may vary within objects instance of a class. Typical examples are lot number, expiry date, number within a lot, S/N etc.. . .
- **Transactional Data**: which changes and grows over the physical object life span, possibly created by more than one actor along a supply chain for instance:
  - **Instance observation**: it records events concerning the Physical Objects and often relates to dimensions like time, location, other EPC, and business process steps
  - **Quantity observation**: records events concerned with measuring the quantity of objects within a particular class. Five dimensions: time, location, object class, quantity, business step.
  - **Business Transaction Observation**: records association between one or more EPC and a business transaction. Four dimensions time, one or more EPCs, business process step, business transaction id.

12.3.3 Security Model

As explained in the EPCglobal Architecture Framework document [EPCglobal], the EPCglobal Architecture Framework allows for many different authentication technologies across the different interfaces. It is however recommended in the EPCglobal architecture document, that the X.509 certificate-based method should

be used by end-users when accessing the EPCIS interface for example. Typical case occurs when the EPCIS Accessing Application of an accessing end-user (referred as Partner user in the architecture framework) is willing to access the EPCIS service of the primary end-user (the one owning the EPCIS data for instance). If used the X.509 certificates are expected to comply with the X.509 Certificate Profile [Cert1.0] which provide minimum level of security.

At the network level some network standards within EPCglobal rely on *Transport Layer Security* (TSL), some others EPCglobal standards rely on HTTPS (HTTP over TLS) for the purpose of Data protection.

At higher level both EPCIS Capturing I/f and EPCIS Query i/f standards are allowing authentication of client's identity so that companies (owners of the data) can decide very precisely whether access to that data can be granted or denied. For the query interface, Applicability Statement 2 (AS2) is used for communication with external partners. This RFC (4130) specifies how to securely transport data over Internet and allows in particular for mutual authentication, data confidentiality and integrity and non-repudiation. Those security qualities are required in the ARM. AS2 uses x.509 certificate as defined above.

The high level interface (AuthX) used for Authentication in the ARM Security Model does authorise for the use of X.509 certificates.

12.4 Reverse Mapping Ucode

The *Ubiquitous ID* (uID) architecture is an architecture proposed by Prof. Sakamura (from the University of Tokyo) (Koshizuka, Sakamura 2010) to implement the concept of Ubiquitous Computing (ubicomp). Ubiquitous computing is a paradigm coined initially by Mark Weiser in the late 80s (see Weiser 1991). It touches many aspects of computing, like OS's, displays, intelligent user interfaces, wireless communication and networking. In the vision of ubicomp, the computer as we use to know it today, has mostly (if not totally) disappeared. It has become invisible and ambient. While IoT as such is not ubicomp (for instance intelligent user interfaces are not clearly part of IoT field) it can be argued that IoT offers means for implementing partly the ubicomp concept, spreading intelligence among objects of extremely different natures, enabling for cooperation between objects and humans and creating awareness about the surrounding (Context awareness) in a fully connected environment.

The intelligent features or Services implemented through this paradigm can be enabled only if information about the objects, places, Devices, etc. is available to those Services. We therefore talk about "intelligent" "smart" or more specifically "context-aware" Services. This only works if those objects, places, Devices of interest can be uniquely identified at any point in time. The uID architecture relies on an identification technique called ucode (ubiquitous code) which can be considered as the cornerstone of the uID architecture. The ucode model is a descriptive technique that establishes relationships between Physical and Virtual Entities through relationships between ucodes.

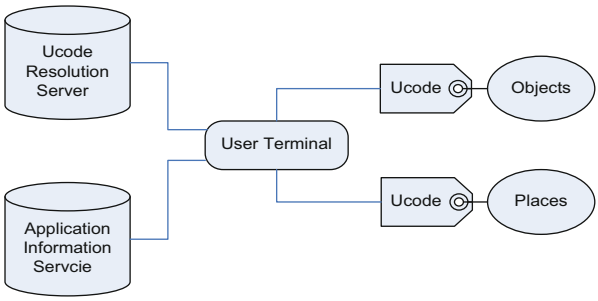

Fig. 12.3 Ubiquitous ID architecture

The basic principles of the uID architecture consist of uniquely identifying entities of interest with ucodes, maintaining databases that contain information about the entities, ensuring data and privacy protection and opening this platform through open APIs.

In order to enable those principles fundamental technologies and mechanisms such as ucode structure, ucode tag, ucode readers and terminals, ucode relational databases managing the entities information and ucode information servers are used. These different components are detailed in the following subsections. The simplified architecture shown in Fig. 12.3 is taken from (Koshizuka, Sakamura 2010).

12.4.1 uCode Model

In the ucode model [UID Architecture], unique identifiers are assigned to:

- **Objects**: tangible objects of the real world (industrial product, piece of art, everyday objects,..) as well as intangible ones like pieces of digital media or source code;
- **Spaces**: monuments, streets, etc.
- **Concepts**: relationships between objects and spaces of the real world, which are also named "entities". Those relationships are used to define complex context information, and are defined using a description framework called ucode Relation (ucR) model. Simple context information relates to objects and places directly.

It is worth noting that the uniquely assigned code does not contain any information about the entity. Relevant information about the tagged entity is stored in an application Information Service which can be located by resolving the ucode. A distinction is made between *physical* ucode which are by definition stored in a Tag attached to the entity, and logical ucode which are not stored in any Tag and are mainly used for identifying intangible objects as described above (including relationships between ucodes).

The main idea behind allocating ucode to relationships between entities comes from the Resource Description Format used to model knowledge. RDF knowledge

is made of triple <**subject**, **relation**, **object**> where each constituent of the triple is made of a URI. In the case of ucode relationship each ucR unit is a triple of ucodes. Information associated with the two entities and the relation can therefore be found querying the ucode resolution server.

In addition resulting from this establishment of relations between entities, are graphs (ucR graph) where single "subject" ucode gets linked to many "object" ucode via various "relations". Objects which are not ucodes are called "atoms". A subject ucode pointing via a relation towards a URL is a typical example of such rules involving atoms.

12.4.2 ucode Resolution Server

The resolution of ucode is achieved in the ucode Resolution Server The simplified resolution consists of taking the ucode read by the reader, searching for ucR units that correspond to that ucode and returning to the mobile terminal the addresses of content associated with the ucode via the relational database introduced earlier (similar to a triple store).

The Ubiquitous ID architecture can be simply described as follows (Fig. 12.3):

From the descriptions of the various entities of the architecture (see Fig. 12.3) above, the following table of concepts could be derived (Table 12.5):

In turn, the reverse mapping produced the following UML (see Fig. 12.4) below:

The uID architecture uses the uCR to describe complex context information via relationships between real-world entities (Koshizuka, Sakamura 2010). So-called uCR units consist of a triple of ucodes: subject ucode, relation ucode, and object ucode. The object ucode can be replaced by simple literals, hence it becomes possible to express attributes of a real-world entity as a uCR unit, e.g., <ucode X, "hasBrandName", "GoldenTea">.

It is not feasible to try to map the uCR model directly to IoT Information Model. The IoT Information Model provides a vocabulary for describing IoT systems and it does not, explicitly prescribe how information should be represented. The uCR, on the other hand, can be used to represent relations between any kinds of objects identified with ucodes much in the same way as RDF is used to represent resources identified with URIs. Therefore, the relation between IoT-A information model and the uCR model is actually complementary by nature and the uCR should be seen as a an alternative way (for XML, RDF, binary etc.) to represent IoT Information Model concepts.

12.4.3 Conclusion

To conclude, when mapped to the generic IoT-A the uID provides implementations for only a small subset of the functionalities defined in IoT ARM. First, the ucode provides a globally unique way identify physical (and virtual) objects. These

Table 12.5 Mapping of uID concepts to the IoT reference model

uID concept	IoT reference model concept	Comments
Tangible object	Physical entity	
Intangible object	Digital artefact	If the intangible object is a representation of a tangible object, then it is also a Virtual Entity
Location	Physical entity	Location is not modelled explicitly in the IoT Domain Model. However, a specific (possibly tagged) place can be regarded as a Physical Entity
uCR model	Relates to information model	uCR can be used for representing IoT-A Information Model instances
ucode	No direct relation	The ucode can be used as an globally unique identifier for any instance of the IoT-A RM concept
ucode resolution gateway	Network-based resource	Provides a ucode resolution over HTTP
ucode signature server	Network-based resource	Prevents ucode counterfeiting by verifying and generating signatures
ucode management server	Network-based resource	Manages the allocated ucode space
ucode issue gateway	Network-based resource	Provides a HTTP interface for obtaining ucode issued by ucode management server
ucode entry update gateway	Network-based resource	Provides a HTTP interface for updating ucode resolution entry
Top level domain server	Network-based resource	Hierarchical component of the ucode resolution server
Second level domain server	Network-based resource	Hierarchical component of the ucode resolution server
ucode resolution server	Network-based resource / Service	The resource would be exposed through a resolution service
Application information service	Service	Provides infrastructure and application services
ucode tag	Tag	
User terminal	(Device)	Is a device that reads ucodes and provides services based on the ucode to a user. A user terminal that is just used to run an application or display some information is not in the scope of the IoT Domain Model. However, a user terminal containing a reader is in the terms of the IoT Domain Model a Device with an embedded Sensor Device
Reader	Device/Sensor	

ucodes can be used as identifiers for any instance of the IoT ARM concept. Second, the uID provides a way to resolve the address of the information service hosting data about the object identified with a ucode. This functionality is basically a subset of the functionality defined for the IoT-A resolution infrastructure. Third, the uID provides methods (i.e. the ucode Relational Model) for representing relations

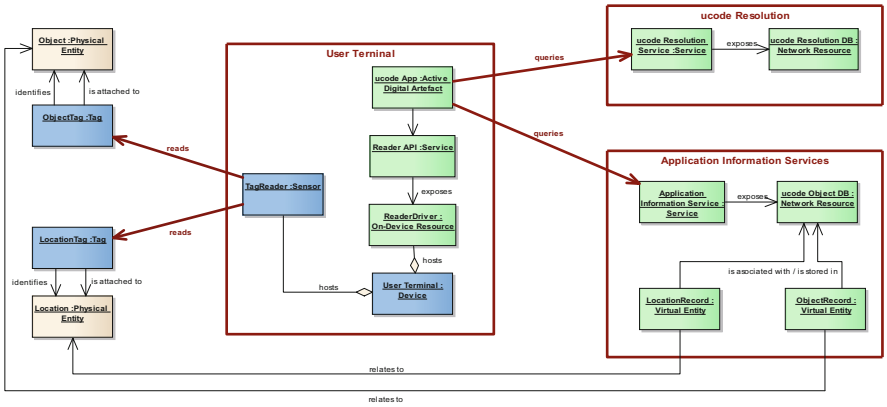

Fig. 12.4 uID architecture fit into the IoT Domain Model

between ucodes. This functionality can be used for representing IoT Information Model concepts.

