Internet of Things - Architecture

IoT-A

Deliverable D7.5

Validation report

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

Doc. Ref.:	D7.5	
Responsible Beneficiary :	FHG IML	
Editor(s):	Martin Fiedler (FHG IML)	
List of contributors:	Nicola Bui (CFR), Jourik De Loof (ALU BE), Martin Fiedler (FHG IML), Edward Ho (HSG), Werner Liekens (ALU BE), Carsten Magerkurth (SAP), Benedikt Maettig (FHG IML), Alexander Salinas Segura (UniWue), Klaus Sperner (SAP), Julinda Stefa (CSD)	
Reviewers:	Nicola Bui (CFR), Martin Fiedler (FHG IML), Carsten Magerkurth (SAP), Klaus Sperner (SAP)	
Contractual Delivery Date:	31.07.2013	
Actual Delivery Date:	30.08.2013	
Status:	Final	
Version and date	Changes	Reviewers / Editors

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

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

Executive Summary

The deliverable D7.5 reports on the overall validation process of work package 7. Work package 7 managed to successfully implement different demonstrators out of the retail/logistics and health domain which were shown at various locations. In total 14 demonstrators were implemented. The first demonstration of prototypes at IoT week 2012 in Venice, Italy received strong interest from stakeholders and external parties. On the next iteration step in development a high integration of IoT-A components and conceptual integration of the IoT ARM was achieved, which was shown at IoT week 2013 in Helsinki, Finland. There, a stakeholder driven scene was integrated in the storyline and shown at the demonstration as well (see Section 3.2.7). The retail/logistics use case was chosen from another stakeholder to do a real-world pilot in a cold chain distribution centre (see Section 6.1).

The technical validation approach of D7.5 has only the definition and implementation of the use cases as subject, a validation of the IoT ARM is covered in D6.4 [Salinas Segura 2013]. The use case specific validation process itself follows different aspects. First it focuses on the technical implementation of use cases modelled with the IoT ARM, integrating the results of other work packages (see Chapter 3). Secondly, a validation by requirements was considered (see Chapter 4). Thirdly, a business analysis was done on both defined use cases (see Chapter 5). Lastly, stakeholder feedback and evaluation were taken as a validation approach (see Chapter 6).

This deliverable follows the report D7.2, which provided the exact definition of use cases in an overall storyline and D7.3, the implementation of first prototypes and D7.4 the final implementation of demonstrators.

While the focus of WP1 was on developing an IoT Architecture Reference Model (ARM), which can be applied in developing Internet of Things systems, D7.5 addresses the **Application** side of the overall project as seen in Figure 1. In the role of integrating the results of the other work packages, this deliverable applies the IoT ARM regarding modelling of use cases in the IoT domain. Furthermore D7.5 explains on what kind of Devices and Functional Components were used for an implementation on a per scene view.

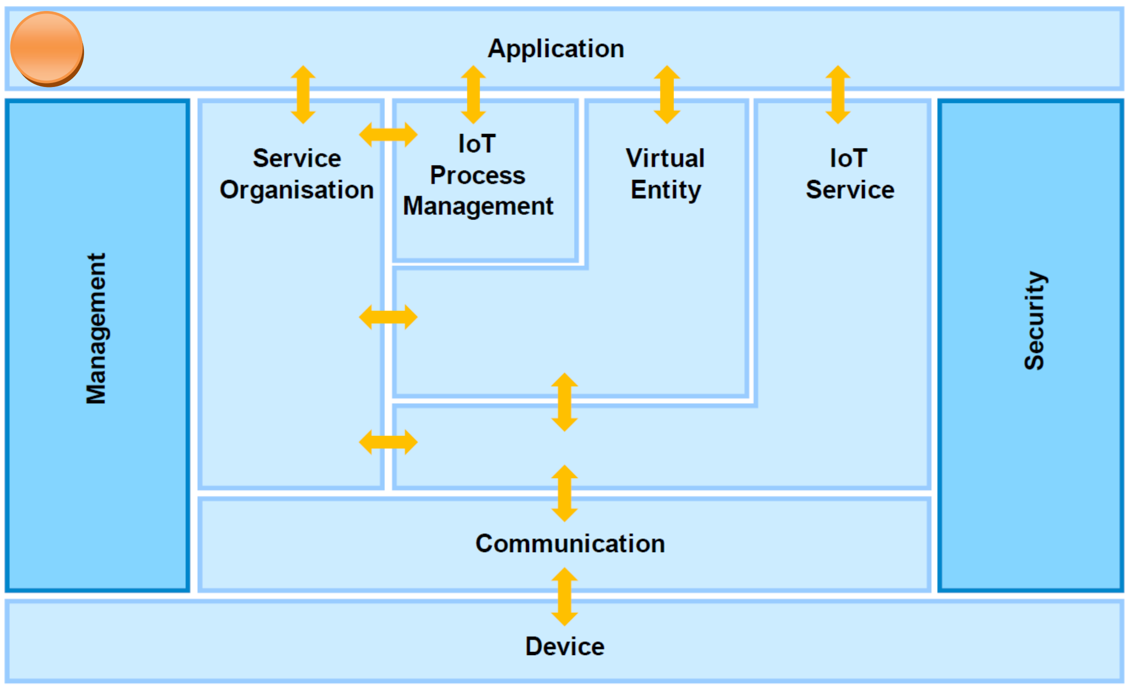


Figure 1: Functional Groups tackled in D7.5

Table of Content

List of abbreviations

Abbreviation	Description
ADA	Active Digital Artifact
API	Application Programming Interface
ARM	Architecture Reference Model
ATP	Availability to Promise
BE	Back End
BPMN	Business Process Modelling Notation
CoAP	Constrained Application Protocol
D	Deliverable
DA	Digital Artifact
DC	Distribution Centre
EHR	Electronic Health Record
ER	Emergency Room
ERP	Enterprise Resource Planning
ESL	Electronic Shelf Labels
GSM	Global System for Mobile Communications
GUI	Graphical User Interface
HCR	Health Care Record
HIS	Hospital Information System
ID	Identification
IoT	Internet of Things
IoT-A	Internet of Things Architecture
IP	Internet Protocol
NFC	Near Field Communication
NPV	Net Present Value
NRC	Non- Reoccurring Costs
PDA	Passive Digital Artifact
PE	Physical Entity
PIA	Privacy Impact Assessment
PoC	Point of Care

Abbreviation	**Description**
POS	Point of Sale
RC	Reoccurring Costs
REST	Representational State Transfer
RFID	Radio Frequency Identification
ROI	Return On Investment
RWIP	Real World Integration Platform
TDMA	Time Division Multiple Access
TFT	Thin-film transistor
UC	Use Case
USDL	Unified Service Description Language
UID	User Identifier
UML	Unified Modeling Language
URL	Uniform Resource Locator
VE	Virtual Entitiy
WLAN	Wireless Local Area Network
Wi-Fi	Wireless Fidelity
WP	Work Package

List of figures

List of tables

1 Introduction

The objective of this Deliverable, D7.5, is to summarize the results of the validation process of task four. The use case specific validation process itself is focused on different aspects.

First, a technical implementation of use cases modelled with the IoT ARM is shown. Here we show the process on how to come from an initial application idea to an IoT-compatible implementation following the IoT ARM Guidance in [Carrez 2013]. In the detailed implementation report we integrated the results of other work packages, e.g. Devices (i.e. hardware of WP5) and Functional Components (i.e. WP2's service orchestration, WP3's communication stack or resolution framework of WP4).

Secondly, since the IoT ARM was derived from requirements, one dimension for validating the presence of the IoT ARM in the WP7 use cases would be to check to what extent the requirements are present. These requirements come from stakeholder aspirations, from the state of the art of IoT projects, and from the technical experts in the consortium; accordingly, the requirements are at different levels in granularity of detail and abstraction.

Thirdly, the business case reveals the utility of the IoT ARM in combination with financial impact, i.e. the business value of the IoT ARM. For this purpose a business case on a quantitative basis was made. It concerns the retail use case from WP7 and the MUNICH platform in terms of healthcare. In both cases we applied a general business case framework which was adapted to our requirements. The business case for retail includes some use case scenes from WP7 and additional scenes which were mainly a result of expert statements tackling certain retail or logistics issues while the business case for the MUNICH platform was calculated for the use case of RFID supported surgeries. In both cases we conducted a comprehensive analysis to identify the potential costs as well as the potential benefits, transformed them into financial figures and finally performed a cost-benefit analysis. As this cost-benefit analysis is based on certain parameters, we supplementary added a sensitivity analysis which takes variations of these parameters into account and provides a range in which the upper and lower bound of the financial impact is indicated.

Finally, stakeholder feedback, evaluation and consideration in industry take-up were included to prove the usefulness of the developed use cases and demonstrators.

The Deliverable is structured as follows. Chapter 2 shows the validation process, from its initial conception in D7.1 [Hagedorn 2011] to the refinement of the use cases in D7.2 [Fiedler 2012], and the subsequent integration of IoT ARM Guidance and components from the technical work packages of the project. The implementation and experienced results in technical terms with specific stakeholder feedback are explained in Chapter 3. The validation by requirements and a business analysis on both use cases follow in Chapter 4 and Chapter 5. Initial industry take-up of demonstrators and an additional scene provided by a stakeholder is explained in Chapter 6.

2 Validation process and objectives

This chapter describes the continuous validation process beginning from M1 of the project timeframe. In the first project months the validation process was defined, while the execution of it began when the first results were available. The validation depicted here follows a technical approach, as the main activity of work package 7 was in the development and implementation of use cases. As a general rule, the work on the use cases created a test bed for the other technical work packages results. As such, the validation activities here focus on both use cases in the definition and implementation phase. Validation regarding the IoT ARM by itself is covered by work package 6 in its Deliverable D6.4 [Salinas Segura 2013].

In an overview, Figure 2 shows the timing of the defined deliverables and corresponding validation activities.

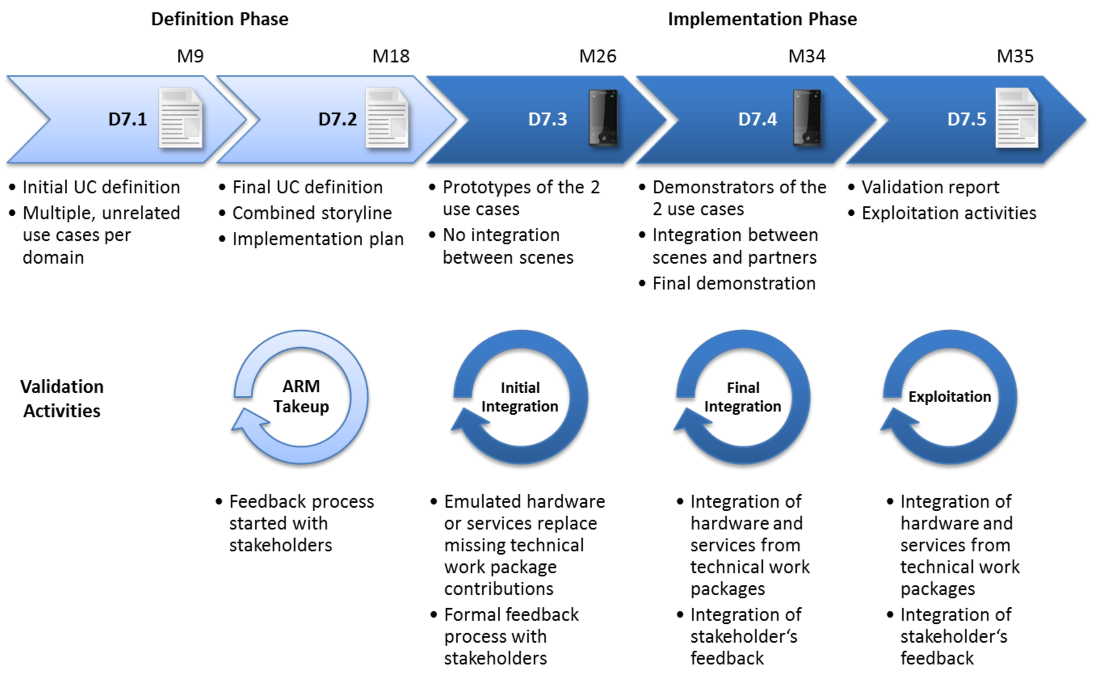

Definition Phase
Implementation Phase
M9
M18
M26
M34
M35
D7.1
D7.2
D7.3
D7.4
D7.5
• Initial UC definition
• Multiple, unrelated use cases per domain
• Final UC definition
• Combined storyline
• Implementation plan
• Prototypes of the 2 use cases
• No integration between scenes
• Demonstrators of the 2 use cases
• Integration between scenes and partners
• Final demonstration
• Validation report
• Exploitation activities
Validation Activities
ARM Takeup
Initial Integration
Final Integration
Exploitation
• Feedback process started with stakeholders
• Emulated hardware or services replace missing technical work package contributions
• Formal feedback process with stakeholders
• Integration of hardware and services from technical work packages
• Integration of stakeholder's feedback
• Integration of hardware and services from technical work packages
• Integration of stakeholder's feedback

Figure 2: Use case continuous validation process

The first Deliverable D7.1 [Hagedorn 2011] contained an initial contribution of multiple, unrelated use cases per domain. The use cases were proposed by different partners and reflected the ideas the different partners had on how the future Internet of Things would look like based on their experience and available technology. Initially, stakeholder opinions were collected to identify to what extent the various use case scenarios were reasonable. The excellent feedback provided gave an indication on which proposed use cases should be followed.

Deliverable D7.2 [Fiedler 2012] introduces integrated storylines for both use cases. A selection of the use cases from the individual contributions in D7.1 was made and a consolidated use case for the retail/logistics and healthcare domain with a realistic storyline was created. Each use case was structured as a storybook, consisting of several scenes linked together to represent day-in-a-life scenarios of everyday life of different roles and characters (e.g. Robert and Salomée) using IoT applications. A scene is defined as an action in a single location and continuous in time, that can be implemented and demonstrated in a standalone way. This gave the partners the flexibility to independently develop the scene without sacrificing the coherency of the storyline.

The first version of the IoT ARM (ARM v0.9, Deliverable D1.2) was considered for modelling purposes of the use cases. The IoT Domain Model was used to build a clear picture of each use case which provided feedback to the IoT ARM development. In general the IoT Domain Model was seen as a mature tool to model, though areas of improvement were identified as several issues arose due to unclear indication in the documentation itself. The IoT Functional View was used to indicate which Functional Components are needed for the upcoming implementation phase. The IoT-aware BPMN modelling of WP2 was used to model process-based use cases, e.g. the “Dynamic Pricing” scene in the retail use case.

As a side note, one defined scene out of each use case domain was included in the IoT Comic Book Special Edition [Presser 2012], namely Smart Medication (see Section 3.2.9) and Smart Logistics (see Section 3.3.1ff). This was seen as a huge acceptance of the IoT community towards the defined use cases of the IoT-A project.

With the beginning of the implementation phase the development of use case prototypes began. Following the implementation plan of D7.2, a periodic update of technical components of other work packages was followed. Here we followed a two-step approach, as most technical work packages were still working on conceptual ideas and did not have a finalized view yet, let alone a working implementation. Therefore in D7.3 the first step was to build prototypes, any components from the technical work packages were emulated by available hardware or by implementation in WP7 because of missing software and hardware components. An integration of the Resolution Framework of WP4 was considered as a first step, as a WP7 implementation was already available.

Following the second step in the implementation, we continuously updated the final demonstrators in D7.4, i.e. the own developed components with the results from the technical work packages. Besides internal tests, WP7 tested for software implementations. The Resolution Framework (WP4) and several hardware components (WP5) were integrated.

The first demonstration of the use cases (Deliverable D7.3), which focused on the retail use case, was shown at the IoT week 2012 in Venice, Italy. The use cases prototypes received strong interest from the audience which may be seen in Figure 3. Other dissemination opportunities regarding the demonstrators were the Future Internet Assembly (FIA) 2013 and the IoT week 2013 in Helsinki, Finland.

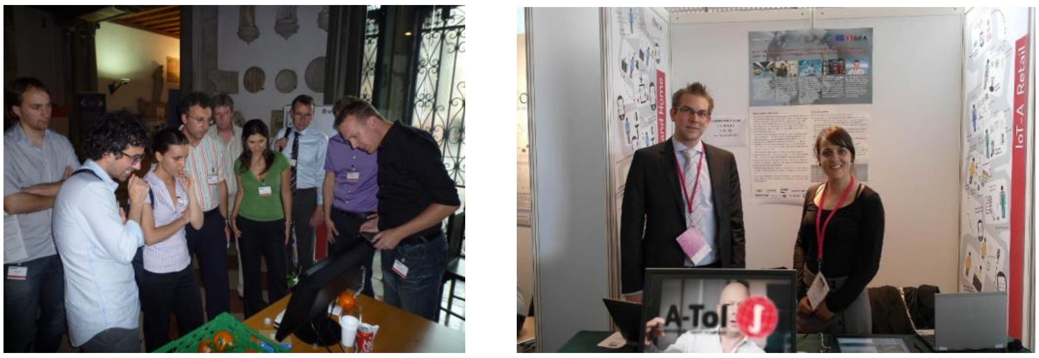


Figure 3: IoT-A demonstration at IoT week 2012, Venice (left) and FIA 2013, Dublin (right)

In total out of the 22 proposed scenes (8 health, 14 retail scenes) in D7.1 [Hagedorn 2011], 19 were redefined in D7.2 [Fiedler 2012] (9 health, 10 retail scenes) and 14 (9 health, 5 retail) were implemented in D7.3 and D7.4.

In this Deliverable the final version of the IoT ARM (ARM v3.0, Deliverable D1.5 [Carrez 2013]) was used to model the individual scenes of the implemented demonstrators. Here a short version of the IoT Guidelines was adapted to follow the process from an application description, to IoT ARM modelling and the final implementation. We used the IoT Domain Model, the IoT

Information Model, the IoT Functional Model and the Design Choices to come to the application. This process is shown in Section 3 of this document.

Another objective is to show to what degree the stakeholder requirements which formed the IoT ARM were considered in each use case. The list of unified requirements was taken as a basis and examined if the containing requirements are reflected in the use case. Chapter 4 covers this topic in detail.

A business analysis of both Health and Retail use cases are part of Chapter 5, which summarizes the results from D6.4 [Salinas Segura 2013]. Finally a privacy impact assessment (PIA) was performed on a specific scene of the Health use case to validate regarding privacy compliance. Section 3.2.1.7 gives a summary on the results.

3 Implementation of Use Case scenes

This chapter contains details on the implemented use case scenes. In total there have been 9 demonstrators been developed for the Health use case and 4 demonstrators for the Retail use case.

The following Section 3.1 gives an overview on which Functional Components are included in the implementation. Section 3.2 and 3.3 contain the details of each use case scene. We included a mini walkthrough on how to use the IoT ARM Guidance from chapter 5 of D1.5 [Carrez 2013] to come from an application idea to the final implementation.

3.1 Demonstrated concepts in implementation

This section gives a summarized overview on what kind of concepts of the IoT ARM are integrated by the demonstrators of both use cases in total.

Figure 4 shows the IoT Functional View of D1.5 [Carrez 2013] which is built out of Functional Groups and containing Functional Components. The figure shows which parts of the IoT ARM are implemented and used in the use cases (green), which components could be integrated in the future (orange) and which parts are not feasible within the current definition of the storyline (red). Some components marked orange could not be integrated because the development of the use cases stopped before the final delivery of the corresponding technical work package's functional component. In the current status about 45% of the ARM components are already integrated in the use case implementation.

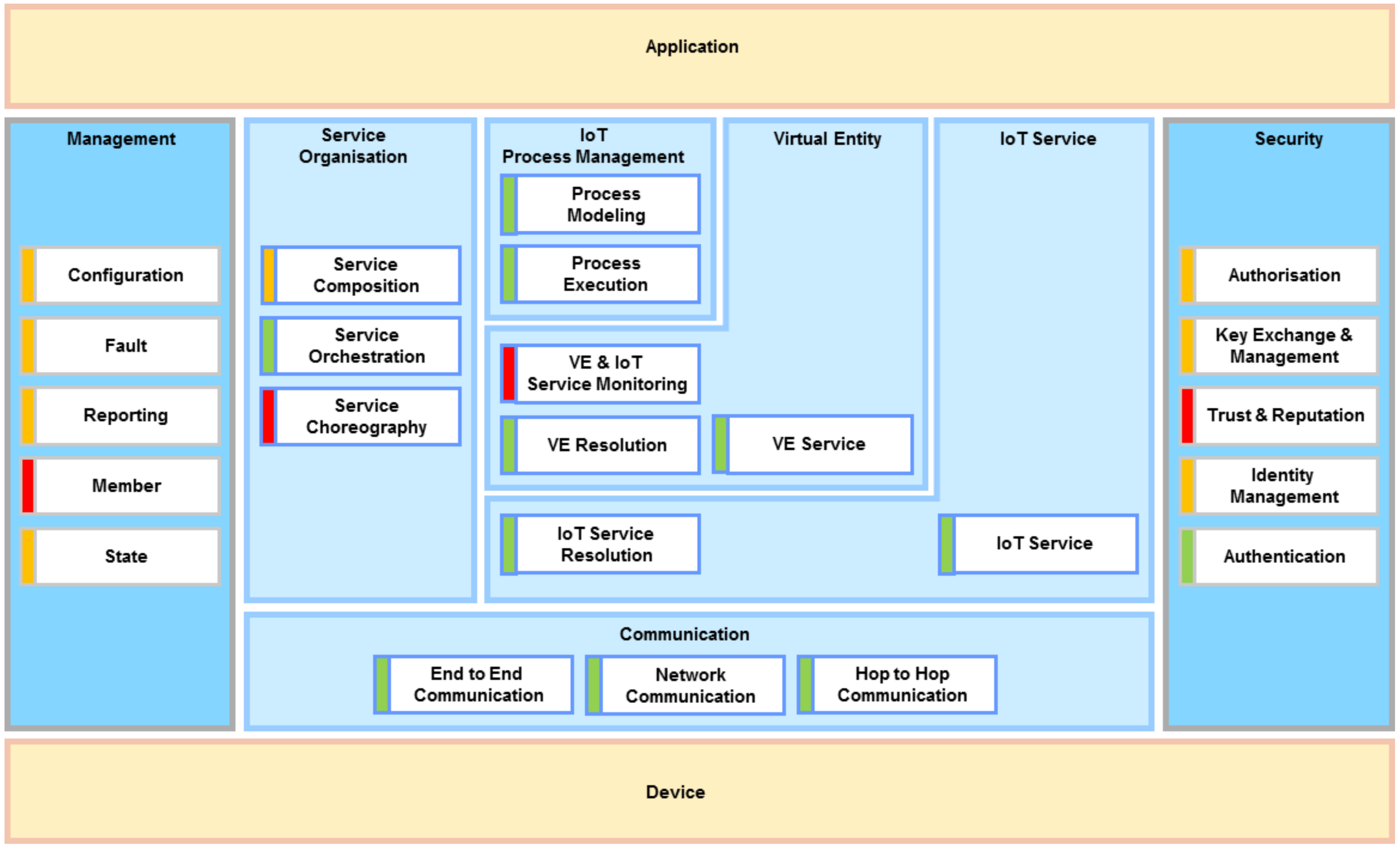


Figure 4: Functional View, mapping of used Functional Components in demonstrators

3.2 Health Use Case

3.2.1 Scene #1: Remote Patient Notification (ALU-BE)

The implementation of this scene is based on the “Remote Patient Notification” health use case scene 1 depicted in D7.2.

3.2.1.1 Step 1: Application Description

The scene shows how patients will benefit from IoT systems to help them take medicines or taking remote measurements on time. This is especially important for elderly patients which might suffer from beginning dementia. By having IoT systems, the patients might be able to stay longer at their own home instead of having to go to specialised care units. In the scene, the patient is notified that a certain action is required to be taken by the patient. In case the patient does not respond, an attempt is taken to draw his attention by using nearby IoT resources such as lights or buzzers.

Robert is reminded every morning that it is time for his daily routine of taking measurements. The time of day for the reminder is part of his Electronic Health Record (EHR). First, an alarm rings on his IoT-Phone which needs to be confirmed by Robert. In case Robert does not confirm, his last known location is determined and nearby IoT devices are located that could be used to draw his attention. In the demo, a nearby light is in the vicinity of Robert and is switched on and off. Unfortunately, that does not succeed either to draw his attention, so the EHR is scanned to look for a list of persons that can be contacted. Out of the list of possible candidates, the person who is closest is contacted to see if he/she can attend the patient. Again, the location is important and is used to select the most appropriate care giver. The demo then continues by sounding an alarm on the IoT-Phone of the care giver who finally manages to contact Robert and Robert acknowledges the alarm on his IoT-Phone.

3.2.1.2 Step 2: Domain Model Representation of the Demo

The domain model for the scene is shown in Figure 5.

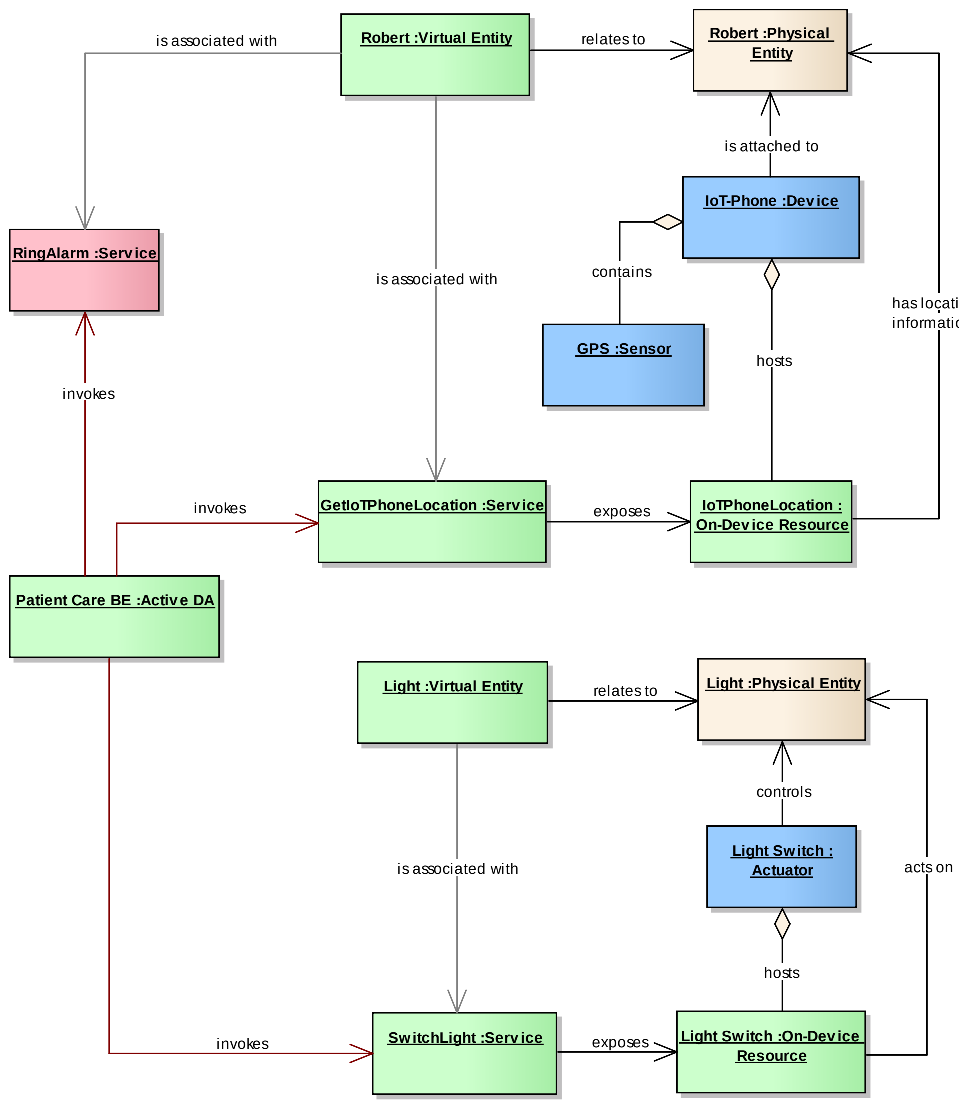


Figure 5: Domain Model of scene 1

Robert is the patient and is hence modelled as a Physical Entity with corresponding Virtual Entity which is associated with an alarm service. This service is running on the IoT-Phone but is considered a legacy service and therefore not further detailed. The alarm service is however integrated in the resolution framework and an association between this service and the Virtual Entity of Robert is made.

Robert's IoT-Phone is a Device which contains a GPS Sensor for location determination. The information from the GPS sensor is modelled as an On-Device Resource, exposed by the IoT-A Service GetSmartPhoneLocation.

Robert's Virtual Entity is associated with the GetSmartPhoneLocation Service so that if an application wants to find out the location of the Physical Entity Robert, it can do a simple lookup on the related Virtual Entity and find it.

The application in charge of driving the first scene is run in the backend and is the Active Digital Artefact Patient Care BE.

In the first scene a discovery mechanism is also illustrated to find a nearby light.

The light is a Physical Entity with a corresponding Virtual Entity. The light switch controlling the light is an Actuator and hosts the light switch On-Device Resource. The SwitchLightService then gives access to this On-Device Resource.

The Virtual Entity of the light is associated with the Service to switch the light on or off.

The Patient Care BE Digital Artefact will invoke the resolution framework and do a discovery for a light service which is located in a certain perimeter from Robert's last location. This service can then be invoked by the application to switch the lights on or off.

3.2.1.3 Step 3: Information Model Representation of the Demo

A graphical representation of the Information Model for scene 1 can be found in Figure 6.