12.5 Reverse Mapping BUTLER Information Model

12.5.1 Introduction

BUTLER's mission is to provide context-aware services within an IoT environment[3]. What is really striking about that is that there does not seem to be an explicit and widely accepted definition of what context-awareness really means. Although an intuitive definition of what context means can be found easily ("the conditions and circumstances that are relevant to an event, fact, etc.") (Dey 2001), a more formal definition is needed. If we look at other projects within the FP7 umbrella, we can see what FI-WARE[4] provides – a data/context management section. Although not straightforward, it states that "Context [..] is represented through context elements" and that these elements "are typically created containing the value of attributes characterising a given entity at a given moment". Therefore, we can state that the context "characterises a given entity at a given moment". That definition gives rise to a discussion of what an entity is and the extent to which characterisation must take place.

[3] http://www.iot-butler.eu/

[4] FI-WARE (http://www.fi-ware.eu/) is a European FP7 Research Project aiming to foster the emerging Future Internet by creating an open architecture and a reference implementation of a novel service infrastructure, building upon generic and reusable building blocks developed in earlier research projects.

12.5.2 Reverse Mapping of IoT Domain Model

With regard to the first issue, the IoT-A Domain Model (IoT-A DM) fits perfectly within such a definition, as it introduces the concept of Virtual Entity. These Virtual Entities are the main concept handled by the IoT-A model since they represent the entities in the real world that designers of IoT applications consider relevant. The remaining main concepts introduced by the IoT-A DM (Resources and Services) are naturally associated to the Virtual Entities. A straightforward conclusion is that context should also be "associated" to Virtual Entities. However, the IoT-A DM does not explicitly consider the context. This is irrelevant; we will return to this issue later.

The second issue (what the context means for a given entity) is actually related to the deployment and implementation of a given IoT scenario and not to the definition of the model. However, it is worth mentioning that the components of the context (the "context elements" mentioned by FI-WARE) are totally dependent on the needs and requirements of the consumers of the functionality exposed by entities (again, in a specific scenario). In BUTLER, we introduce a model in which, given a Virtual Entity, it is possible for the consumer of the functionality (usually an application developer) to define at any time the relevant context for this entity. This type of context declaration operation defines the context elements and the data sources these context elements will depend on. It is also important to acknowledge that a given entity context relies not only on the information that devices can gather about it but also (and sometimes mainly) on dynamic data sources that are not actually "device-originated".

BUTLER has taken the IoT-A DM as its main inspiration. However, it has been simplified somewhat to increase the readability and clarity of the model. For instance, the Augmented Entity is not considered; Digital Artefacts – and therefore a non-Human User – have been removed as well; Network Resources have been also dismissed and therefore, the BUTLER model contemplates only On-Device Resources. UML has been used to illustrate the model graphically similarly to the way the IoT-A DM does. We will highlight the main differences and additions we have considered.

The relationship between Users, Physical and Virtual Entities is almost identical to the ones suggested by the IoT-A DM. Besides the simplification already mentioned, it is worth noting that the BUTLER Domain Model introduces additional relationships that are not expressed by the IoT-A DM. For instance, there can be additional relationships between *Users* and *Physical Entities*. The most obvious is the "ownership" (or at least entitlement to the management of the Physical Entity). The relevance of this relationship relies on the access permissions it derives (that is, the owner of a house will have the "right" to get information about his home and to adjust the desired temperature, while a stranger will not, at least not until the owner gives him the right to). On the other hand, the BUTLER Domain Model introduces the BUTLER terminology and therefore, instead of talking about Devices, we

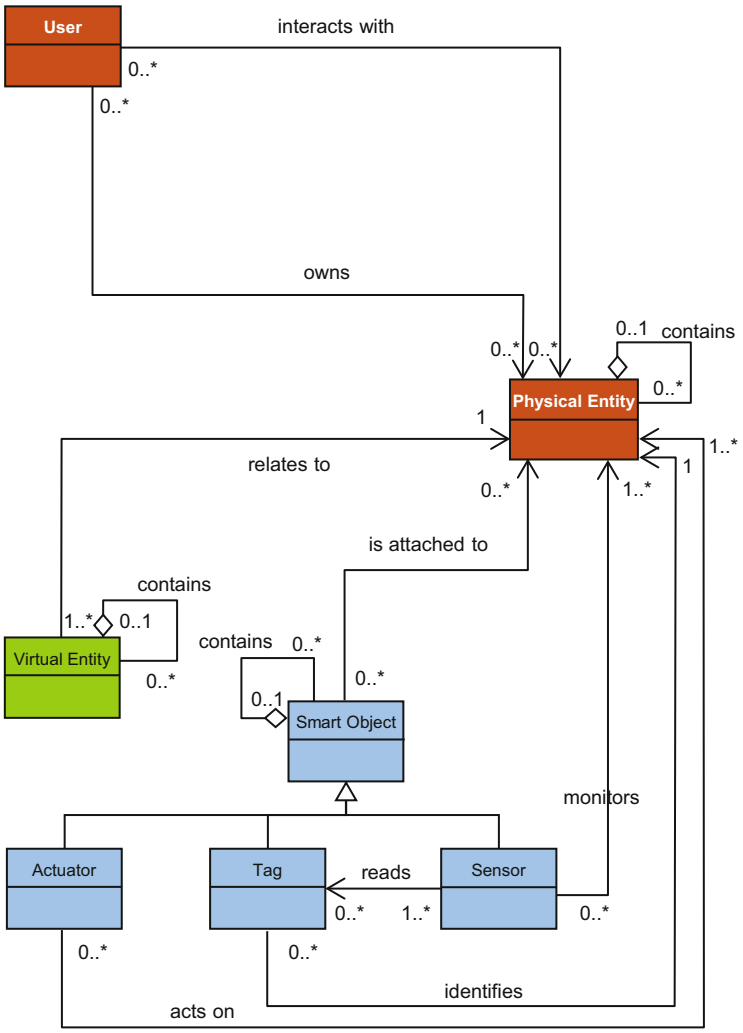

Fig. 12.5 Relationships between users, physical and virtual entities

introduce the Smart Object concept, which is equivalent to the Device concept within the IoT-A DM (Fig. 12.5).

As with the IoT-A DM, Resources are introduced to bridge the gap between the *Virtual Entities* and the *Smart Objects*, enabling the monitoring and manipulation of *Physical Entities* from the digital world. *Resources* are the software components that actually provide information about, or enable the actuation on *Physical Entities*. BUTLER simplifies the management of Resources, focusing on *On-Device Resources* (those deployed locally on the *Smart Object* attached to the *Physical Entity*; these types of Resources are typically sensor *Resources* that

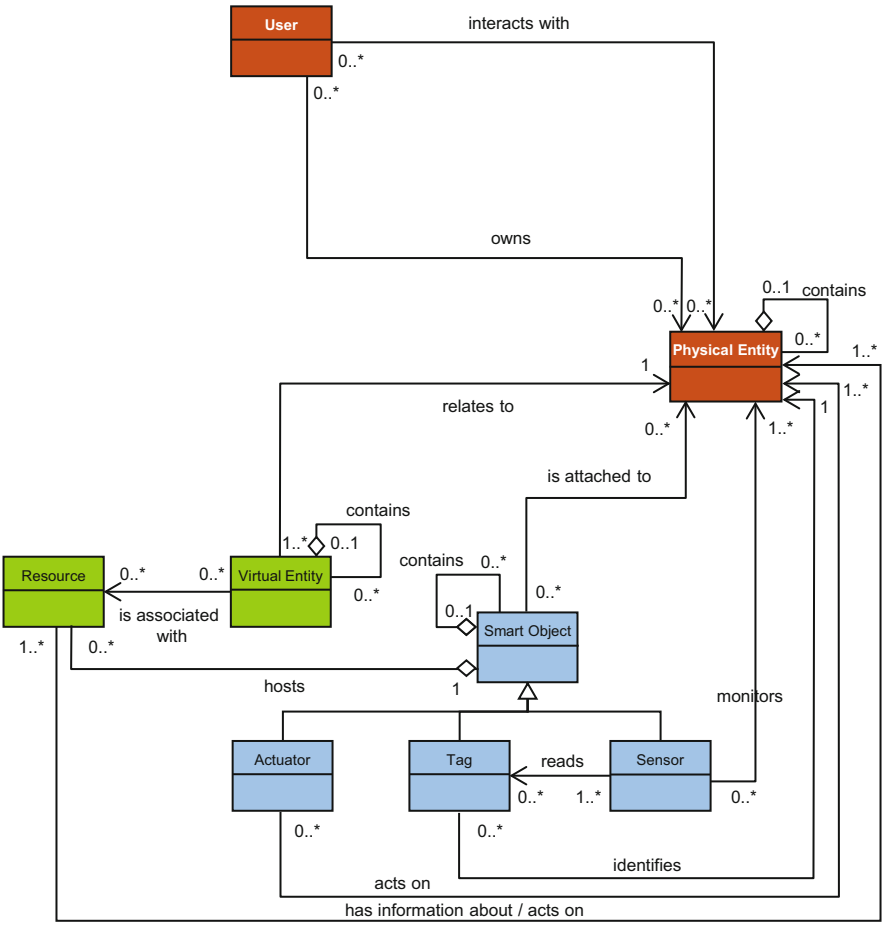

Fig. 12.6 Introduction of the resource concept and its relationship to devices and physical entities

provide sensing data, or actuator *Resources*). It is possible to model other Resources deployed externally to *Smart Objects* that run somewhere in the network as generic *Network Resources*. These Resources can process data, for example, taking sensor information as input and generating aggregated or higher-level information as output (for instance, a dynamic data source providing dynamic weather forecasts or energy consumption estimates). Also, *Network Resources* can be storage *Resources* storing information coming from *On-Device Resources* and thus provide information about *Physical Entities* (i.e. location and state-tracking information (history), static data, such as product type information, and many other properties). Other external data sources, even Human Users, can also update the information in a storage *Resource* (Fig. 12.6).