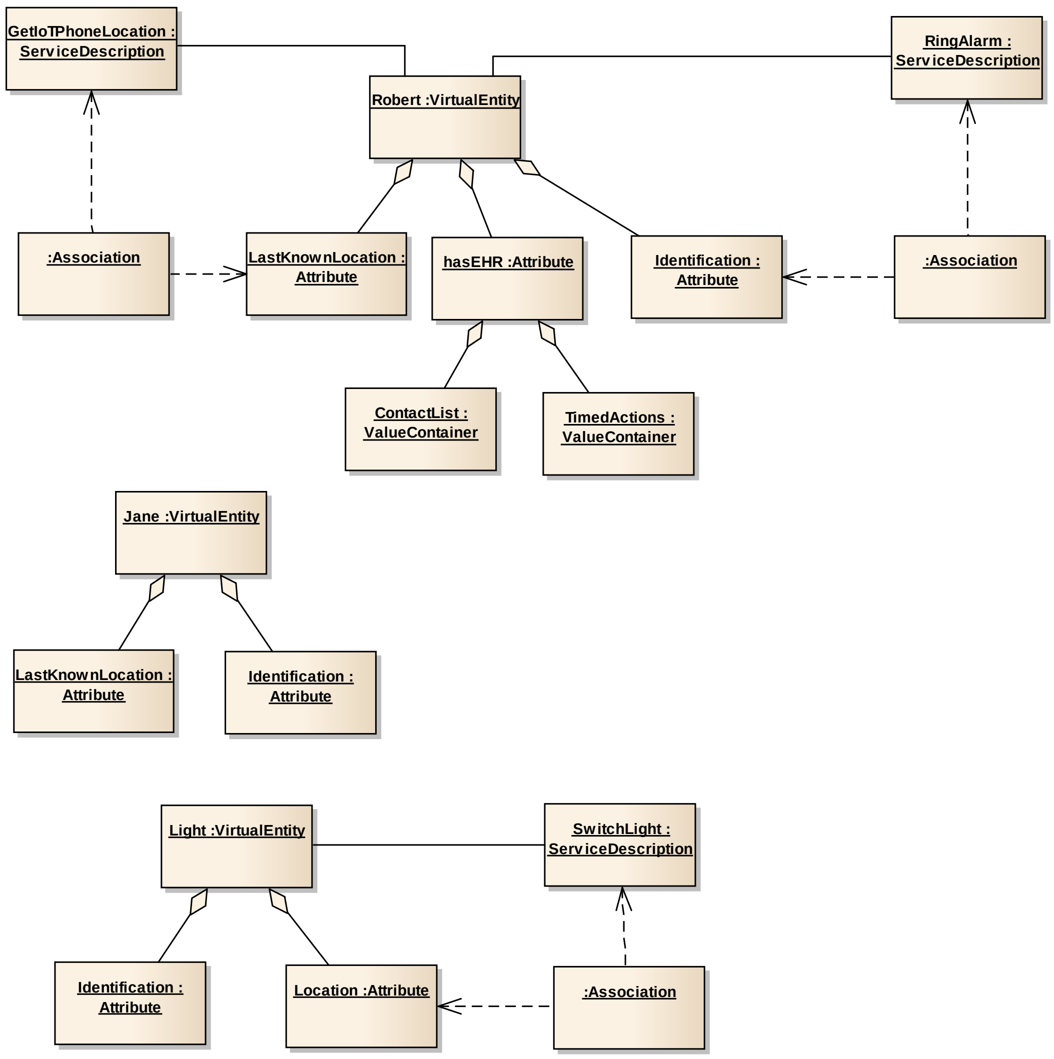

GetIoTPhoneLocation : ServiceDescription
Robert :VirtualEntity
RingAlarm : ServiceDescription
:Association
LastKnownLocation : Attribute
hasEHR :Attribute
Identification : Attribute
:Association
ContactList : ValueContainer
TimedActions : ValueContainer
Jane :VirtualEntity
LastKnownLocation : Attribute
Identification : Attribute
Light :VirtualEntity
SwitchLight : ServiceDescription
Identification : Attribute
Location :Attribute
:Association

Figure 6: Information Model of scene 1

There are three Virtual Entities pictured: Robert, Jane (care giver) and the Light that is used to draw the attention of Robert.

Robert has three attributes: his identification, his EHR and his location. The EHR contains two value containers: one with the contact list in case Robert does not answer his alarm and the other one which contains the action that should be executed (ringing the alarm for example).

Depicted as well is the Association between the Virtual Entity of Robert and a Service that provides location information of Robert.

The second Virtual Entity is that of Jane. She only has two attributes: her Identification and her last known location.

The last Virtual Entity is the light used to draw the attention of Robert. The light has two attributes: its identification and its location since the location is what is used by the discovery services of the resolution framework to find the light.

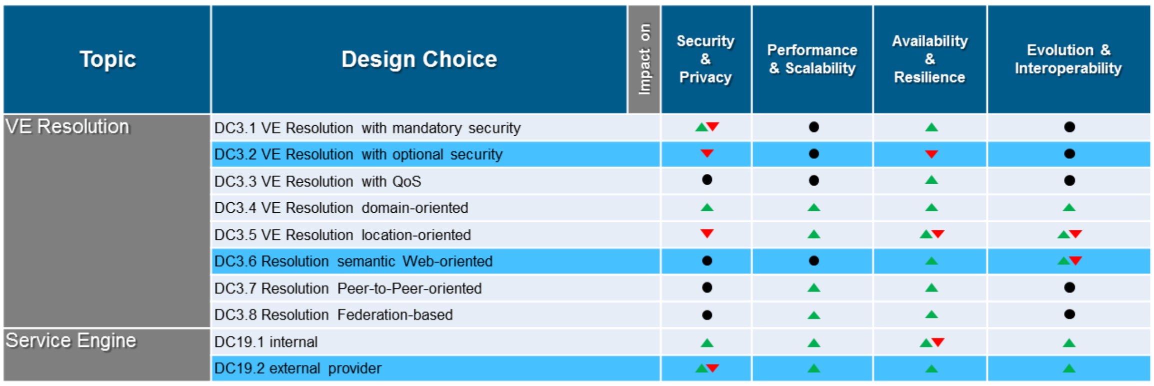
3.2.1.4 Step 4: Relevant Design Choices for Implementation

In order to build a demonstrator for the scene we took many aspects into consideration that do not have straightforward solutions, in fact under many implementation fields it is possible to find issues that can be solved in different ways and, while a given solution may result to be the best under a particular view, it is also possible that other solutions outperform the former adopting different views.

For scene 1 we consider the following design choices, which are also shown in Table 1.

- VE Resolution – handles functions needed for handling with resolution, monitoring, and storage of history of the virtual entity.

- Service Engine – In order to retrieve and discover and associating the services a choice had to be made where to run software in charge of this. The system integrator can choose to deploy it internally or rely on third parties.



Topic	Design Choice	Impact on: Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
VE Resolution	DC3.1 VE Resolution with mandatory security	▲▼	●	▲	●
	DC3.2 VE Resolution with optional security	▼	●	▼	●
	DC3.3 VE Resolution with QoS	●	●	▲	●
	DC3.4 VE Resolution domain-oriented	▲	▲	▲	▲
	DC3.5 VE Resolution location-oriented	▼	▲	▲▼	▲▼
	DC3.6 Resolution semantic Web-oriented	●	●	▲	▲▼
	DC3.7 Resolution Peer-to-Peer-oriented	●	▲	▲	●
	DC3.8 Resolution Federation-based	●	▲	▲	●
Service Engine	DC19.1 internal	▲	▲	▲▼	▲
	DC19.2 external provider	▲▼	▲	▲	▲

Table 1: Design choices relevant for scene 1. The highlighted choices are those picked for the demonstrator.

3.2.1.5 Step 5: Technical realisation of the Demo

The demo setup consists out of a number of servers, gateways and end devices with different communication technologies. The demonstrator application is running on the application server in the network and uses the WP4 resolution server for discovery, lookup of the services needed for scene 1. On the IoT phone a local eHealth demonstrator application is running which exposes services to the resolution network. Figure 7 gives an overview.

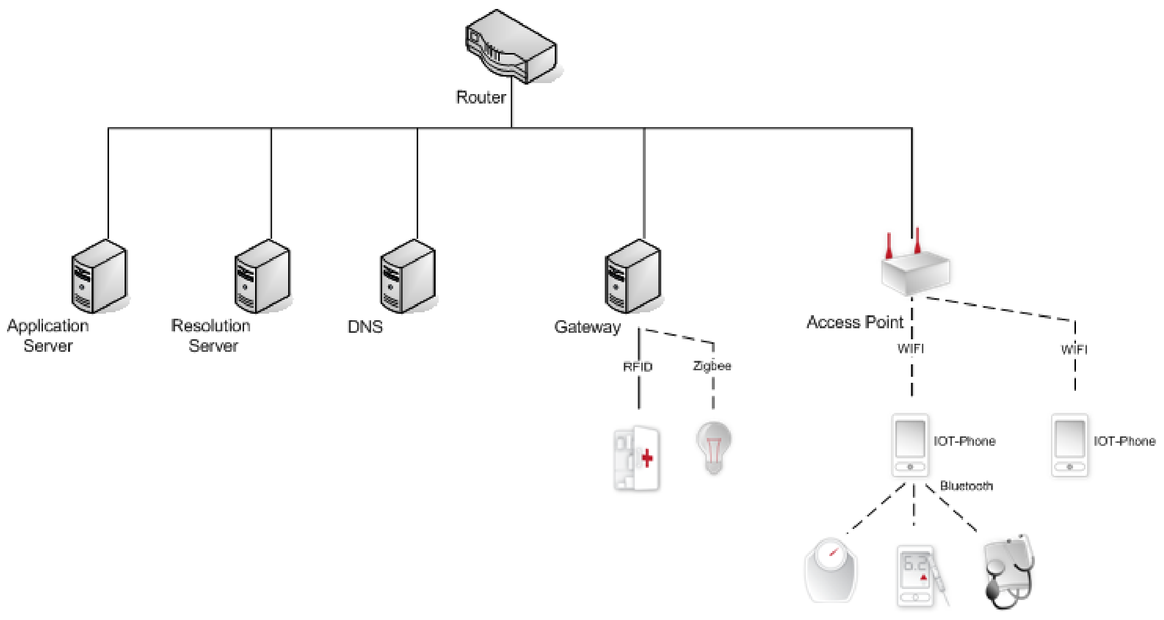


Figure 7: Physical setup for implementation of scene 1

The demonstrator application on the application server communicates with the fixed gateway/IoT phone over IPv6 network layer. For application layer communication CoAP and HTTP Rest is used.

IoT phone is connected to the network with WiFi.

The lamp controller uses 802.15.4 zigbee communication and is connected to the fixed gateway. On the gateway zigbee is translated to CoAP.

Description	Communication
IoT Phone	WiFi
Lamp Controller (switch)	802.15.4 (zigbee)
Gateway	WiFi + 802.15.4 + Ethernet

Table 2: Used Devices in implementation for scene 1

An overview of the physical implementation is shown in Figure 8 for scene 1.

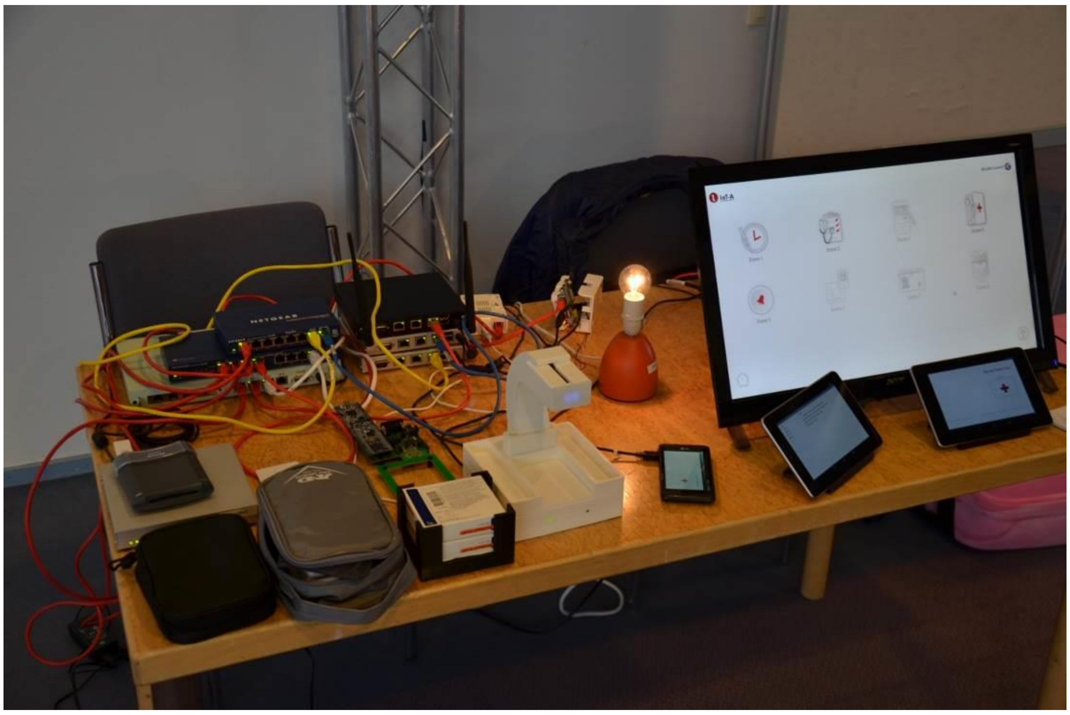


Figure 8: Demo setup at IoT week 2013, Helsinki

3.2.1.6 Feedback

The demonstrator was used to show the practical applications of the abstract IoT-A concepts – in particular the IoT-A resolution service and concepts (WP4) and internet of things protocol of WP3.

The spectators have always viewed the three demo described. The most technical remark or questions was on the service layer architecture (WP4). Non-technical is more about legal issues related to the eHealth domain.

3.2.1.7 Privacy Impact Assessment (PIA)

This scene has been taken as an example to do a full scale privacy impact assessment (PIA) on. Our approach was to examine existing frameworks on their usability in IoT environments. We choose to use the approach in [BSI 2011] for our analysis, as it directly covers the RFID technology field, which is regarding technology details in some parts similar to the IoT field. We applied the 6-step BSI guideline. Following the application description in step 1, the concerning privacy targets which relate to the chosen scenario were defined in step 2 and weighted in step 3 with protection demands for specific views. These steps are all related to the application itself and consequences of misuse of user data. Step 4 and 5 go more into the identification of threats and identification of possible controls in the (possibly planned) implementation. It was found that IoT ARM components may help to address specific threats.

Details on the fulfilled PIA may be found in D6.4 [Salinas Segura 2013].

3.2.2 Scene #2: Remote Patient Measurements (ALU-BE)

The implementation of this scene is based on the “Remote Patient Measurements” health use case scene 2 depicted in D7.2.

3.2.2.1 Step 1: Application Description

The scene shows how remote measurements can help to minimise hospital stays or ensure premium patient care beyond the hospital room reducing the cost of overall healthcare. Additionally, remote measurements can alert caretakers in case of injury or harm to the patient and assure a prompt medical intervention. In this scene, the patient will be assisted by the application through the process of taking regular measurements such as blood glucose level or blood pressure.

After Robert acknowledges the daily reminder as described in the scene above, he sees that it is time to take his daily measurements. Robert is guided through the measurements, first weight, then blood pressure and finally blood glucose level is measured. After each measurement, the data needs to be confirmed. The data is stored in his EHR. After the data is uploaded, an automatic data analysis of the results is performed. In case the analysis would show an anomaly, the caretaker is notified so he can login to the system and analyse the measurements to see if any action is needed or any adjustment to the medication should be proposed.

3.2.2.2 Step 2: Domain Model Representation of the Demo

In Figure 9, the modelling for scene 2 is depicted for the first part of the demo where the measurements are being taken.

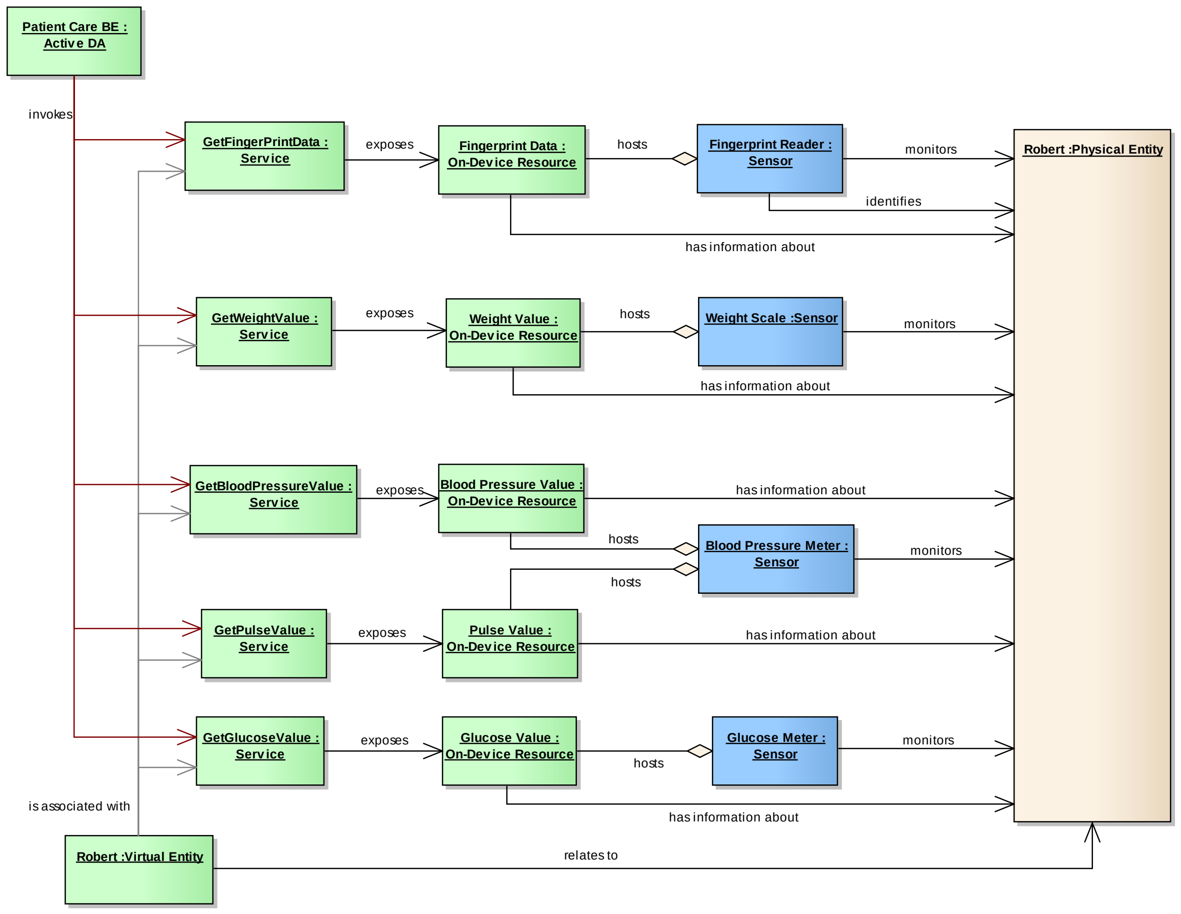


Figure 9: Domain Model of Remote measurements of scene 2

The modelling of the different sensors used in scene 2 for taking the weight, blood pressure and blood glucose level is quite straightforward. Each device such as the weight scale or the blood pressure meter is modelled as a Sensor, hosting an On-Device Resource. This On-Device Resource is then exposed by a Service.

For easy lookup, the Virtual Entity of Robert is associated with the Service corresponding to each device.

The backend application is Patient Care BE, which is an Active Digital Artefact (and thus a User) and will look up the associations to find the different services before invoking them.

Figure 10 shows the modelling for the application where the automatic data analysis takes place.

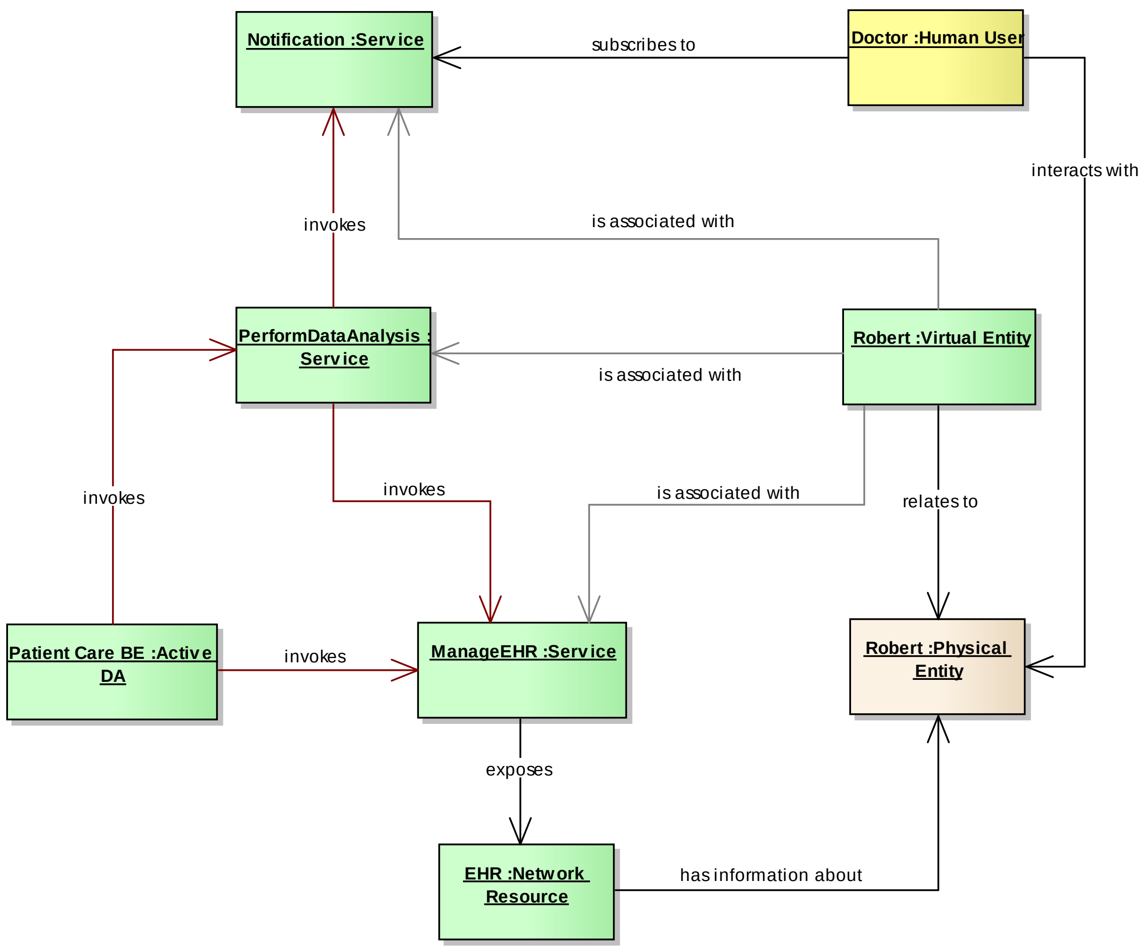


Figure 10: Domain Model of Data analysis of scene 2

Once all measurements are taken, the data must be archived and an analysis is performed.

The data is stored in the Electronic Health Record, seen in the figure as a Network Resource. This Network Resource is exposed by the EHR Service.

Robert's Virtual Entity is associated with this service, so that the backend application Patient Care BE can perform a lookup and find the service to update the Service providing access to the EHR of Robert.

Once the data is archived, the Patient Care BE will invoke the Service Data Analysis, which will fetch the data in the EHR of Robert, check if the measurements did not exceed limits and in case they did, invoke the Notification Service. This Service is again associated with the Virtual Entity of Robert.

The doctor, a Human User has subscribed to the Notification Service and receives notifications sent by the Data Analysis Service.

After receiving the notification, the doctor will typically access the EHR to consult the patient's data and to decide if some actions are required. This step in the use case is not represented in the diagram above.

3.2.2.3 Step 3: Information Model Representation of the Demo

The graphical representation of the Information Model for the first part of the scene where the measurements are taken is depicted in Figure 11. It is a generic representation that applies to all measurement devices of the scene. I.e. it equally applies to the devices for taking blood pressure, blood glucose level, weight, heartbeat etc.

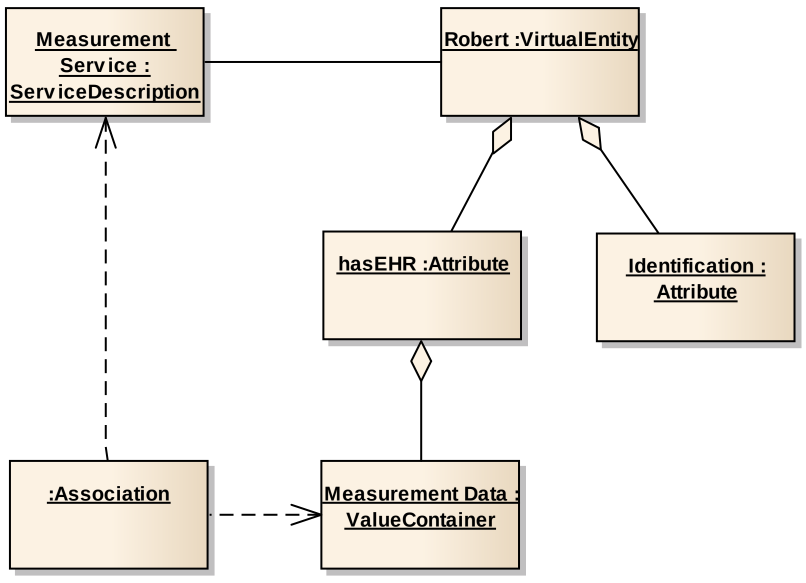

Measurement Service : ServiceDescription
Robert :VirtualEntity
hasEHR :Attribute
Identification : Attribute
:Association
Measurement Data : ValueContainer

Figure 11: Information Model of scene 2 - taking measurements

Robert has an Attribute which is the EHR. The EHR then contains a Value Container which holds the measurement data. Associated with the Virtual Entity of Robert is the Service that exposes the sensor that is taking the actual measurements. The application driving the whole scene is not depicted in this model.

The second part of the demo where the automatic analysis is performed is shown in Figure 12.

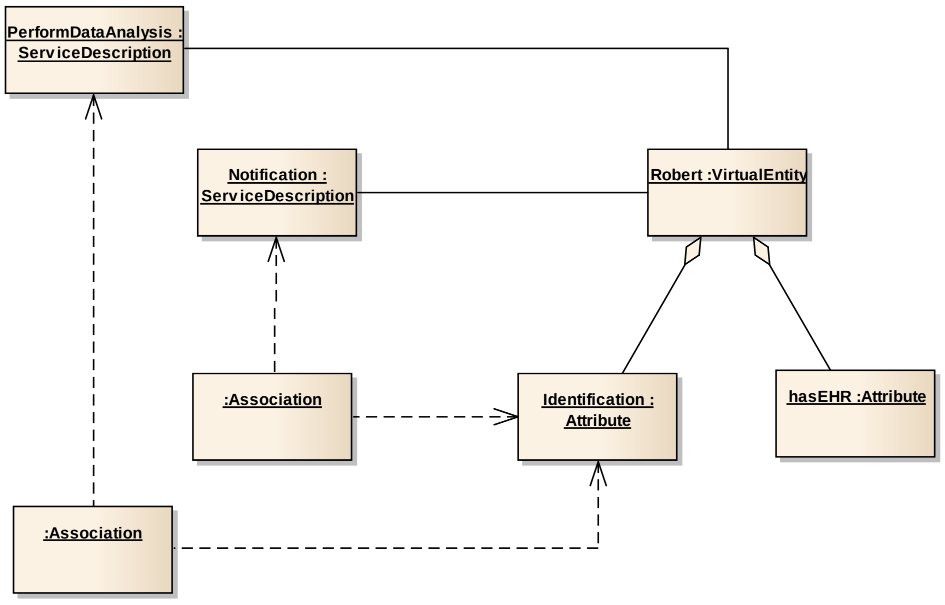

PerformDataAnalysis : ServiceDescription
Notification : ServiceDescription
Robert :VirtualEntity
:Association
Identification : Attribute
hasEHR :Attribute
:Association

Figure 12: Information Model of scene 2 - data analysis

Robert has two Attributes: his identification and his EHR. Associated to Robert there are two services: one that performs the data analysis and the other one that notifies the doctor in case an anomaly has been detected.

3.2.2.4 Step 4: Relevant Design Choices for Implementation

In order to build a demonstrator for the scene we took into consideration many aspects that do not have straightforward solutions, in fact under many implementation fields it is possible to find issues that can be solved in different ways and, while a given solution may result to be the best under a particular view, it is also possible that other solutions outperform the former adopting different views

For scene 2 we consider the following design choices, which are also shown in Table 3.

- VE Resolution – handles functions needed for handling with resolution, monitoring, and storage of history of the virtual entity.
- Information Storage – deals with where and how to store information. For medical information there could be some legal issues be involved that influence the decision on where to store the data.

Topic	Design Choice	Impact on			
		Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
VE Resolution	DC3.1 VE Resolution with mandatory security	▲▼	●	▲	●
	DC3.2 VE Resolution with optional security	▼	●	▼	●
	DC3.3 VE Resolution with QoS	●	●	▲	●
	DC3.4 VE Resolution domain-oriented	▲	▲	▲	▲
	DC3.5 VE Resolution location-oriented	▼	▲	▲▼	▲▼
	DC3.6 Resolution semantic Web-oriented	●	●	▲	▲▼
	DC3.7 Resolution Peer-to-Peer-oriented	●	▲	▲	●
	DC3.8 Resolution Federation-based	●	▲	▲	●
Information Storage	DC20.1 local only	▲	▼	▲▼	●
	DC20.2 web only	▼	▲	▲▼	●
	DC20.3 local and web cached	▲▼	▲▼	▲▼	●

Table 3: Design choices relevant for scene 2. The highlighted choices are those picked for the demonstrator.

3.2.2.5 Step 5: Technical realisation of the Demo

The demo setup consists out of a number of servers, gateways and end devices with different communication technologies. The demonstrator application is running on the application server in the network and uses the WP4 resolution server for discovery, lookup of the services needed for scene 2. On the IoT phone a local eHealth demonstrator application is running which exposes services to the resolution network. Figure 13 gives an overview.