The primary relationship between *Physical Entities* and their digital counterparts, the *Virtual Entities* on one hand, and the *Smart Objects* and the

Resources they host on the other, is achieved by means of associations. Therefore, *Users* are enabled to act on or to know about *Physical Entities* by means of the associations between *Virtual Entities* and *Resources*. For each *Virtual Entity* there can be associations with different *Resources* that may provide different functionalities, such as retrieving information or enabling the execution of actuation tasks on the Virtual Entities. When a *User* wishes to acquire information about or to actuate on a given *Physical Entity*, she would perform a discovery process determining which *Resources* associated to the *Virtual Entities* representing the *Physical Entities* enable actuation or data access. Next, the User would pick up the *Resources* that match her requirements and invoke them. However, it is unlikely that the User would directly invoke *Resources*. She would do it instead through a *Service* or application that accesses Resources to perform its business logic.

Finally, it is necessary to acknowledge that both *Smart Objects* and *Users* can be modelled as a *Physical Entity*. The same may happen with *Smart Mobiles* (the client device used by users in the BUTLER terminology) (Fig. 12.7).

Here we can see a main divergence from the IoT-A DM, since that model introduces an explicit relationship between Services and Virtual Entities. Although the nature of the relationship is not explicit in the model, the IoT-A Information Model offers additional information about what such a relationship looks like: the Virtual Entity attributes are used to associate Services to Virtual Entities. We prefer a model in which the context is made explicit in the BUTLER Domain Model (and not disguised as the Virtual Entity attributes).

As described in the initial section of this chapter when describing the FI-WARE data/context, the context elements are associated to the entities the system handles. BUTLER proposes to associate *Contexts* to *Virtual Entities*. Therefore, it will be possible to handle the context of the *Physical Entities* represented by the *Virtual Entities* the *Context* is associated to. On the other hand, several *Contexts* can be associated to a given *Virtual Entity* just to reflect the fact that different "consumers" will have a different need or view of the context associated to a given entity (Fig. 12.8).

On the other hand, the attributes of *Contexts* in BUTLER will be mostly created from data obtained from *Resources*. Each attribute will be the result of an operation executed over data elements from one or several Resources. Such Resources may or may not be associated to the *Virtual Entity* the *Context* is associated to (Fig. 12.9).[5]

Finally, Services will be entitled to use not only low-level *Resources* when they need to know about the status of *Physical Entities*, but also richer *Contexts* (Fig. 12.10).

[5] For instance, the context associated to a house can include the outdoor temperature. This temperature value can be exposed through a Resource associated to the Weather Service, which in turn has also been modeled as a Virtual Entity. Although the Resource exposing the temperature is not associated to the Virtual Entity representing the given house, an element of its context relies on this "external" Resource.

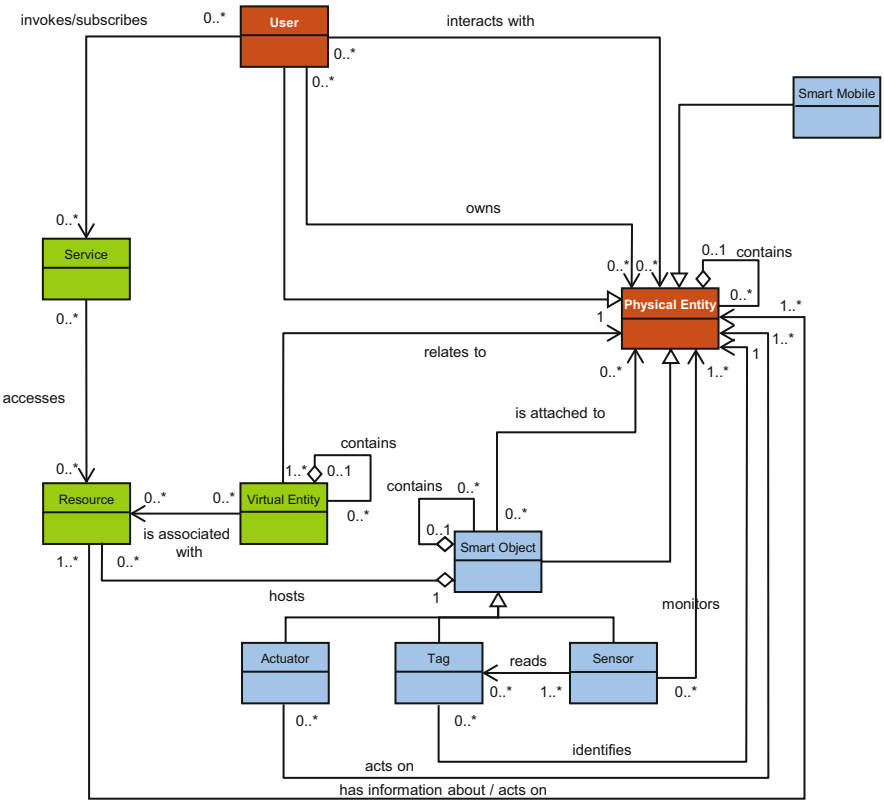

Fig. 12.7 Users to service/resource relationship

12.6 Reverse Mapping MUNICH Platform

The goal of reverse mapping an existing system towards the IoT Reference Model is to show that an existing system that has been designed without applying the IoT ARM can be redesigned according to the IoT ARM. By doing so the IoT ARM shows its potential for being a reference model for any kind of IoT systems.

12.6.1 Use Case Description

The use case is about counting "stomach towels" which are used inside the abdomen during surgery of a human. After the operation it needs to be assured that no towels are retained in the abdominal cavity of the patient's body. Therefore, each towel is fitted with a 13.56 MHz RFID tag which enables tracing the towels

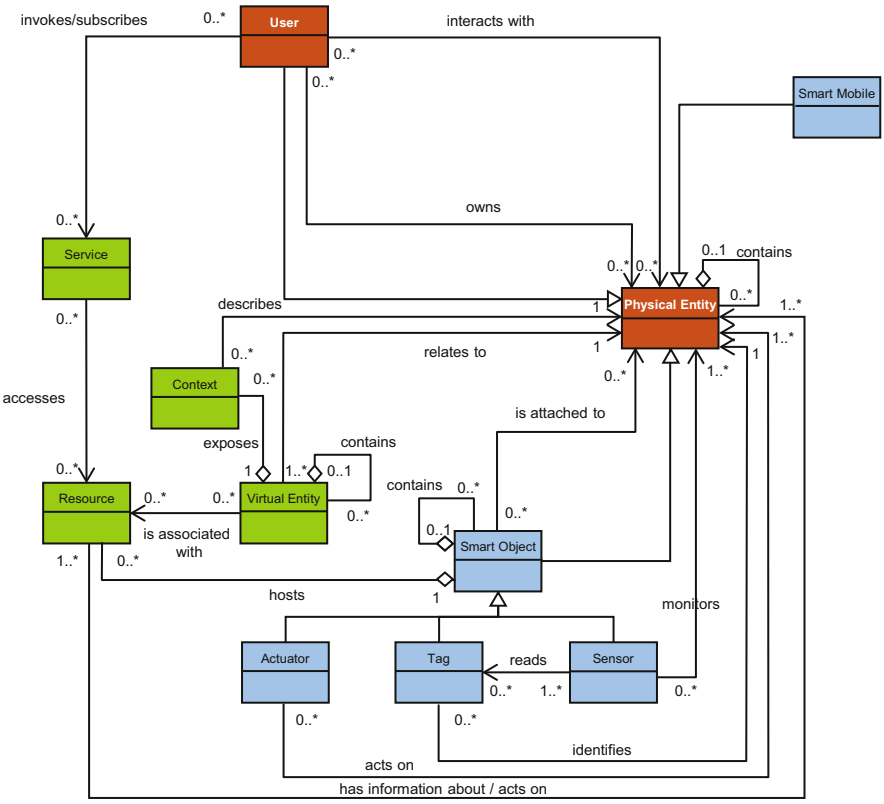

Fig. 12.8 Context to virtual entity relationship

before, during, and after the surgery. The RFID-tagged towels may be tracked by three antennas from different positions in the operating theatre:

- Mayo stand (instrument table): towel is unused;
- Operation table: towel is in use;
- Used towel container: towel is used

Each towel will be used in a specific order: First a batch of "unused" stomach towels resides on the instrument table. Towels that are put into the abdominal cavity are declared as "in use". Finally, towels that are not needed anymore after the surgery are put into the towel container where their status is set to "used".

Every time an RFID reader recognises a tagged towel appearing or disappearing in its range an event is generated and stored in an event-log database hosted in the cloud.

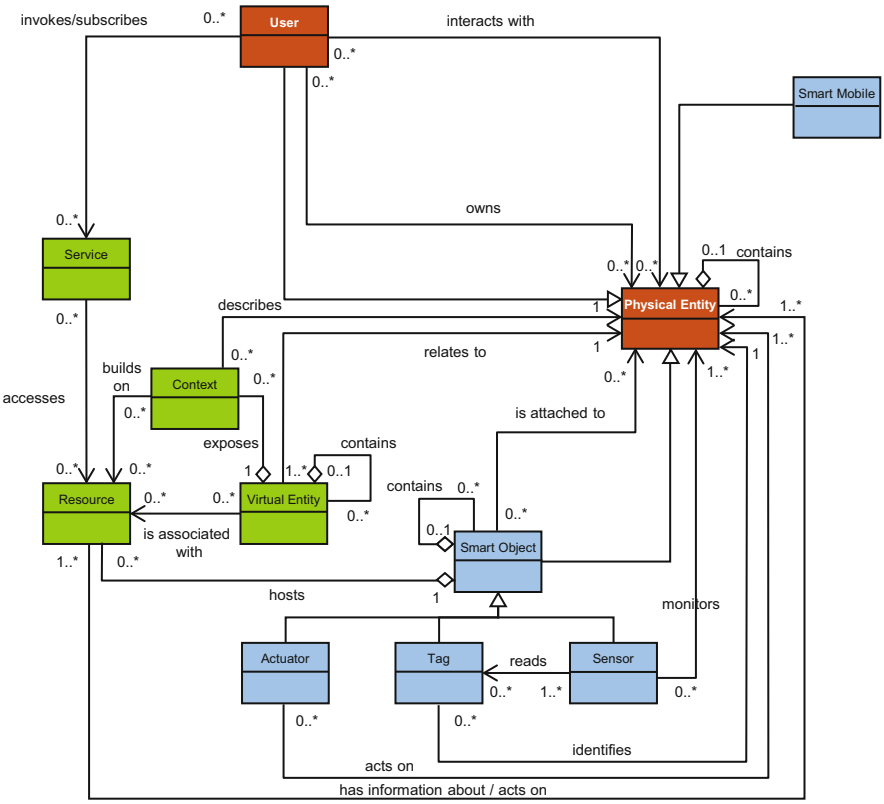

Fig. 12.9 Context to resource relationship

12.6.2 Use Case Objective

It must be assured that no towels are left inside the patient's abdomen when the operation has finished. In more technical terms it means that after finishing the operation all the towels that were "in use" must be in state "used" meaning in the used towel container.

12.6.3 Current System Architecture

So far the use case has been designed to run with a certain type of RFID-readers only that are connected via USB-cable to a laptop computer that is hosting the application. The MUNICH-platform (MUNICH 2010) depicted in Fig. 12.11 provides a cloud storage system indicated as 'Open Nebula Core' that stores the events captured every time the 'Object Inventory Service' notice a change in the

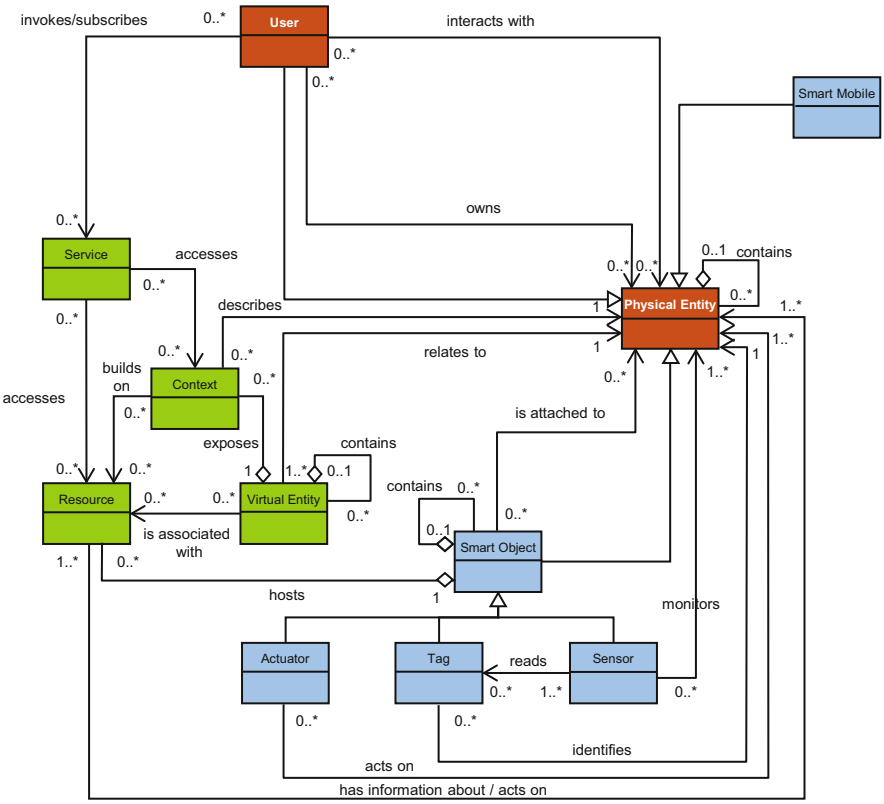

Fig. 12.10 Service to context relationship

number of towels in their respective range by invoking the 'Event Service'. The application that monitors the status of the towels during the operation invokes methods provided by the 'Operation Theatre Service'. The API to store and retrieve information from and to the cloud storage system is technology-specific. If an architect decides at a later point in time to change from Open Nebula to another technology the system needs to be adapted to the changes in the API.

12.6.4 Enhancement by Using IoT Reference Architectural Model

Making the use case demonstrator IoT-A conform means making the system more evolvable and future-proof. By using RFID reader services a technology agnostic layer is introduced that is not so much dependent on today's lifecycle of technologies. With the current solution the software needs to be updated when a

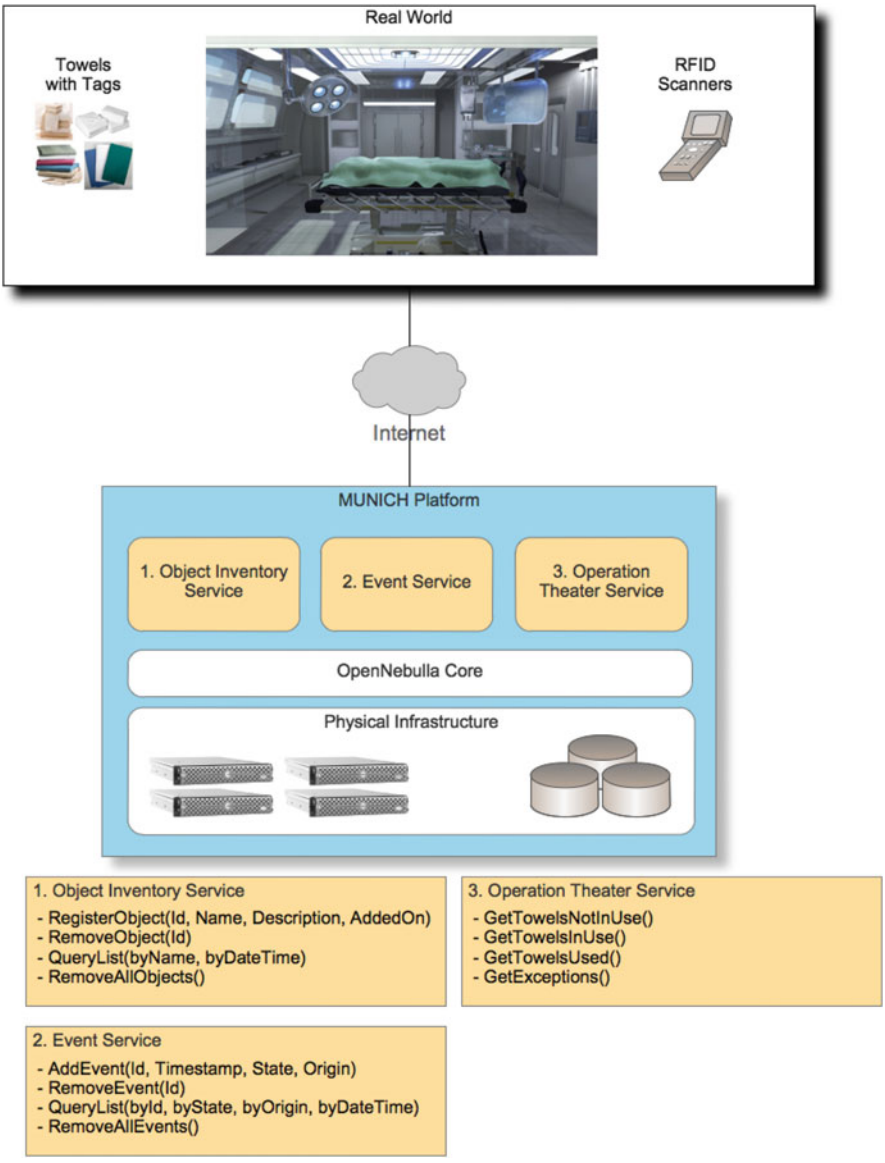

Fig. 12.11 Current architecture of MUNICH platform (MUNICH 2010)

new type of RFID reader needs to replace a current one. Also extending the use case with another RFID reader or another type of sensor will be much easier once IoT-A is applied. Thus the IoT ARM contributes towards scalability in this use case too. The restriction in evolvability applies to the cloud storage component too since the current system is designed to be used with certain cloud storage software. It is not

easy to substitute the component in case the software is discontinued or no longer appropriate. In case the services are modelled according to technology agnostic IoT-A specifications the system will be more future proof. In order to make the use case IoT-A-compliant, the following architectural process will be undertaken.

1. Specification of Business Process Model;
2. Specification of Domain Model;
3. Specification of Information Model;
4. Specification of Functional View;
5. Specification of Services and Interactions between components.