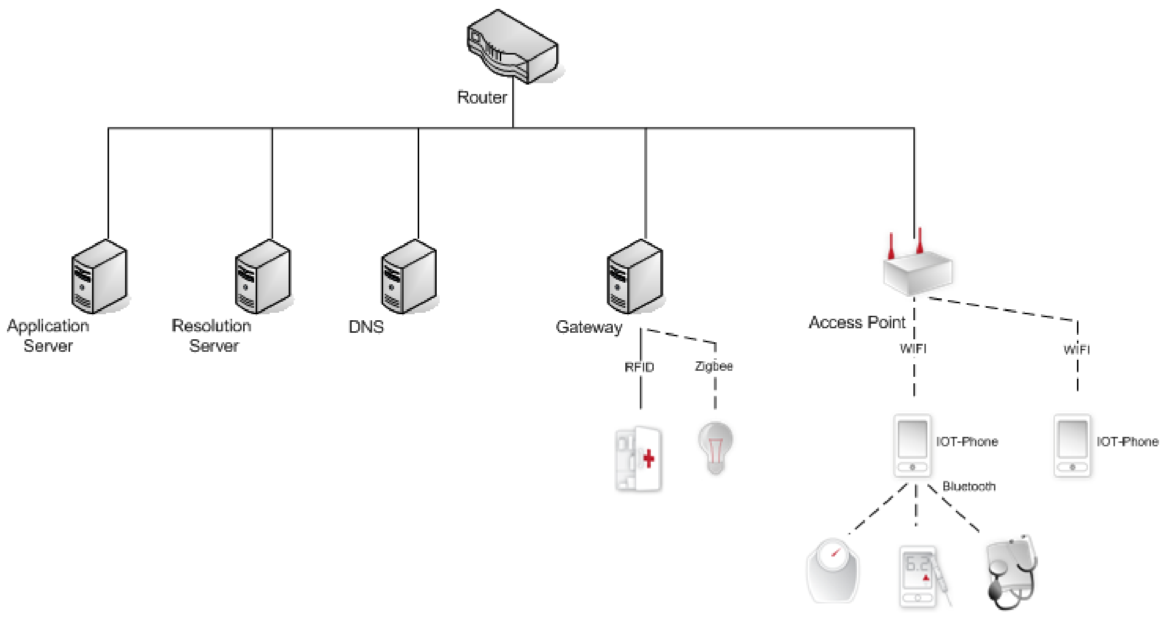


Figure 13: Physical setup for implementation of scene 2

Description	Communication
IoT Phone	WiFi
Weight Scale	Bluetooth
Blood Glucose Level Reader	Bluetooth
Blood Pressure Meter	Bluetooth

Table 4: Used Devices in implementation for scene 2

The IoT phone is used as mobile gateway to connect to the medical measurement devices over Bluetooth. The IoT phone is connected to the network over WiFi.

Communications on the network layer are inline with WP3 configuration prescription, namely IPv6. On the application layer HTTP rest and CoAP is used.

An overview of the physical implementation is shown in Figure 14 for scene 2.

Figure 14: Demo setup at IoT week 2013, Helsinki

3.2.2.6 Feedback

The demonstrator was used to explain the practical applications of the abstract IoT-A concepts – in particular the IoT-A resolution service and concepts (WP4) and internet of things protocol of WP3.

The spectators have always viewed the three demo described. The most technical remark or questions was on the service layer architecture (WP4). Non-technical is more about legal issues related to the eHealth domain.

3.2.3 Scene #3: Remote Patient Care: insulin alarm (CFR)

The implementation is based on use case I: Health and Home scene 3. The scene showed how IoT systems can help users to perform periodic measurements such as blood glucose level and can provide user friendly mechanisms to keep the Health Care Record (HCR) up to date. In particular, the demonstrator consisted in remote application reminding the user to perform the measure and connectivity feature enabling direct interactions among the measuring instrument, the user PDA and the remote database.

The IoT-A concepts shown in the demonstrator were WP3 connectivity features and WP4 resolution mechanisms.

3.2.3.1 Step 1: Application Description

Jane is associated in the digital domain to the BloodGlucoseControl service, which is a remote service to remind her for making periodic measurement. This service periodically monitors Jane's HCR for the time of her last measure and if the time elapsed from the last is longer than a given threshold an alarm is generated.

The resolution engine is in charge of resolving all the logic association into devices addresses in order for the proper communication to be realised.

The generated alarm is then forwarded to Jane's health care application as a notification to perform the measurement. The application waits in the background until the measurement instrument replies with the new data. Finally, the updated information is stored in the database.

3.2.3.2 Step 2: Domain Model Representation of the Demo

The domain model for this scene is shown in Figure 15 and Figure 16, of which the former illustrates the interaction between Jane, her devices and the services, while the latter introduces the human users (Jane and doctor) and their relationships.

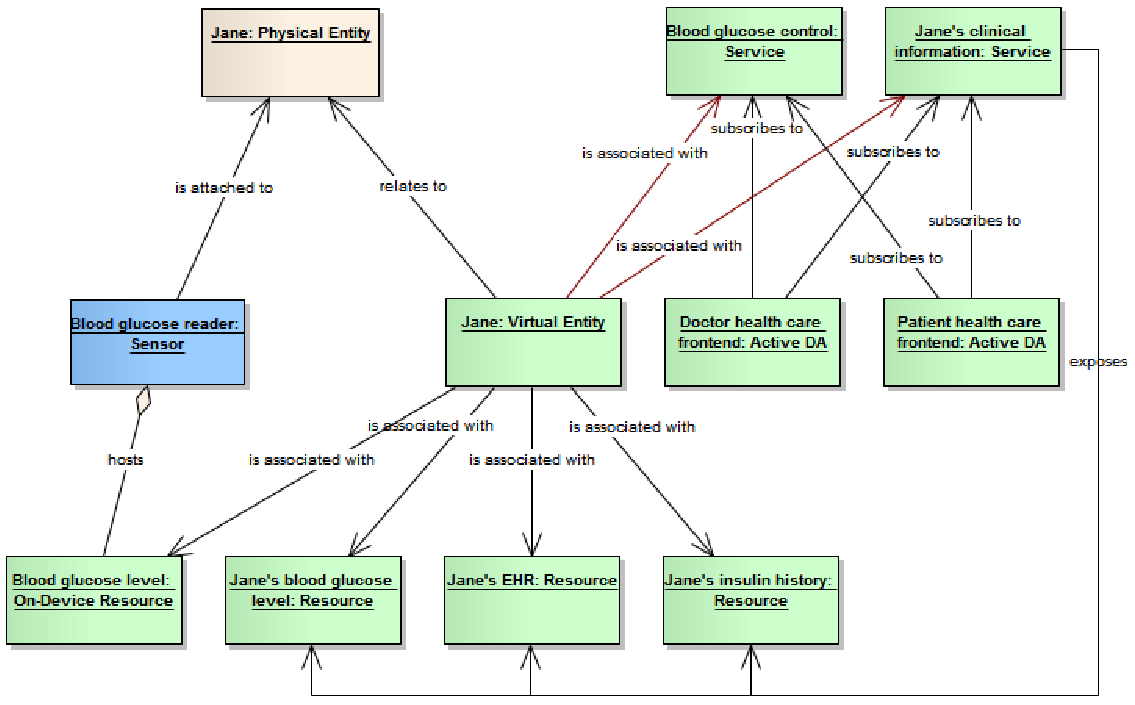

Jane: Physical Entity
Blood glucose control: Service
Jane's clinical information: Service
is attached to
relates to
is associated with
subscribes to
subscribes to
subscribes to
is associated with
subscribes to
Blood glucose reader: Sensor
Jane: Virtual Entity
Doctor health care frontend: Active DA
Patient health care frontend: Active DA
exposes
hosts
is associated with
is associated with
is associated with
is associated with
Blood glucose level: On-Device Resource
Jane's blood glucose level: Resource
Jane's EHR: Resource
Jane's insulin history: Resource

Figure 15: Domain Model of scene 3 part A

The whole domain model of the scene revolves around Jane's virtual entity, which is associated with the resources needed to execute the services. In particular, the clinical information service exposes Jane's blood glucose level, EHR and insulin history resources, while another blood glucose level resource is hosted by the blood glucose reader device. The blood glucose level service is in charge of checking Jane's insulin history through the clinical information service and, if needed, to remind Jane of her measurement. The blood glucose control service synchronises Jane's EHR and blood glucose level resources after every measurement.

Finally, the user plane is managed through two digital artefacts, the doctor's and the patient's health care front ends. These digital artefacts are the human-system interface needed to let the human users to interact with the services (see Figure 16).

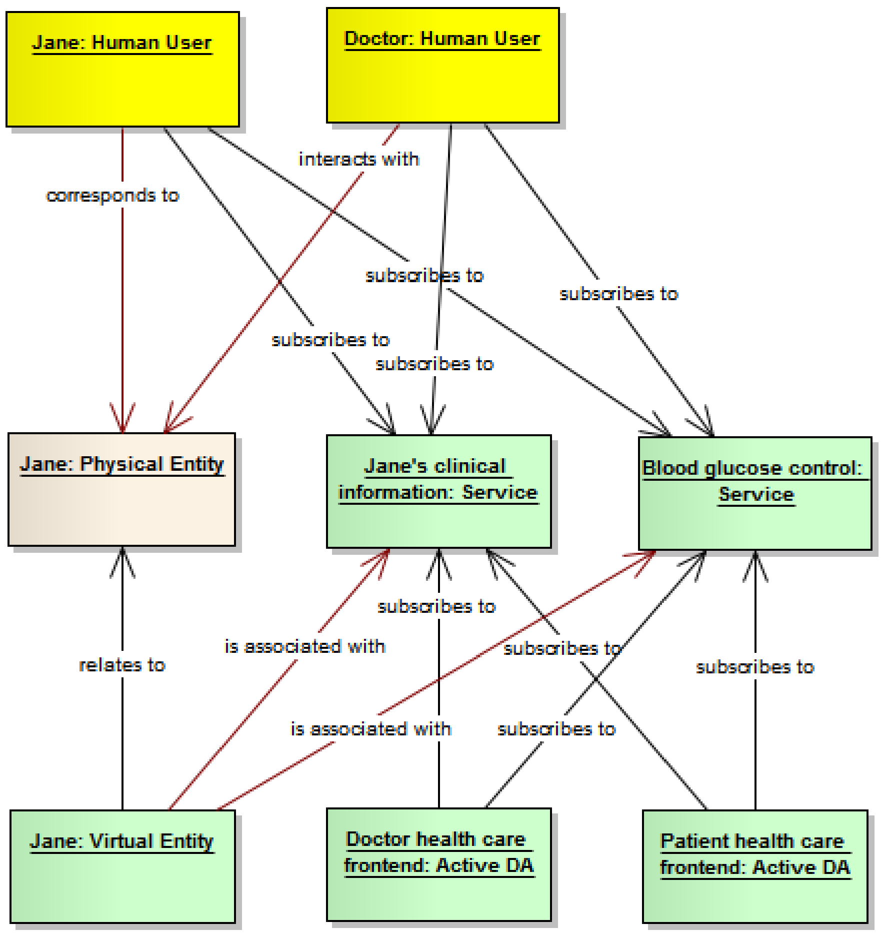


Figure 16: Domain Model of scene 3 part B

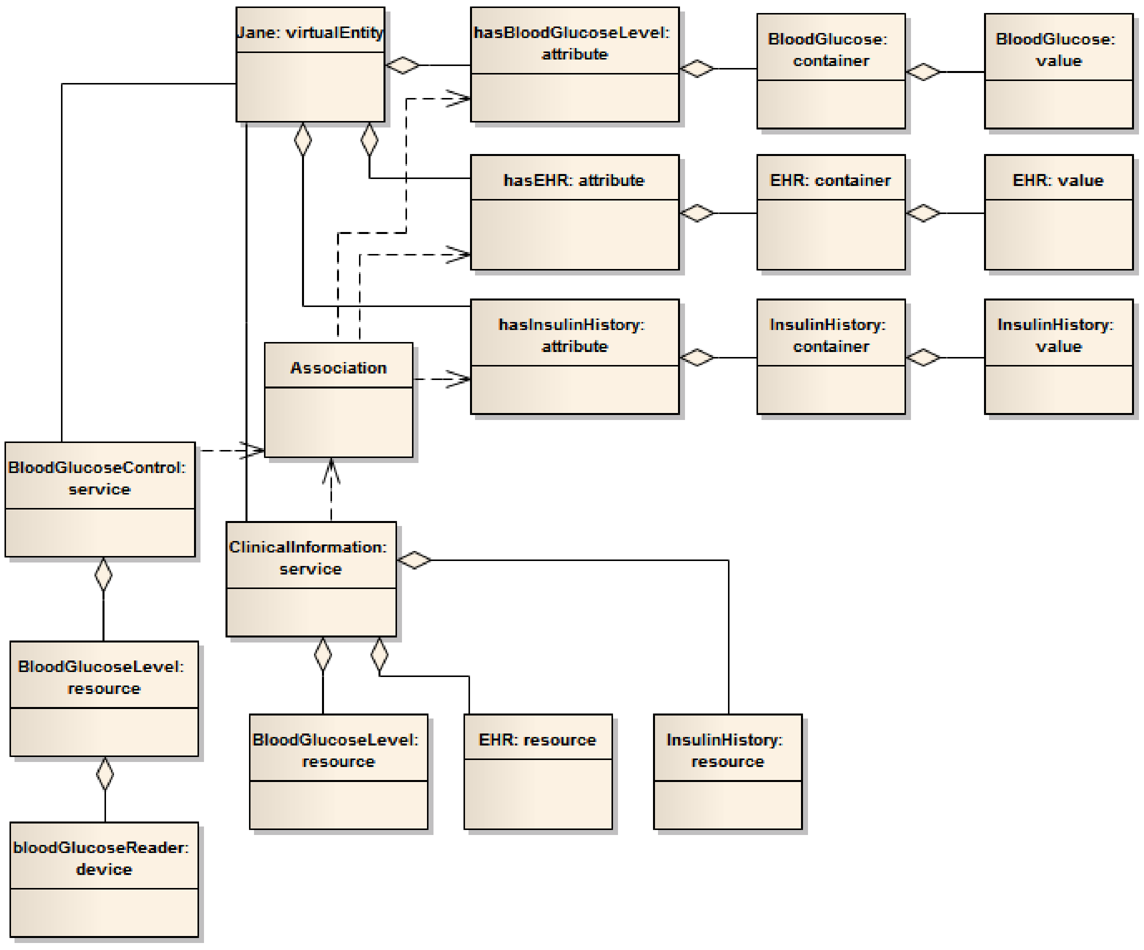
3.2.3.3 Step 3: Information Model Representation of the Demo

Figure 17 provides a graphical representation of the information model described in D1.5 [Carrez 2013] specialised to scene 3 data. In the upper part of the figure Jane's virtual entity data is listed: three attributes specify her blood glucose level, her EHR and insulin history.