12.6.5 Specification of IoT Business Process Model

The use case has been formalised as IoT Business Process Model by a domain expert in Fig. 12.12. The modelling notation used is described in (Meyer et al. 2011). The operation scenario is a sub-process of the overall Emergency operation process that may include the arrival of the patient via ambulance and the availability of data record for the patient in the hospital's data base. The towels being used during the surgery are associated to the patient identified in the database record. This way it is possible to verify which towels have been used for which patient. The towels are the entities of interest (depicted by the box with the cow icon) in this scenario. The RFID reading processes are running in parallel on all three positions in the operating theatre that are equipped with the RFID readers. The used towel container is denoted as waste bin in Fig. 12.12. Each RFID reader sub-process sends events to the Event History database upon detection of tagged towels. The 'Monitor towel process' analyses the events that have arrived in the database, determines the current state for each towel, and calculates the number of towels that are currently inside the body of the patient.

12.6.6 Specification of IoT Domain Model

Based on the Business Process Model presented before, a domain model can be derived that identifies the Physical and Virtual Entities, the IoT Services, the Devices, Resources, and the users that are involved in the use case. The Human User is the doctor or other medical staff who is responsible to monitor the towels in the operation theatre. The actual monitoring of the towels by comparing the used towels with the ones currently in use is done by software implementing the 'Monitor towel process' as depicted in Fig. 12.12. The User checks only that no towels are still in use when the operation is about to end. The software 'Operation Theatre Application' is modelled as Active Digital Artefact. Each towel is a Physical Entity that has one RFID tag attached so that the number of towels corresponds to the number of tags. Each physical towel has a digital counterpart

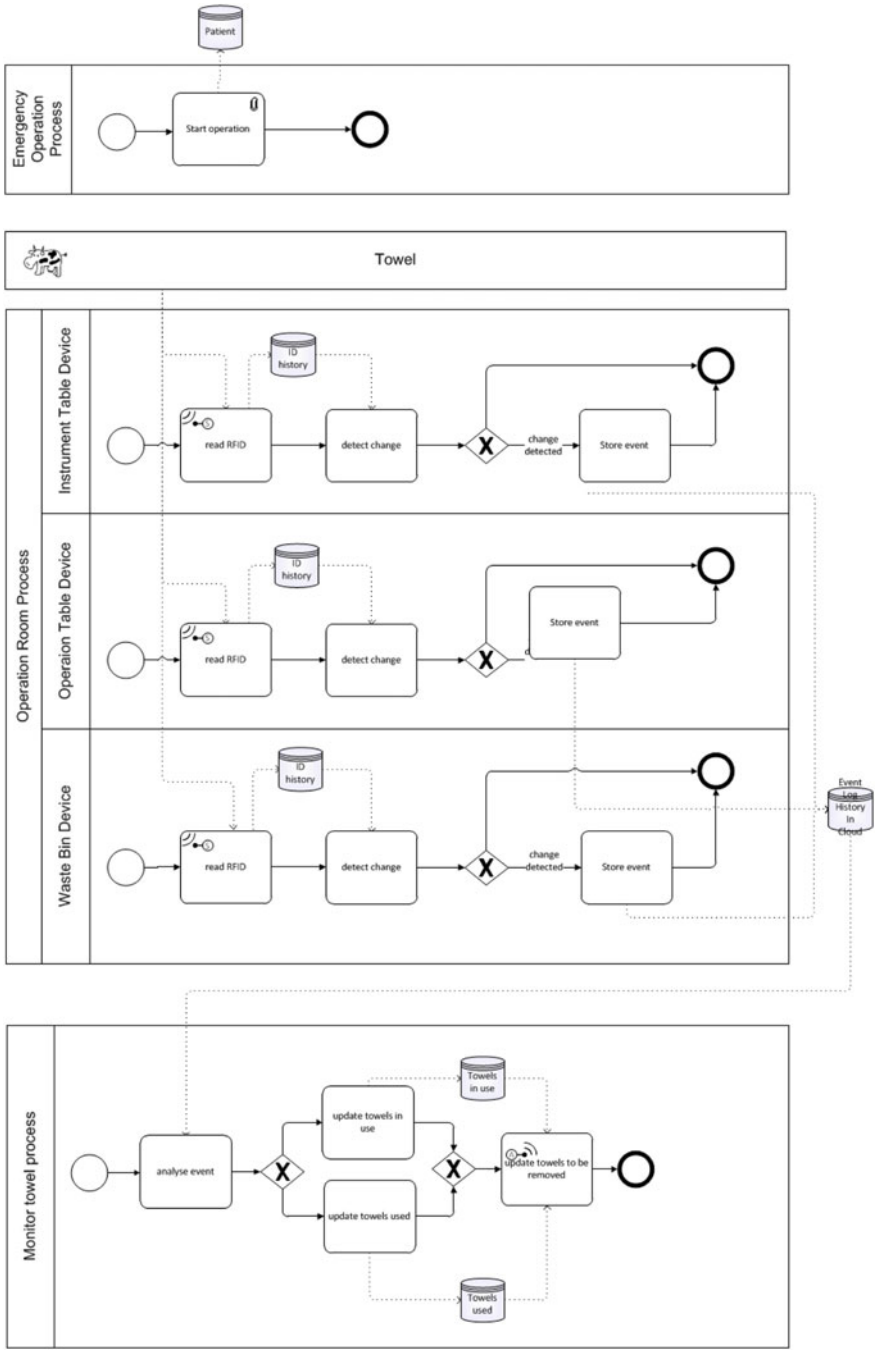

Fig. 12.12 IoT business process model of MUNICH use case

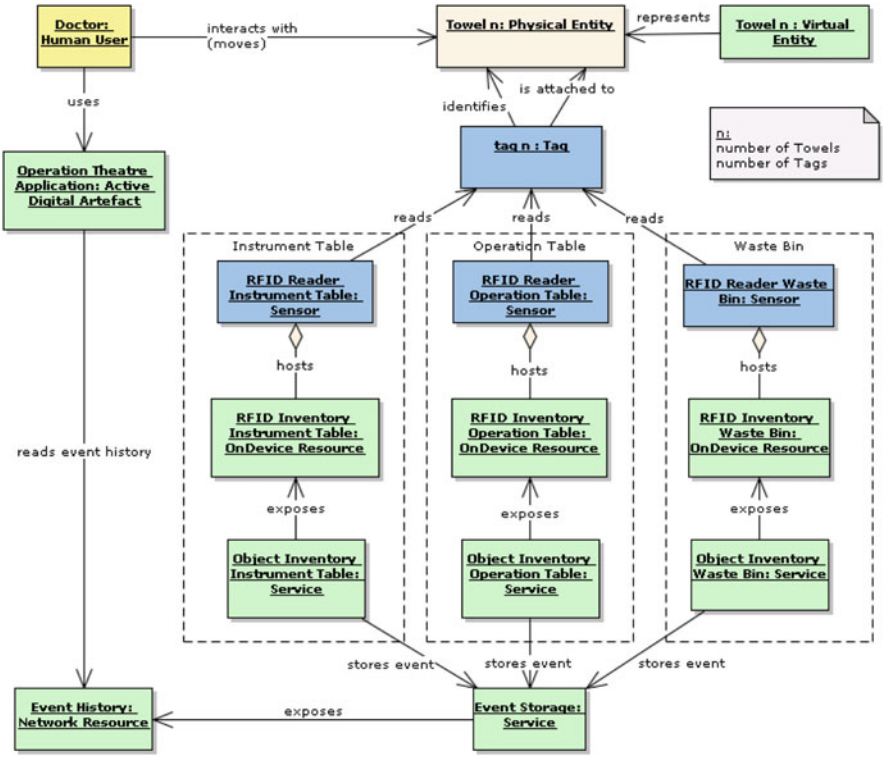

Fig. 12.13 Domain model of MUNICH platform

modelled as Virtual Entity. There are three RFID readers deployed in the scenario at different significant locations of the operation theatre (Instrument Table, Operation Table, and Waste Bin) that are modelled as Sensor Devices. Each of the Sensors hosts an OnDevice Resource that is exposed by an 'Object Inventory Service' as depicted in Fig. 12.11. These services store events by invoking the 'Event Storage Service' that exposes the Network Resource 'Event History'. This Resource is also exposed to the 'Operation Theatre Application' by the Event History (Fig. 12.13).

12.6.7 Specification of Functional View

The realisation of the use case according to the IoT ARM a Functional View is tailored to the use case needs to be specified. The Functional View for the MUNICH platform is depicted in Fig. 12.14. No IoT Service Resolution is required, because all needed services are already known to the system at design time. A VE Resolution FC is included in the FV. This FC is able to resolve particular towels to the IoT

Service they are currently associated with. The 'VE & IoT Service Monitoring' FC is used to update the current state of towels whenever these VEs change their position in the operating theatre. Whenever VEs change their positions their associations between the VEs and the IoT Services reading the RFID tags change too. No Service Organisation functions are required in this use case since the binding of services is static and can therefore be hardwired. To accommodate IoT Business Process Management functionality that is required in the MUNICH platform the respective FG is included in the FV. The process model diagram depicted in Fig. 12.12 was created by the 'Process Modelling' FC and this model is executed by the 'Process Execution' FC. The Functional View of the MUNICH platform includes IoT Services for the RFID readers and for Event Storage Resources. The Application in the FV is the use case as described the beginning of this Section. The Devices are the RFID readers and Tags used in the operational theatre which communicate to the IoT Services by 'End To End Communication' and 'Network Communication' FCs. The entire FV is depicted in Fig. 12.14.

12.6.8 Specification of IoT Information Model

The IoT Information Model specified for this use case also addresses relationships between entities that are not depicted in the IoT Domain Model before. Some more entities appear in the IoT Business Process Model shown before in Fig. 12.12. For instance it is depicted that an 'Operation' is held for a 'Patient' and thus the 'PatientIdentifier' (valid in the clinic) is assigned to an 'Operation'. Operations are processes with a defined status at any point in time: 'before', 'in', and 'after Operation'. There is also an unknown status in case the status cannot be obtained. The towels are represented as VEs with domain attributes that are essential for the use case. The towel's identifier stored into a RFID tag is one of the attributes as well as the current state of a towel that can be one of 'unused', 'in use', and 'used'. Again there is an 'unknown' state specified in case the state cannot be obtained by the system. The aforementioned designated locations of the operating theatre are reflected in the Information Model as attributes of the VE 'Towel'. For simplification the allowed values for this attribute {InstrumentTable; OperationTable; WasteBin; unknown} are not visualised as ValueContainer. With the aforementioned attribute values the OperationTheatreApplication is able to relate the current location of the towels (retrieved through the RFID readers) to the respective state of the towel: {instrument table = 'unused'; operation table = 'in use'; waste bin = 'used'} (Fig. 12.15).

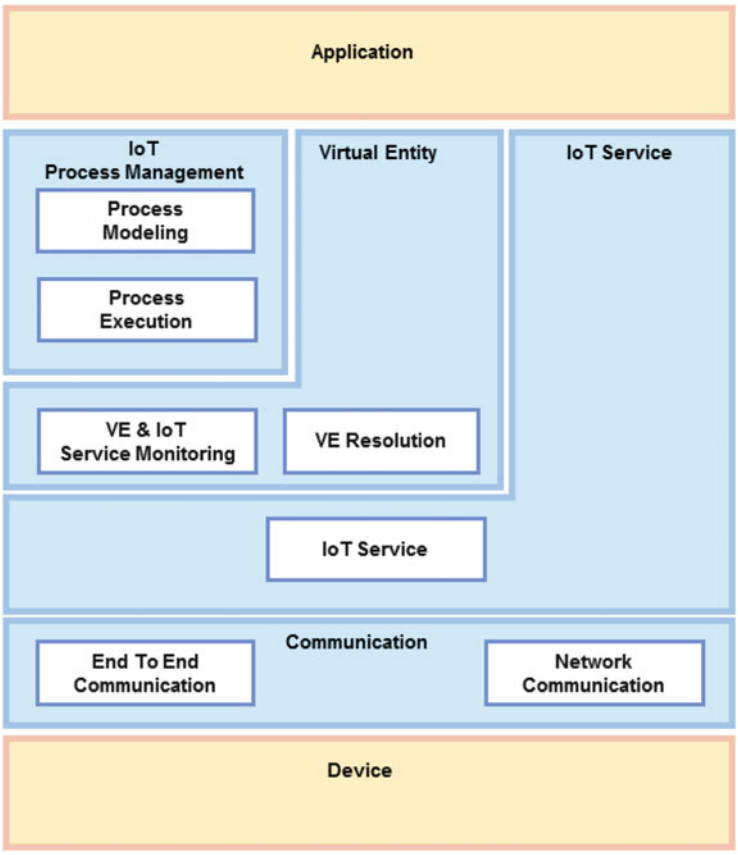

Fig. 12.14 Functional view of the MUNICH platform

12.6.9 Specification of IoT Services and Interactions

In the following an example description is given for one of the three 'Object Inventory Services' specified in the IoT Domain Model before (Fig. 12.16).

The sensing service 'ObjectInventoryServiceOPtable' exposes the Resource RFIDInventoryOperationTable hosted on the Sensor that observes the area OperationTable during the operation. The duration is determined by the Op123Schedule. The output of the service is described in a domain specific operation-ontology by the class ListOfRFID that defines a list of identifiers the RFID reader has detected. The service can be invoked by accessing the service endpoint objInventoryOPtableRestSE that provides a RESTful web service on the endpoint host optablehost. An HTTP GET method call on port 4355 on the root path '/' of this host will return the list of identifiers the RFID sensor has read.

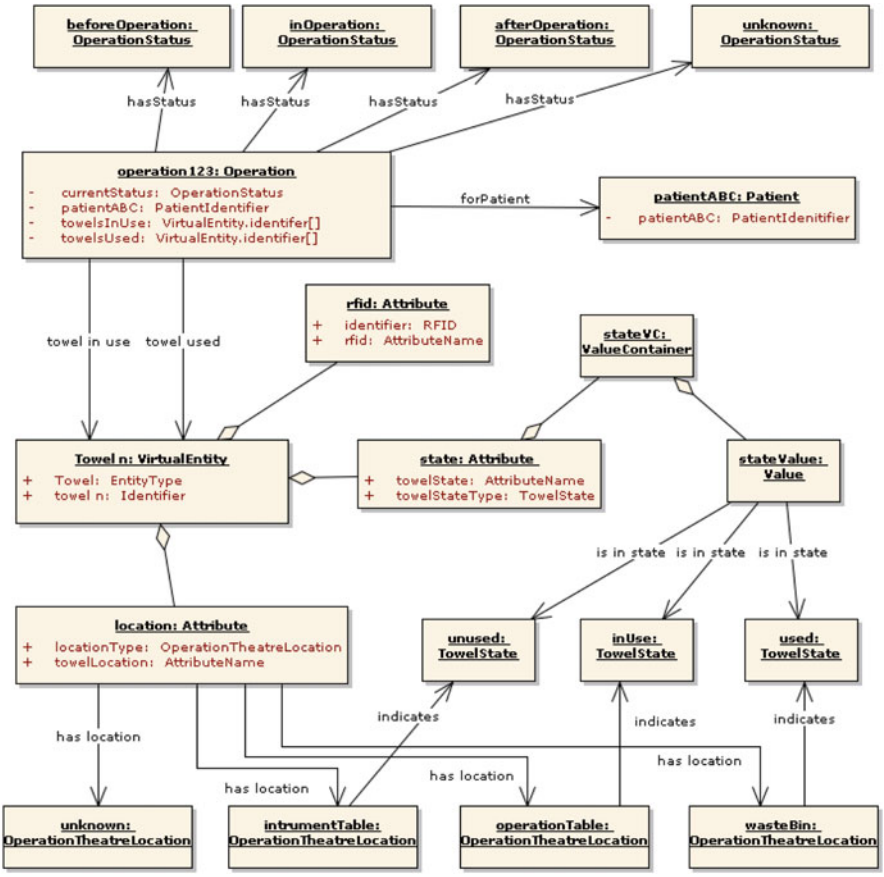

Fig. 12.15 Information model of MUNICH platform

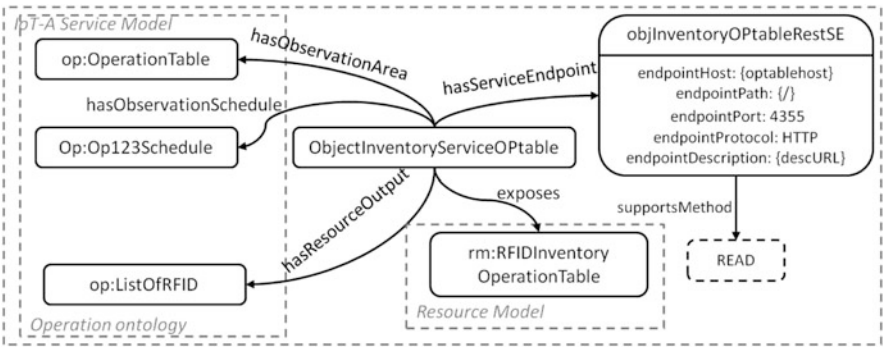

Fig. 12.16 Service description MUNICH platform

The use case is driven by events using asynchronous communication. Events are sent to the Event History network resource every time an RFID reader recognises a change in the number of RFID-tags in its observation area by using IoT Service storeEvent(event). The Event History resource provides another IoT Service that allows the subscription to notifications about the change in the status of towels, e.g. from unused to in use.

The structure of an Event data type is given as follows:

- Origin: {RFID reader instrument table; RFID reader operation table; RFID reader waste bin}
- Type: {RFID tag gone; RFID tag added; unrecognised tag}
- Time stamp

The sequence diagram below illustrates the interactions between Physical Entities and Functional Components of the architecture. The doctor takes a new towel out of the box on the instrument table and uses it in the patient's abdomen located on the operation table. The system detects the move of the towel from the instrument to the operation table by the disappearance of the respective RFID tag that is attached to the towel together with the appearance of the same RFID tag on the operation table. The Event Storage Service evaluates these single events towel disappeared on instrument table and towel appeared on operation table to a complex event towel in use (Fig. 12.17).

12.6.10 MUNICH Platform Conclusion

The previous Sections have shown that an existing system can be reverse engineered by applying the IoT ARM. Beginning from an existing system the modelling of the IoT Domain Model and Information Model has been demonstrated. With the help of these models the respective IoT Service Descriptions have been derived and the interactions between the Resources have been specified. The exercise did not include all the steps of the process to derive a concrete architecture based on the IoT ARM. There was neither a requirements analysis nor a security risk analysis undertaken. The purpose of this exercise is to demonstrate the usage of the models in first place. Since the functionality of the system has not changed a comprehensive requirements analysis has been skipped. Also the security risks are seen as manageable since the operating theatre is a well-secured and closed environment anyways. Only the event related service makes connections to external environments, but that was the case for the original system already and therefore no changes in security risks are expected. Particular platforms and solutions to implement the use case are not recommended here; technologies that would be suggested in this document might be outdated by the time of reading this document and therefore obsolete.