For every attribute a data container is used to contain one or more values of the information: while the blood glucose and the insulin history attributes maintain multiple values, the EHR is a single valued attribute.

In the bottom part of the figure, instead, the two main services are shown: the blood glucose control service, to which the blood glucose resource is connected to, and the clinical information service which provide the blood glucose, the EHR and the insulin history resource.

Finally the blood glucose resource of the blood glucose control service is connected to the blood glucose reader device.



Jane: virtualEntity
hasBloodGlucoseLevel: attribute
BloodGlucose: container
BloodGlucose: value
hasEHR: attribute
EHR: container
EHR: value
hasInsulinHistory: attribute
InsulinHistory: container
InsulinHistory: value
Association
BloodGlucoseControl: service
ClinicalInformation: service
BloodGlucoseLevel: resource
BloodGlucoseLevel: resource
EHR: resource
InsulinHistory: resource
bloodGlucoseReader: device

Figure 17: Information Model of scene 3

The two parts of the information model are connected through associations between service and Jane's Virtual Entity. These associations are mapped through Jane's attributes.

3.2.3.4 Step 4: Relevant design choices for Implementation

In order to build a demonstrator for the scene we took into consideration many aspects that do not have straightforward solutions, in fact under many implementation fields it is possible to find issues that can be solved in different ways and, while a given solution may result to be the best under a particular view, it is also possible that other solutions outperform the former adopting different views.

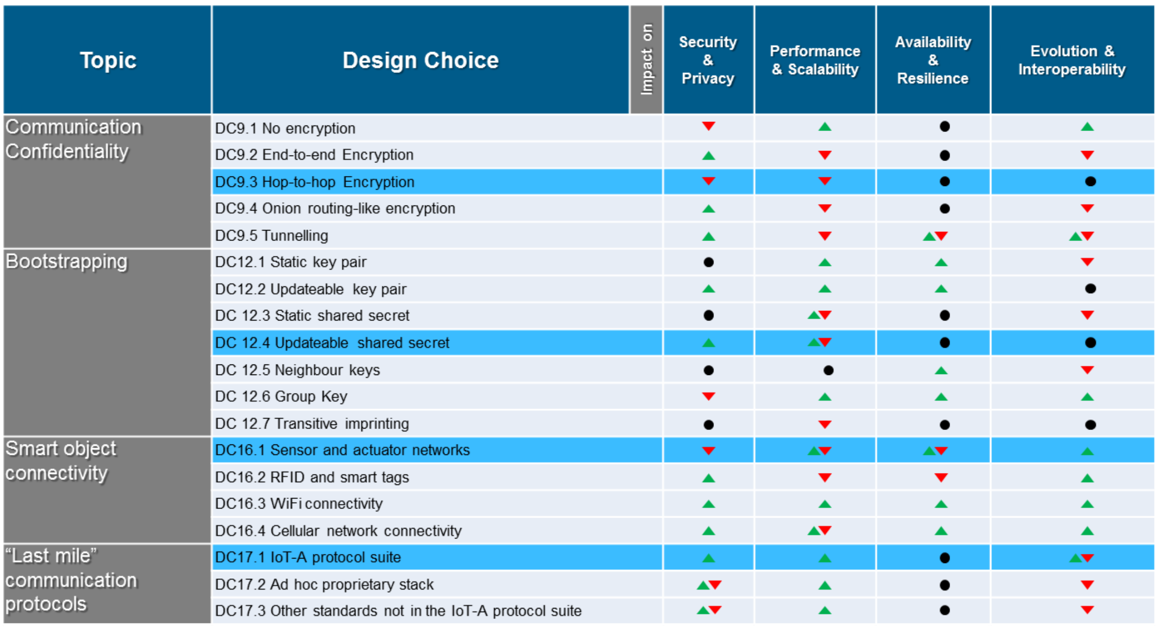
All that said, we thought that in this particular scene the design choices that impacted the most on the final result are the following (see also Table 5):

- Communication confidentiality – dealing with medical information is a very sensible operation, thus enforcing confidentiality is of paramount importance. While the tunnelling solution looks to be the best overall, we have been forced to adopted hop-by-hop encryption due to computational constraints.
- Bootstrapping – security and confidentiality depends on intrinsically secure elements. In order to provide devices with these secure elements, many solutions can be adopted.

Our demonstrator is built adopting the updateable shared secret, which, not being the best, offered a good trade off between feasibility and optimality.

- Smart object connectivity – this design choice is key for many deployments and installation. In fact, from this choice depend the constraints and the capabilities of the smart objects. Since, our demonstrator deals with mobiles, sensors and the internet, no single solution can be used. However, most of the machine to machine communication happens through wireless sensor networks.

- "Last mile" communication protocols – in order to interconnect smart objects we also have to decide which language they will speak. In order to support the maximum possible level of interoperability we chose to adopt the IoT-A protocol suite.



Topic	Design Choice	Impact on Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
Communication Confidentiality	DC9.1 No encryption	▼	▲	●	▲
	DC9.2 End-to-end Encryption	▲	▼	●	▼
	DC9.3 Hop-to-hop Encryption	▼	▼	●	●
	DC9.4 Onion routing-like encryption	▲	▼	●	▼
	DC9.5 Tunnelling	▲	▼	▲▼	▲▼
Bootstrapping	DC12.1 Static key pair	●	▲	▲	▼
	DC12.2 Updateable key pair	▲	▲	▲	●
	DC 12.3 Static shared secret	●	▲▼	●	▼
	DC 12.4 Updateable shared secret	▲	▲▼	●	●
	DC 12.5 Neighbour keys	●	●	▲	▼
	DC 12.6 Group Key	▼	▲	▲	▲
	DC 12.7 Transitive imprinting	●	▼	●	●
Smart object connectivity	DC16.1 Sensor and actuator networks	▼	▲▼	▲▼	▲
	DC16.2 RFID and smart tags	▲	▼	▼	▲
	DC16.3 WiFi connectivity	▲	▲	▲	▲
	DC16.4 Cellular network connectivity	▲	▲▼	▲	▲
"Last mile" communication protocols	DC17.1 IoT-A protocol suite	▲	▲	●	▲▼
	DC17.2 Ad hoc proprietary stack	▲▼	▲	●	▼
	DC17.3 Other standards not in the IoT-A protocol suite	▲▼	▲	●	▼

Table 5: Design choices relevant to scene 3. The highlighted choices are those picked for the demonstrator.

3.2.3.5 Step 5: Technical Realisation of the Demo

The demo setup consisted in a minipc acting as application and database server and as a connectivity gateway between the user IoT phone and the measuring instrument, a sensor node mimicking the behaviour of the measuring instrument and a mobile phone running the health care application through which the user could update her record.

Communications were configured according to WP3 prescription, namely 6LoWPAN and IPv6 were used for the network layer, CoAP and HTTP for the application in the constrained and the unconstrained networks, respectively.

Description	Communication
Blood glucose level reader	802.15.4
IoT phone	Wi-Fi
Gateway (minipc)	Wi-Fi + 802.15.4 + Ethernet

Table 6: Used Devices in implementation for scene 3

The telosb sensor nodes mimicking the blood glucose level reader were connected to the gateway via a 802.15.4 connection. While the IoT phone used a Wi-Fi connection to interact with the gateway.

The personal health care application and the remote server were running on the user mobile phone and the gateway, respectively. Figure 18 gives an overview.

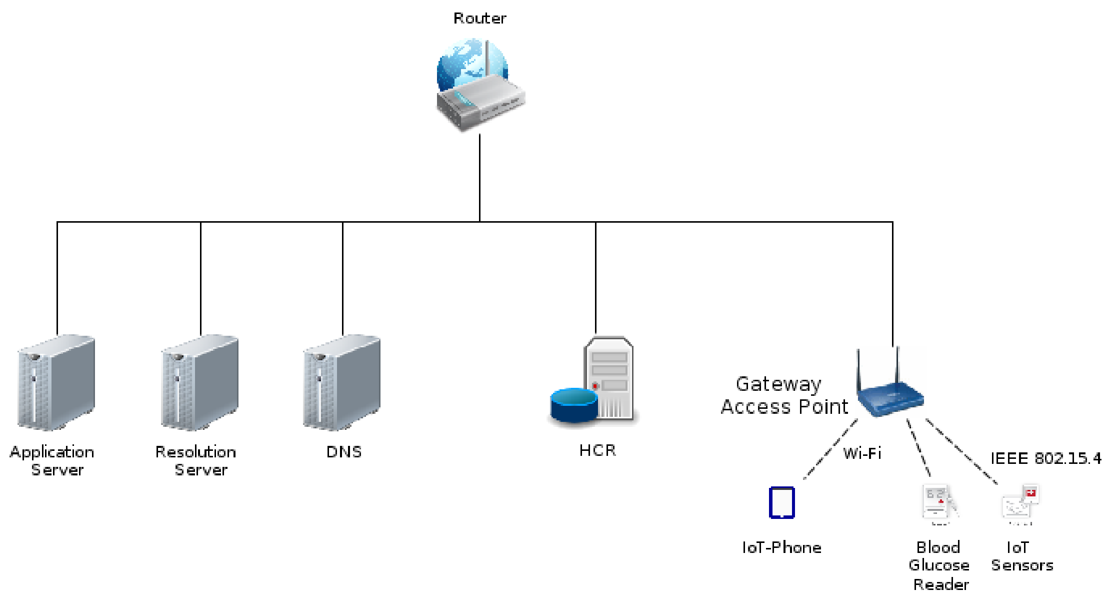


Figure 18: Scene 3 demonstrator setup

3.2.3.6 Feedback

Although the demonstrator of this scene has not been shown in Helsinki, comments from similar demonstrator can apply.

In particular, the audience often asked for services capable of instantiating a direct communication between Jane and the doctor in order to provide her with assistance during the measurement phase and advice in case something wrong is detected in her EHR.

3.2.4 Scene #4: Low Insulin Supply (link to Retail UC) (ALU-BE)

The implementation of this scene is based on the “Remote Patient Panic Event” health use case scene 5 depicted in D7.2.

3.2.4.1 Step 1: Application Description

The scene shows how the medicine supply of patients can be monitored and actions can automatically be taken in case the supply of medicines goes below a certain threshold. In the scene, the patient’s insulin drops below the threshold which triggers his doctor which will write an electronic prescription for more insulin. The insulin is then collected at the pharmacy by a care giver.

Each ampoule of insulin is tagged with a RFID tag. The ampoules are stored in a medicine cupboard which is also equipped with an RFID reader.

Robert takes the last ampoule of insulin out of his medicine cupboard. The system detects that the ampoule is removed and will update Robert’s EHR.

In case the EHR shows that this was the last ampoule of insulin and there are no more ampoules left in the cupboard, a notification to his doctor is sent so he can make a prescription for more insulin. He does that by updating the EHR of Robert with an electronic prescription. In Robert's EHR, Salomée is designated as his representative for collecting medicines. So she gets a notification to her IoT-Phone that a new subscription is available. She receives the prescription in an encoded form on her IoT-Phone. She goes to the pharmacy, buys the insulin and replenishes the insulin supply of Robert.

3.2.4.2 Step 2: Domain Model Representation of the Demo

In Figure 19, the Domain Model representation of the scene is depicted.

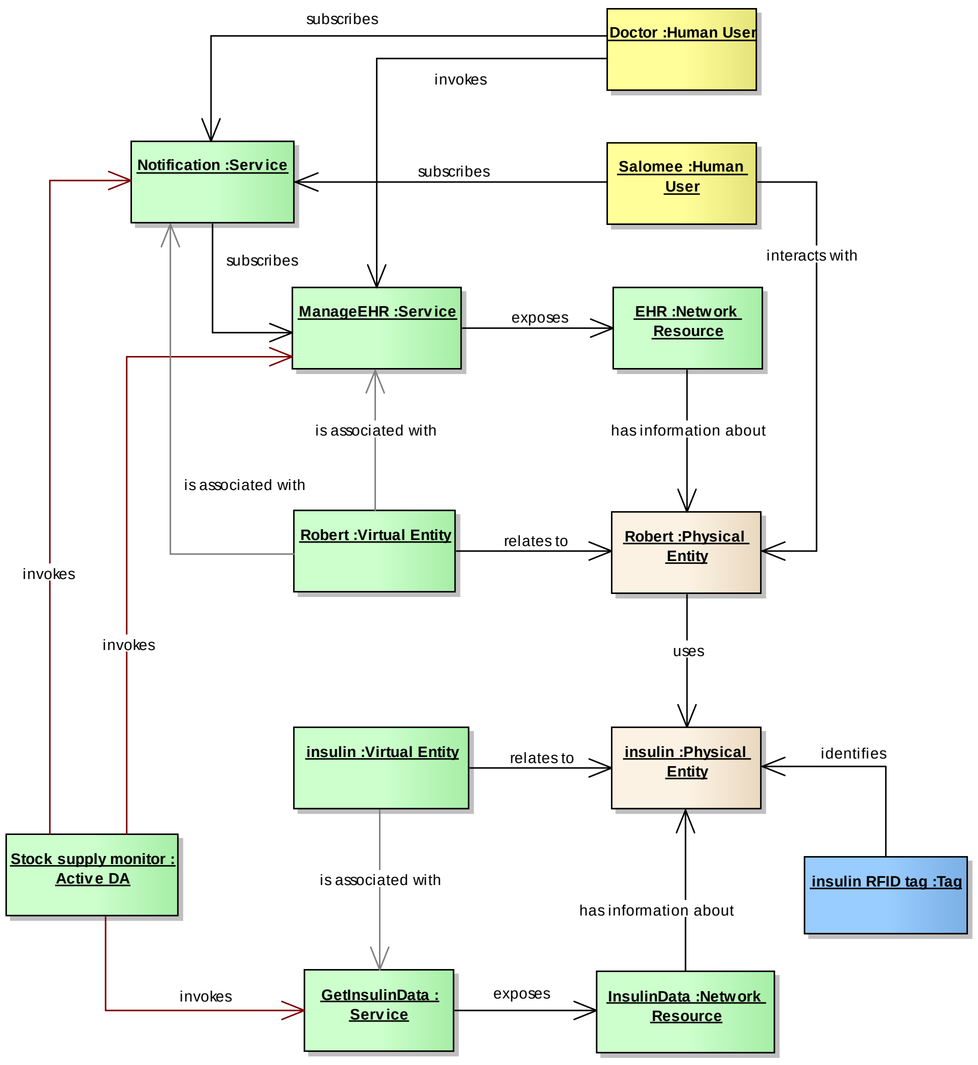


Figure 19: Domain Model of scene 4

Each ampoule of insulin is tagged with an RFID tag that identifies the Physical Entity of the insulin ampoule. The insulin is then represented in the digital domain with the insulin Virtual Entity.

The information about the insulin is stored in the Network Resource InsulinData that is accessed via the GetInsulinData Service.

The Stock supply monitor Active Digital Artefact invokes the GetInsulinData service to obtain more information on this Virtual Entity.

The Stock supply monitor Active DA will as well update the EHR of Robert in case a box of insulin is taken out of the medicine cupboard (not modelled).

The EHR is modelled as a Network Resource which is accessible via the EHR Service.

Part of the EHR is the amount of insulin that Robert still has available.

When the Stock supply monitor Active DA reads from the EHR that the amount of insulin has dropped below the threshold, the Notification service associated to Robert's Virtual Entity will be invoked. Robert's Doctor is subscribed to the Notification service and writes in the EHR of Robert a new prescription for insulin. Salomée, modelled as a Human User, is also subscribed to the Notification service and will receive the notification that a new prescription is available and that she needs to go for more insulin. She goes to the pharmacy, buys the insulin and the stock is replenished.

3.2.4.3 Step 3: Information Model Representation of the Demo

The mapping of the scene to the Information Model can be seen in Figure 20 below:

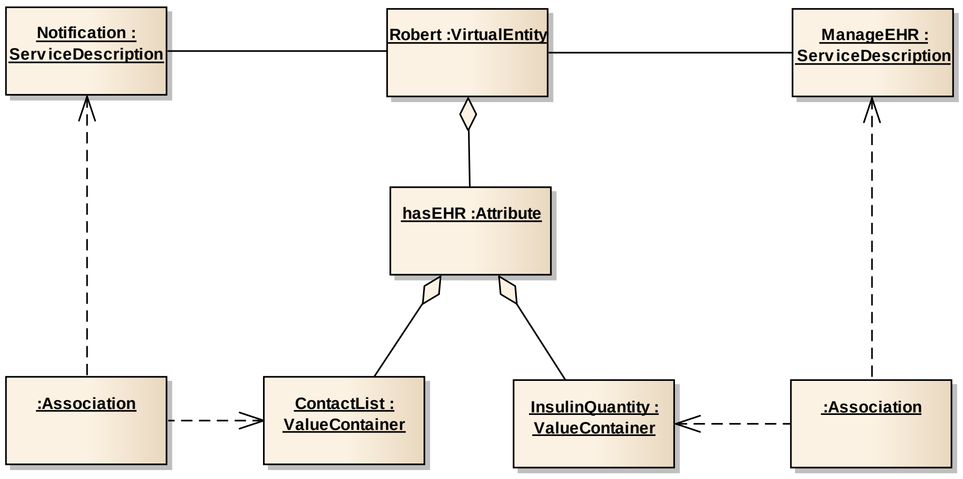

Notification : ServiceDescription
Robert :VirtualEntity
ManageEHR : ServiceDescription
hasEHR :Attribute
:Association
ContactList : ValueContainer
InsulinQuantity : ValueContainer
:Association

Figure 20: Information Model of scene 4 - Low Insulin Supply

Robert is a Virtual Entity for which in this scene the EHR is the most important. So the Virtual Entity has an attribute EHR with two Value Containers: the contact list (where the name of Salomee will be present) and the quantity of insulin still in possession of Robert. The Virtual Entity of Robert has as well associations to the notification service and the service that manages the EHR.

3.2.4.4 Step 4: Relevant Design Choices for Implementation

In order to build a demonstrator for the scene we took into consideration many aspects that do not have straightforward solutions, in fact under many implementation fields it is possible to find issues that can be solved in different ways and, while a given solution may result to be the best under a particular view, it is also possible that other solutions outperform the former adopting different views.

For scene 4 we consider the following design choices, which are also shown in Table 7.

- VE Resolution – handles functions needed for handling with resolution, monitoring, and storage of history of the virtual entity.

- Storage of Information History – Information that is been gathered from IoT Resources can be cached for later further processing. The information can be stored locally, remotely or both.

Topic	Design Choice	Impact on: Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
VE Resolution	DC3.1 VE Resolution with mandatory security	▲▼	●	▲	●
	DC3.2 VE Resolution with optional security	▼	●	▼	●
	DC3.3 VE Resolution with QoS	●	●	▲	●
	DC3.4 VE Resolution domain-oriented	▲	▲	▲	▲
	DC3.5 VE Resolution location-oriented	▼	▲	▲▼	▲▼
	DC3.6 Resolution semantic Web-oriented	●	●	▲	▲▼
	DC3.7 Resolution Peer-to-Peer-oriented	●	▲	▲	●
	DC3.8 Resolution Federation-based	●	▲	▲	●
Storage of information history	DC11.1 Storage of history locally	▲	▼	▼	●
	DC11.2 Storage of History remotely	▲▼	▲	▲	●
	DC11.3 Storage of History locally and remotely	▲▼	▲	▲	●

Table 7: Design choices relevant for scene 4. The highlighted choices are those picked for the demonstrator.

3.2.4.5 Step 5: Technical realisation of the Demo

The demo setup consists out of a number of servers, gateways and end devices with different communication technologies. The demonstrator application is running on the application server in the network and uses the WP4 resolution server for discovery, lookup of the services needed for scene 1. On the IoT phone a local eHealth demonstrator application is running which exposes services to the resolution network. Figure 21 gives an overview.

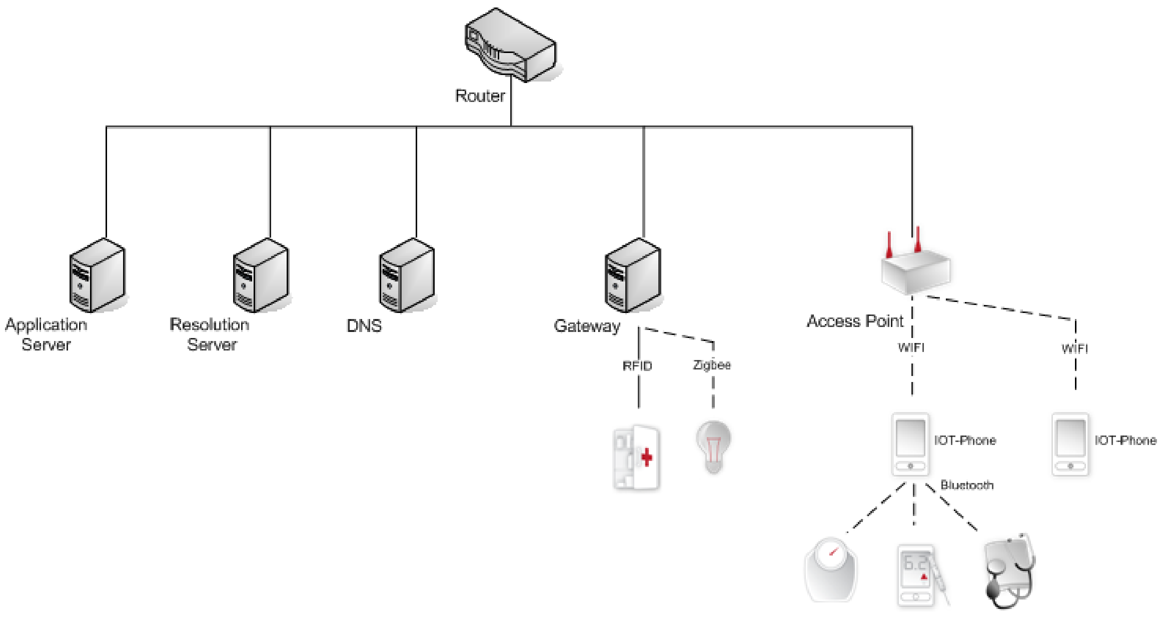


Figure 21: Physical setup for implementation of scene 4

Description	Communication
IoT Phone	Wifi
Medical Cabinet	Rfid Reader
Gateway	WiFi + Rfid

Table 8: Used Devices in implementation for scene 4

The IoT phone is connected to the network over WiFi.

Communications on the network layer are inline with WP3 configuration prescription, namely IPv6. On the application layer HTTP rest and CoAP is used.

An overview of the physical implementation is shown in Figure 22 for scene 4.

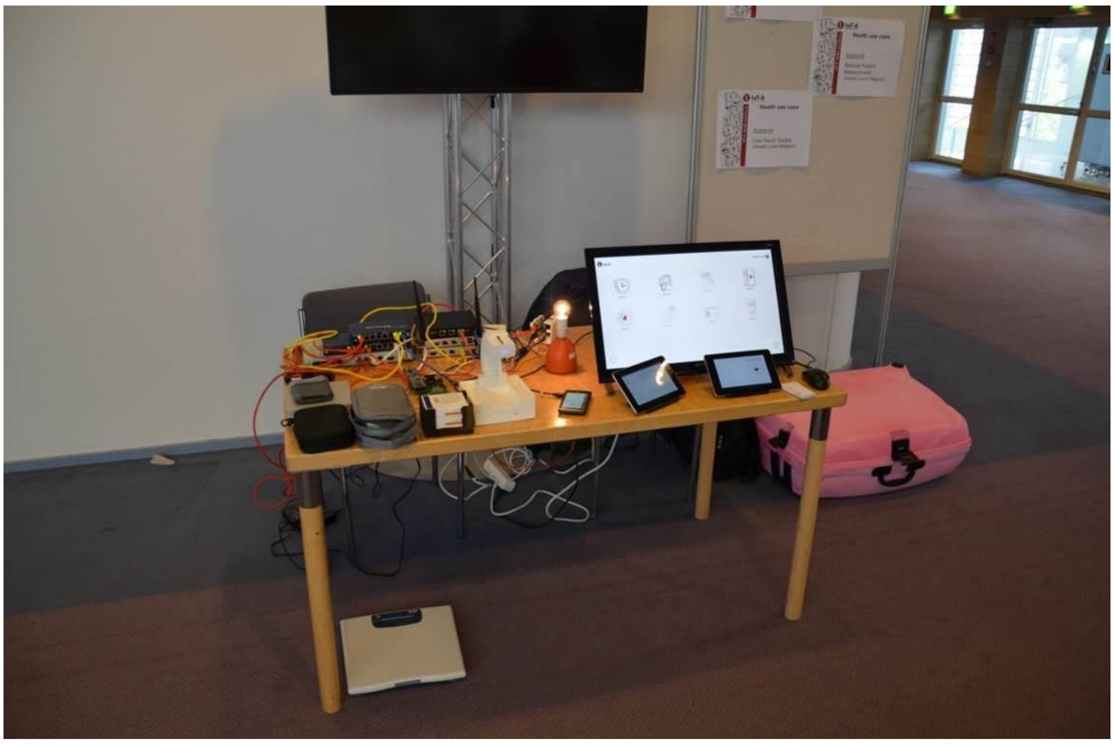


Figure 22: Demo setup at IoT week 2013, Helsinki

3.2.4.6 Feedback

The demonstrator was used to explain the practical applications of the abstract IoT-A concepts – in particular the IoT-A resolution service and concepts (WP4) and internet of things protocol of WP3.

The spectators have always viewed the three demo described. The most technical remark or questions was on the service layer architecture (WP4). Non-technical is more about legal issues related to the eHealth domain.

3.2.5 Scene #5: Accident and hospitalisation: car accident (CFR)

The implementation is based on use case I: Health and Home scene 5. This demonstrator illustrated how automatic emergency operations and alarms can be realised through IoT communication and services. In particular, three different sensors have been used, accelerometers, heart beat measure and breath rate measure. The output of these three devices was combined to detect sudden danger events, such as a car accident. In such an unfortunate event, the system tries to verify the responsiveness of the user through a notification, which, if not stopped before a given timeout, raises an emergency call towards the nearest ER.

3.2.5.1 Step 1: Application Description

Robert's IoT phone is associated to an alarm generating service triggered by abnormal reading of the IoT phone accelerometers. Upon the reception of an alarm, the service interrogates all the other vital parameter sensors associated with Robert.

Only when both vital parameter and accelerometer readings indicate that an accident might have happened a notification is visualised on the IoT phone.

In order to prevent the system to raise false alarm a guard interval is conceded to the user to stop the emergency procedure. Hence, if the user is able to react before a timeout, his condition are supposed to be good enough to avoid the emergency call, in the negative case an automatic emergency call is sent to the nearest ER.

The nearest ER is found thanks to the resolution service.

3.2.5.2 Step 2: Domain Model Representation of the Demo

The domain model for the use case is shown in Figure 23 and Figure 24: the former portrays the relationships among Robert's Virtual Entity, Resources, Devices and Services, while the latter focuses on the role of Human Users in the scene.

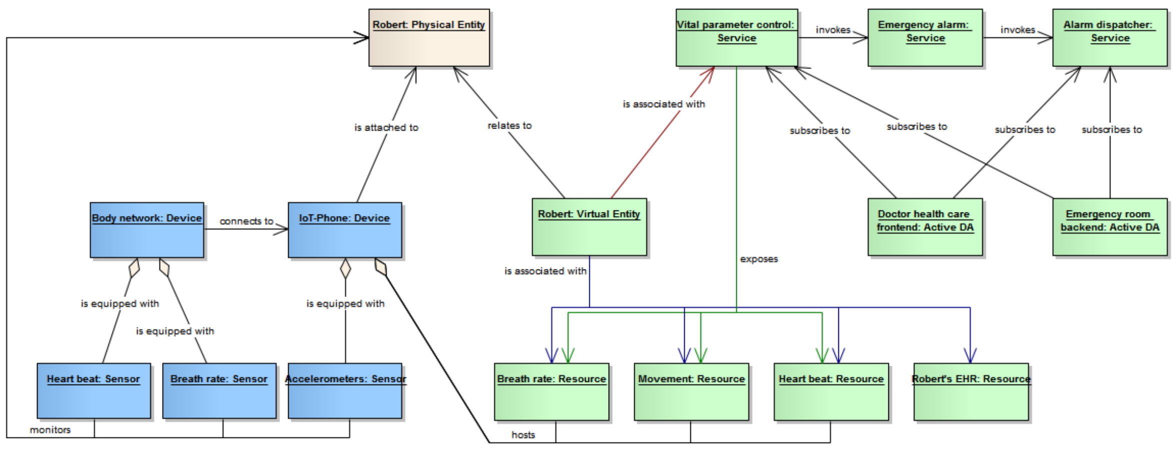


Figure 23: Domain Model of scene 5 part A

The first figure consists of two parts: the devices are drawn in blue, while software is shown in green. Of the devices, the main one is the IoT-Phone which is equipped with movement sensor and is connected to Robert's body network, which, in turn, has breath rate and heart beat sensors.

On the software part, the vital parameter control service is the most important since it exposes and monitors Robert's vital parameters resources.

These resources are: his movement, his breath rate and his heartbeat, which are hosted on the IoT-Phone device. A fourth resource is exposed by the vital parameter control service: Robert's EHR which keep track of Robert's medications and health history.

Finally, two additional services are used: the first, the emergency alarm service, is needed to generate the alarm report from Robert's vital parameter control service information, while the second, the alarm dispatcher, is in charge of locating the closest assistance centre, either a hospital or a clinic, capable of providing Robert with the adequate level of assistance.