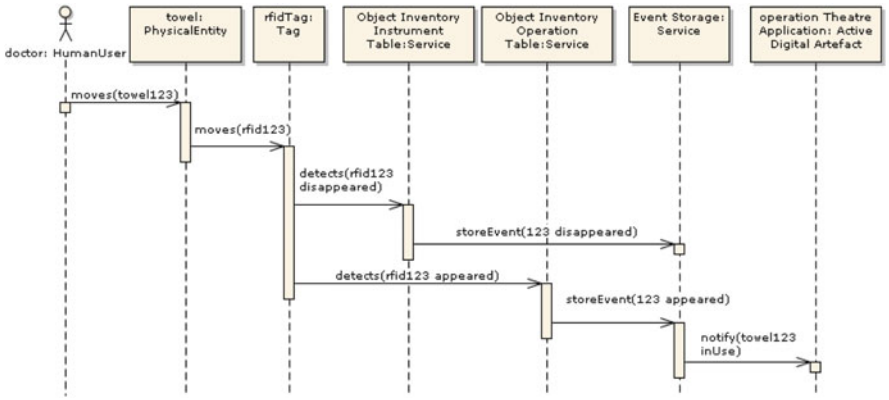

Fig. 12.17 Interactions MUNICH platform

12.7 Conclusions About Reverse Mapping

In this Section we have provided a reverse mapping of the IoT Architectural Reference Model with several standards from the field of IoT as well as a concrete architecture in order to provide an architectural validation, namely whether it is possible to map existing standards to the IoT ARM. If this was not possible, then the validity of the ARM itself would be questionable.

As we have seen in the detailed discussion of the different standards, whether a mapping is possible or not largely depends on the level of detail that we apply to the mapping. Especially for the Domain Model this becomes clear when we pick up the concept of a "Service": All the standards we looked at provide services in one way or the other, so that at a superficial glance a mapping is trivial. However, when we take the exact definition of that term in the different standards, we realize that there is not always a 1:1 correspondence between the standards. For instance, in ETSI M2M a service is not defined as "exposing resources on devices, but can interact with the devices." A resource concept as in IoT-A does not exist, so that compared to the definition of services and resources in the ARM, the distinction between a resource and the service as it is made in IoT-A does not exist in ETSI M2M. From a high-level perspective, though, the Domain Model usually maps rather well to the different standards. Also, the Communication Model and security aspects are rather compatible between the standards and the ARM. The latter is not surprising, as security aspects in the world of IoT are commonly derived from a well-established body of security research with fixed and clear terminology, quite unlike the Internet of Things domain. Also, it must be noted that the scope of IoT-A is broader than the scope of any of the individual standards. This is not surprising, as IoT-A aims to provide a Reference Architecture for all different kinds of specific architectures and use cases, and therefore must be broader by definition. Different parts of the IoT ARM are therefore only partially or not covered at all by different standards. For

instance, EPCglobal is highly RFID centric and therefore neglects certain aspects such as the IoT Communication Model, however the mapping to the IoT Domain Model and also to the Security and Information Model works reasonably well at the appropriate level of abstraction.

While the mapping of the different standards can be regarded as successful, when being performed at the appropriate level of detail, the real litmus test is the mapping of a concrete architecture to the IoT ARM. We have provided such a mapping for the MUNICH platform and have provided detailed information about the Domain Model, the Information Model, a process modelling based on the BPMN extensions developed in IoT-A (Meyer et al. 2013) and have discussed the service modelling in detail. Of course, we cannot generalize this successful exercise to any existing concrete architecture, but it still demonstrates nicely, how the IoT ARM can be applied to a concrete architecture. We are confident that other architectures from the domain of IoT map equally well to the IoT ARM.

12.8 Business Case Evaluation Example

12.8.1 Introduction

In the healthcare use case, to show the real-world value of the ARM, we focus on an IoT system that has already been implemented. In combination with the reverse mapping (see Sect. 12.6), we show that not only can the IoT ARM describe existing IoT systems (and by extension, help realise such systems), but that these systems also bring value. We evaluated the operating efficiency and profitability of such an IoT system.

This use case was implemented and carried out by several companies and universities in the framework for the Initiative for Cloud Computing in Health Care (henceforth referred to as the "MUNICH platform"). The MUNICH platform addresses two main problems: debris left in the human body after surgery and time-consuming process steps with no added value ("non-productive time"). A third auxiliary problem is the on-going integration of software and solutions from third party providers, which the IoT-A ARM would address.

Regarding the debris problem, in spite of safety checks already implemented, debris (tools, towels, consumables) is still left in the body during surgical procedures in 1:10,000 cases (Kranzfelder et al. 2012). In these cases, 70 % of the debris comes from surgical towels and 30 % from other surgical equipment (Kranzfelder et al. 2012). The consequences for the patient are a 40 % morbidity rate with a 5 % mortality rate (Kranzfelder et al. 2012). Regarding non-productive time, this refers to steps such as documenting and registering towels before the operation, subsequent counting of towels during the operation, and searching for towels when something is amiss; none of these steps add value, but instead address a problem created by the process itself.

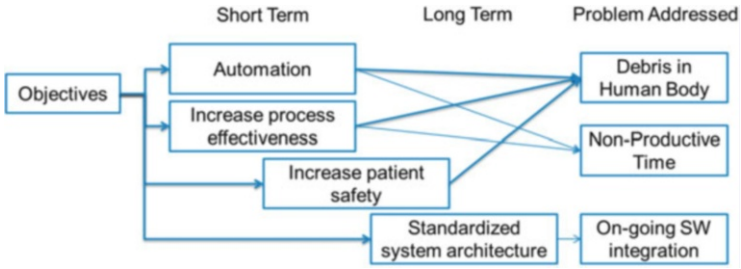

Fig. 12.18 Objectives of the healthcare use case and the problems addressed

Accordingly, a solution that addresses the tracking of surgical towels would mitigate these problems significantly. We can therefore map the MUNICH platform's objectives and solutions as shown in Fig. 12.18.

Real-time monitoring and location of all towels reduces the risk of debris in the human body because manual error-prone counting and searching is avoided (MUWS 2013). Therefore, the automation reduces manual errors. The process improvement increases the transparency of the process and reduces the risk of documentation errors that can also lead to debris in the human body. Experts estimate that a 100 % failure protection is possible with this solution (Kranzfelder et al. 2012). Addressing the debris problem meets short-term objectives of automation and improved process effectiveness, and in the mid-term, increases patient safety.

For the non-productive time problem, automation and the resultant process improvement remove the error-prone steps of documenting and registering towels before the operation, subsequent counting of towels during the operation, and searching for towels when something is amiss.

For the long-term problem of integrating new software developments from the hospital and their third party solution providers, the IoT-A ARM provides a standardised reference architecture. This would simplify the complexity of the architecture and make integration of new components into the system easier.

12.8.2 Cost and Benefit Models

The inputs for our analysis consisted of a cost model and a benefit model. The cost model factored in non-recurring costs (NRC) such as the RFID antenna and readers. The main cost driver is the hardware investment for the RFID antennas, which amounts to €49,500 – 58 % of the total non-recurring cost (€85,600). Beyond this initial investment, the cost model also factored in recurring costs (RC), such as the RFID-tagged towels, the software and system licensing fee, staff training, and the maintenance costs. The main cost driver of the recurring cost group is the operating fees of the system provider. This cost element has the most significant impact on the

cost model and accounts for 98 % of the yearly RC of €1,034,000. A price change in the service fee has a dramatic impact on the total cost structure over time. Therefore, this price change will be part of a specific sensitivity analysis.

The total cost (NRC+RC) development over a 6 year period was subsequently computed and input into a combined cost-benefit model (see 0 Cost-benefit analysis).

The benefit model is composed of three benefits; the calculated yearly benefits are in brackets: RFID-supported surgery (€815,000), cost savings from prevention of surgical errors (€370,000), and RFID-supported surgery preparation (€104,000). The "RFID-supported surgery" model provided the highest benefit, accounting for 63 % of total benefits. Non-tangible benefits not directly linked to a monetary outcome include an increase in surgical scheduling each year due to reduced preparation time, and hospital reputation improvements due to improved safety.

The total benefit over a 6 year period was subsequently computed and input into a combined cost-benefit model (see 0 Cost-benefit analysis).

12.8.3 Cost-Benefit Analysis

Figure 12.19 presents the yearly and cumulative cash flows. The cost-benefit analysis demonstrates a positive investment result. The discount factor is assumed at 8 % and the net present value is €805,000. The payback period is less than 1 year. Within Germany, according to healthcare experts, this would meet the requirement of a 1 year payback period for new investments in a German hospital.

12.8.4 Sensitivity Analysis

With the sensitivity analysis, we can investigate the impact of changing the major calculation variables. The following impacts shown in Table 12.6 will be discussed:

The results of the sensitivities are always evaluated with respect to the final effect on the discounted cumulative cash flow. The sensitivity analysis will be summarized with a best/worst case scenario.

12.8.4.1 Sensitivity Analysis for the Cost Model

The cost model sensitivity analysis investigates the impacts on the cost model if a parameter is changed. Reducing the critical risk factors (CRF) by 10 % leads to an increase in the total cash flow from €805,000 to €1,187,000, which is an increase in the net present value of 47 %. On the other hand, increasing the CRF by +10 % or +20 % due to higher NRC and RC lowers the net present value to €423,000 or €41,000 respectively.

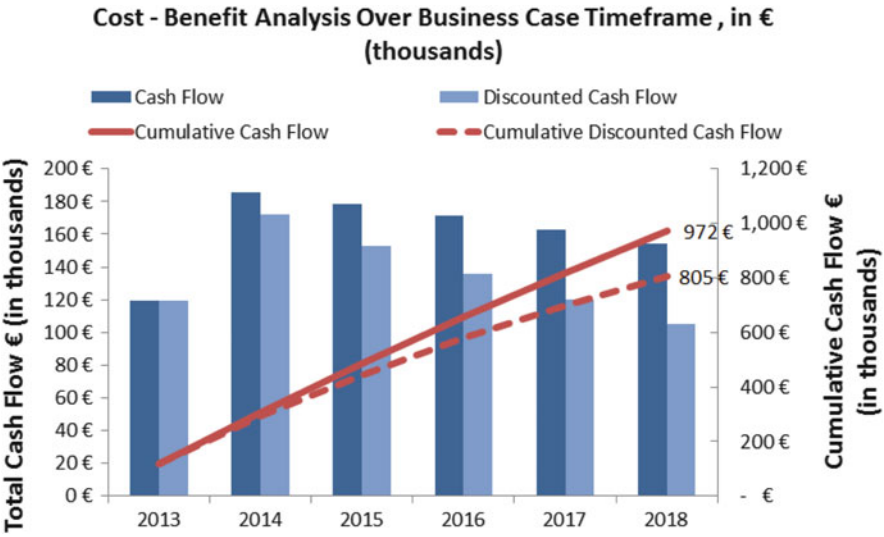

Fig. 12.19 Cost-benefit analysis over the business case timeframe (healthcare case)

Table 12.6 Models and parameters varied in the sensitivity analysis for the healthcare case

	Model element changed		
	Cost model	Benefit model	(c) General calculation assumptions
Change in variables	Critical risk factors: Software risk = SR Hardware risk = HR Personnel risk = PR Maintenance risk = MR System service fee (SFS)	Benefit variation factor (BSF)	Discount Rate (DF) Frequency of surgeries (TAoS)

The main cost driver for recurring costs (RC) is the system service fee. An increase of 10 % in the service fee per surgery from €20 to €22 reduces the net present value by two thirds to €270,000. The profitability limit is reached by increasing the fee to €23/surgery. The cost model sensitivity analysis is depicted in Fig. 12.20.

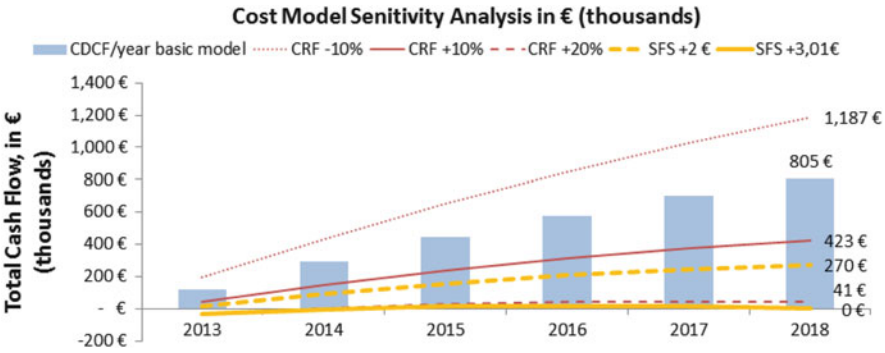

Fig. 12.20 Cost model sensitivity analysis (healthcare case)

12.8.4.2 Sensitivity Analysis Regarding Benefit Model Robustness

This analysis aims to investigate the robustness of the benefit model and the impact on the cost-benefit results. To demonstrate the development of the model, three different scenarios are simulated: (1) Benefits increase by 10 % (2) Benefits decrease by 10 % and (3) Benefits decrease by 15 %. The results of these simulations are summarized in Fig. 12.21. The net present value is exactly 0 when the benefits are reduced by 12.4 %.

Notably, the net present value is very sensitive to changes in the benefit model; there are large benefit differences between (1) and (2).

12.8.4.3 Sensitivity Analysis for the Assumptions in the General Calculation

After the sensitivity analysis of costs and benefits variations, the analysis is extended to variations of the general calculation assumptions that affect both models. Two parameters are used to simulate the results. The first is the change in the discount rate (DF) to reflect different risk perceptions and interest rate influences. The second parameter concerns the frequency of the surgeries per year (TAoS), which is a basic quantity variable (see Fig. 12.22).

A variation in the discount rate of ±2 % leads to an increase/decrease in the net present value of ±4 %. If a 12 % discount rate is assumed, the net present value falls to €741,000 (−8 %).

If the hospital performs 25 % fewer surgeries per year, the net present value decreases to €524,000 (−35 %). In contrast, if the number of surgeries per year increases by 25 %, the net present value rises by 35 % (€1,087,000). The net present value is zero if the hospital performs 71.5 % fewer surgeries per year.

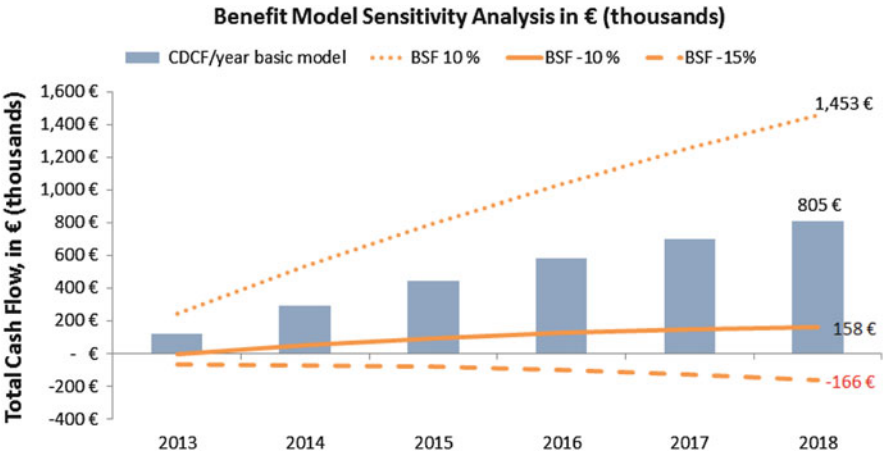

Fig. 12.21 Benefit model sensitivity analysis (healthcare case)

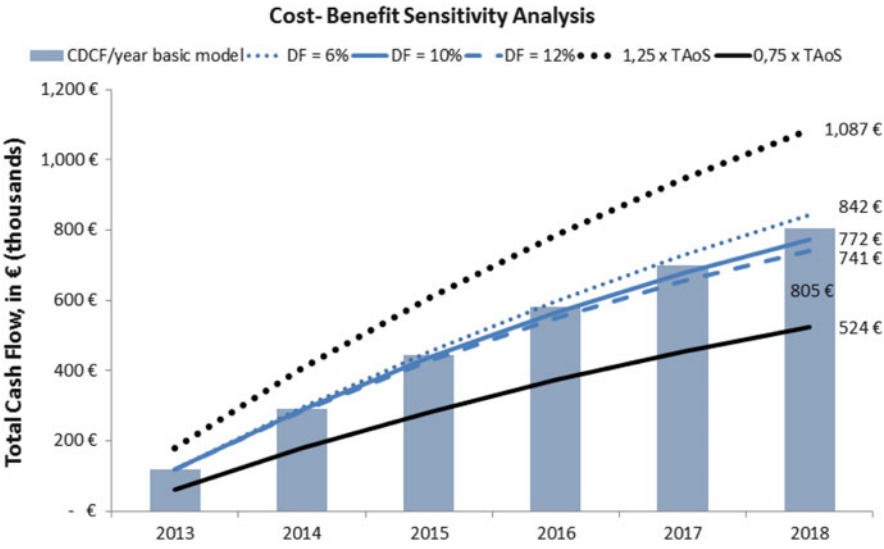

Fig. 12.22 Cost-benefit sensitivity analysis (healthcare case)

12.8.4.4 Best/Worst Case Scenario

By combining cost and benefit variation in the sensitivity analysis, best and worst case scenarios can be elaborated. For example, if the system service cost is reduced by €1/surgery ($= -5$ %) and the hospital performs 25 % more surgeries annually, then the net present value rises significantly to €1,421,000 (+77 %). The best case scenario is based on the assumption that the service provider can lower the cost of the service fee due to cheaper maintenance costs, additional development support

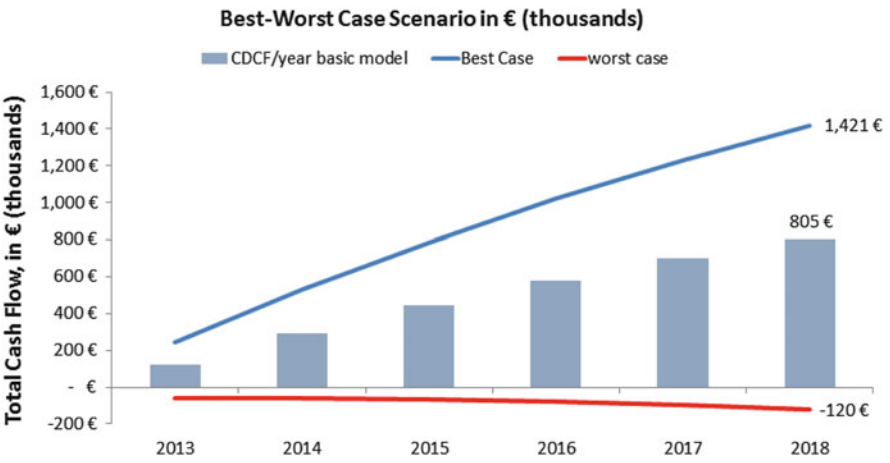

Fig. 12.23 Best and worst case scenario (healthcare case)

from using the IoT ARM, and from economies of scale effects. As a result of using the system and thereby reducing the errors, it is assumed that the hospital gains a better reputation and is more efficient, and accordingly, the number of surgeries per year rises.

In a worst case scenario it is assumed that the benefits are lowered by 5 %, the system service fee is €2/surgery more expensive (+10 %), and the number of the surgeries is reduced by 25 %. In this worst case scenario, the net present value is completely destroyed and always negative (see Fig. 12.23).

We observe that the economic feasibility of the case depends to a high degree on the system service fee of the service provider. The feasibility is also sensitive to fluctuations in the benefits. Further investigation about the reliability of the cost estimates is necessary. This information can be gained from the pilot deployments of the system with RFID-equipped towels. A test case is currently running in Munich at the university hospital "Rechts der Isar". When the pilot case is finished, a more reliable assessment of cost and benefits will be possible. The service provider would then also have better information for the calculation of the cost of the service fee.