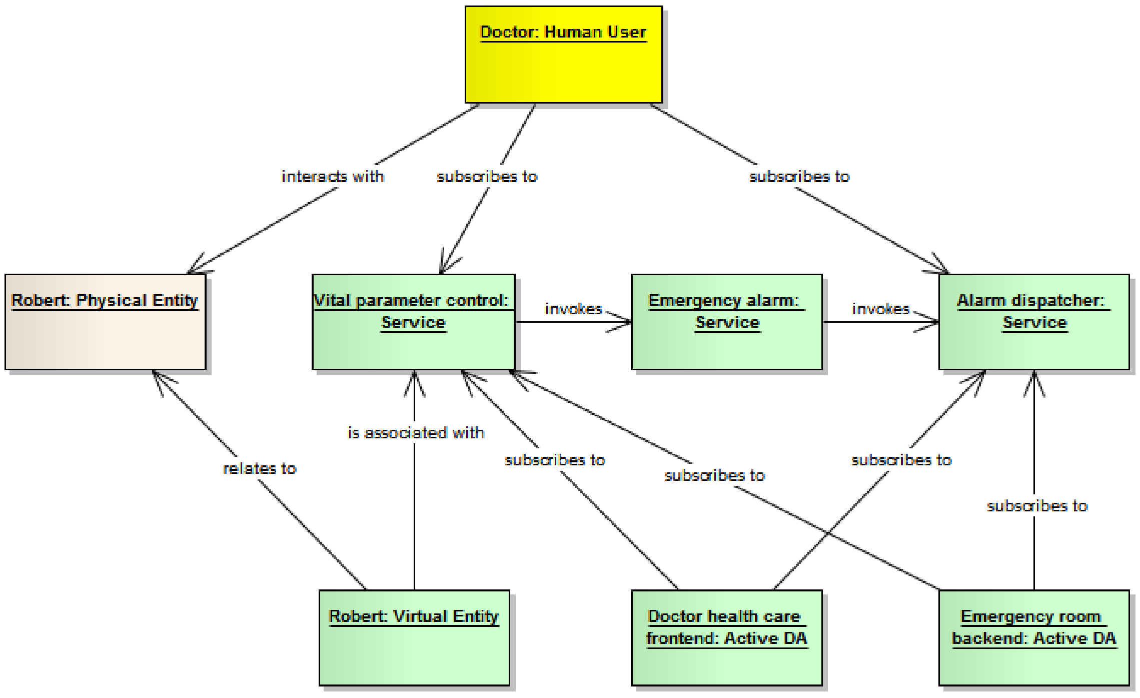


Figure 24: Domain Model of scene 5 part B

The second part of the domain model illustrates the role of the doctor and the digital artefacts used to enable the doctor and the emergency room to promptly react to Robert's accident and prepare his hospitalisation if needed.

3.2.5.3 Step 3: Information Model Representation of the Demo

Figure 25 shows the information model specialised to Robert's accident scene. In the upper part of the figure Robert's virtual entity is described with three attributes: his movement, his breath rate and his heartbeat. Each of the attributes is connected to a specific container which is, in turn, connected with data values.

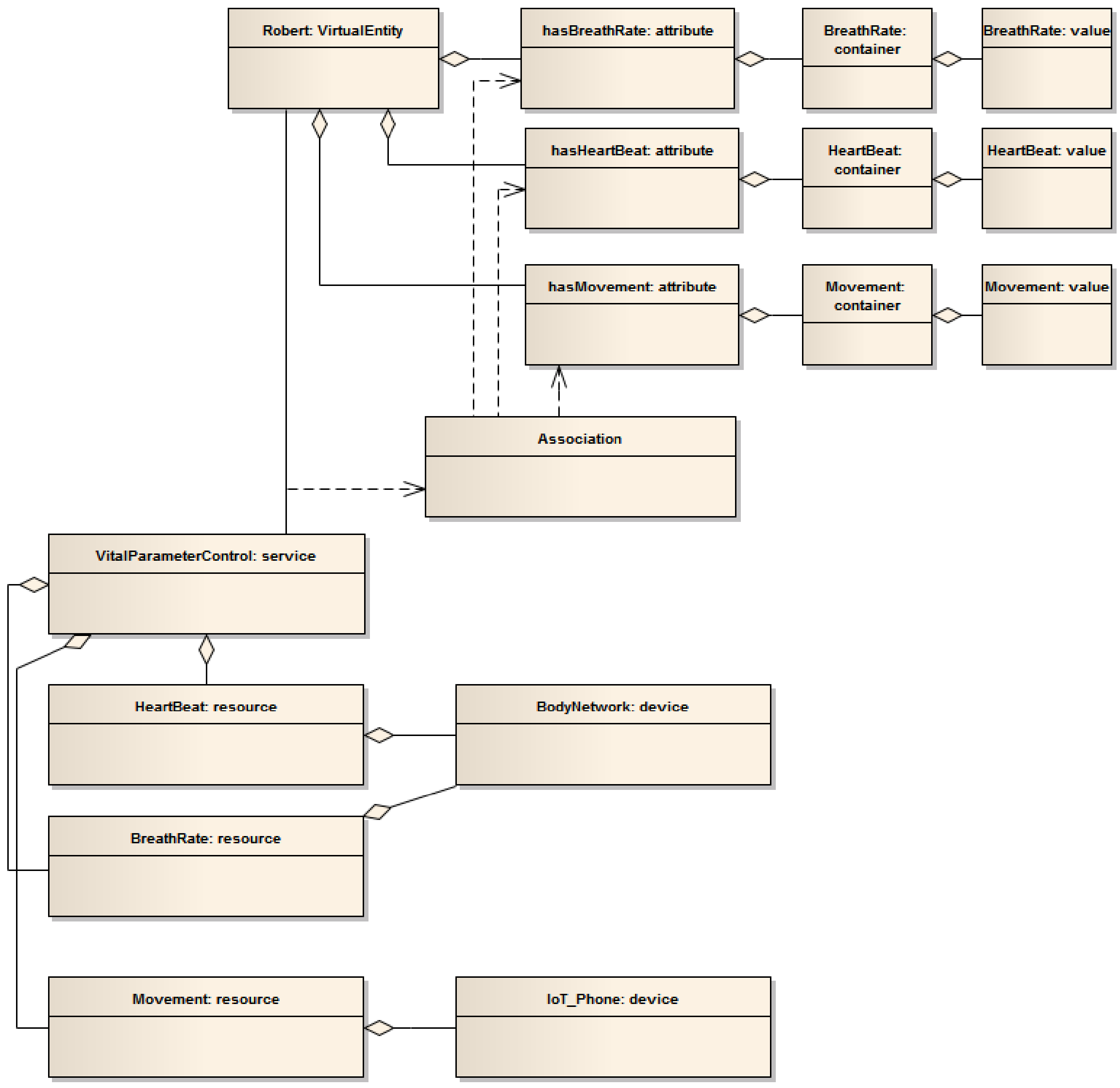


Figure 25: Information Model of scene 5

In the bottom part of the figure the vital parameter control service is shown along with its resources: the heart beat and the breath rate resources are connected to Robert's body network, while Robert's movement resource is connected to the IoT-Phone.

3.2.5.4 Step 4: Relevant design choices for Implementation

This scene has been implemented with the same criteria of scene 3 of the health use case thus the motivation behind the most relevant design choice can be read in Section 0. Here, we reported the illustration (see Table 9) in order to summarise the process at glance.

Topic	Design Choice	Impact on			
		Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
Communication Confidentiality	DC9.1 No encryption	▼	▲	●	▲
	DC9.2 End-to-end Encryption	▲	▼	●	▼
	DC9.3 Hop-to-hop Encryption	▼	▼	●	●
	DC9.4 Onion routing-like encryption	▲	▼	●	▼
	DC9.5 Tunnelling	▲	▼	▲▼	▲▼
Bootstrapping	DC12.1 Static key pair	●	▲	▲	▼
	DC12.2 Updateable key pair	▲	▲	▲	●
	DC 12.3 Static shared secret	●	▲▼	●	▼
	DC 12.4 Updateable shared secret	▲	▲▼	●	●
	DC 12.5 Neighbour keys	●	●	▲	▼
	DC 12.6 Group Key	▼	▲	▲	▲
	DC 12.7 Transitive imprinting	●	▼	●	●
Smart object connectivity	DC16.1 Sensor and actuator networks	▼	▲▼	▲▼	▲
	DC16.2 RFID and smart tags	▲	▼	▼	▲
	DC16.3 WiFi connectivity	▲	▲	▲	▲
	DC16.4 Cellular network connectivity	▲	▲▼	▲	▲
"Last mile" communication protocols	DC17.1 IoT-A protocol suite	▲	▲	●	▲▼
	DC17.2 Ad hoc proprietary stack	▲▼	▲	●	▼
	DC17.3 Other standards not in the IoT-A protocol suite	▲▼	▲	●	▼

Table 9: Design choices relevant to scene 5. The highlighted choices are those picked for the demonstrator.

3.2.5.5 Step 5: Technical realisation of the Demo

The demo setup consisted in a minipc acting as a connectivity gateway between the user IoT phone and the vital parameter readers, two sensor nodes mimicking the behaviour of the vital parameter measuring instruments and a mobile phone running the health care application in charge of monitoring the accelerometers on the device and comparing the reading with those of the other sensors.

Communications were configured according to WP3 prescription, namely 6LoWPAN and IPv6 were used for the network layer, CoAP and HTTP for the application in the constrained and the unconstrained networks, respectively.

Description	Communication
Hearth beat rate sensor	802.15.4
Breath rate sensor	802.15.4
IoT phone	WiFi + 3G
Gateway (minipc)	WiFi + 802.15.4 + Ethernet

Table 10: Used Devices in implementation for scene 5

The telosb sensor node mimicking the vital parameter readers were connected to the gateway via a 802.15.4 connection. While the IoT phone used a WiFi connection to interact with the gateway and a 3G connection to forward the emergency call towards the ER.

The personal health care application was running on the user mobile phone and the gateway, respectively.

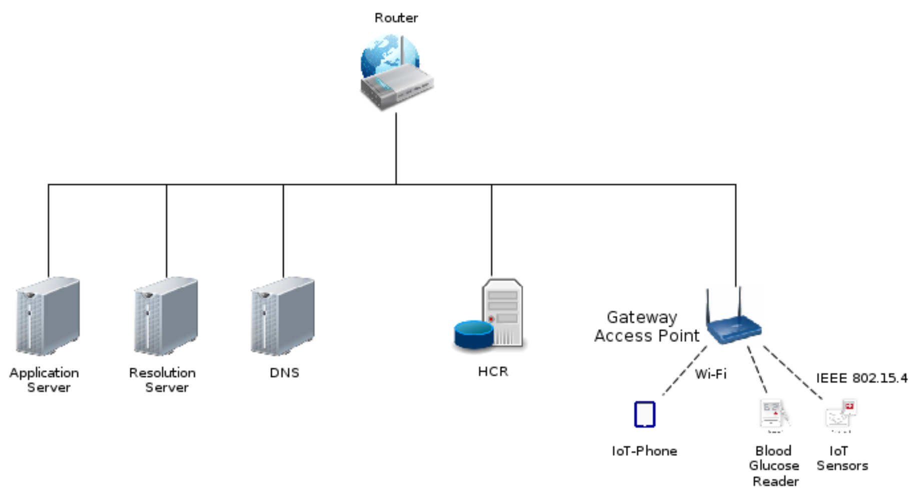


Figure 26: Scene 5 demonstrator setup

3.2.5.6 Feedback

During the IoT week 2013 in Helsinki this scene received quite a few comments about the network interactions: in particular, since Robert's accident may happen in areas where only cellular networks are available it may be of interest to study the impact of sending the alarm through the standard or the emergency network or through data connections.

In addition, since the notification from the IoT-Phone may be missed by Robert, it could be useful to let the emergency room make a call to Robert before sending the ambulance, in order to avoid false positives.

3.2.6 Scene #6: Expedited Checking into a Hospital (HSG)

The implementation of this scene is based on the "Expedited Checking into a Hospital" health use case scene 7 defined in D7.2.

3.2.6.1 Step 1: Application Description

This scene shows how ubiquitous sensors in consumer goods - such as the IoT-Mouse (a mouse with multiple sensors integrated) - when paired with the Internet of Things Architecture (IoT-A), could speed up hospital check-in and make initially acquired information available in later applications. In addition to fast hospital check-in, the application saves personnel time & improve data accuracy vs. manual entry.

In particular, unlike many scenes which focus on machine-to-machine interaction, the scene shows what a direct human-to-machine interaction with IoT-A would be like. As such, the overlap between the existing internet and the future internet of things is shown.

Our prototype also serves as an education tool to highlight concepts from the IoT-A project – namely the resolution infrastructure in WP4 – with our technology such as the IoT-A Mouse.

3.2.6.2 Step 2: Domain Model Representation of the Demo

The domain model for the use case is shown in Figure 27.

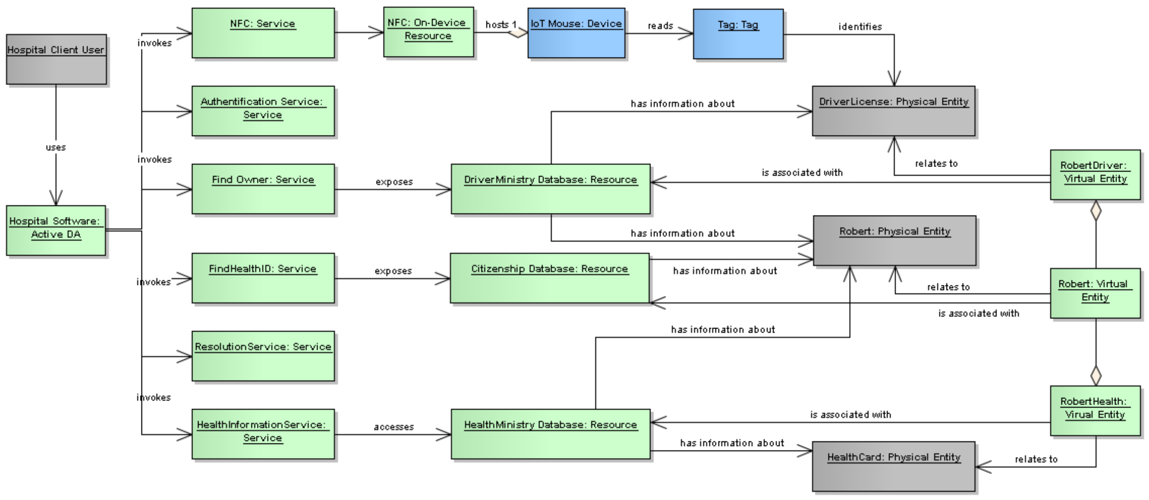


Figure 27: Domain Model of scene 6

The hospital software acts as an active digital artefact which invokes sequentially a series of services to eventually check a person into a hospital. First, the clerk reads driver's license card (a tag) with a sensor embedded in a mouse. The hospital is authenticated and authorised to retrieve further information about the driver. The driver identifier refers to the person's virtual representation as a driver (the driver virtual entity) and is passed into the resolution service to discover services associated with the driver. The resolution service retrieves associations, which tell the hospital software what services associated with the driver are available and where they are located. Subsequently, the findOwner service is called, which exposes the ownerID from a resource. The ownerID refers to another virtual representation of the person, the owner virtual entity. Once again, this ID is given to the resolution service to discover services associated with the owner virtual entity. This reveals a service called findHealthID. When this service is called, we get back the healthID of the person, which points to their virtual representation in the health domain. Once more, this ID is given to the resolution service to find our last service, the one that retrieves the medical folder of the patient, thereby checking him in.

3.2.6.3 Step 3: Information Model Representation of the Demo

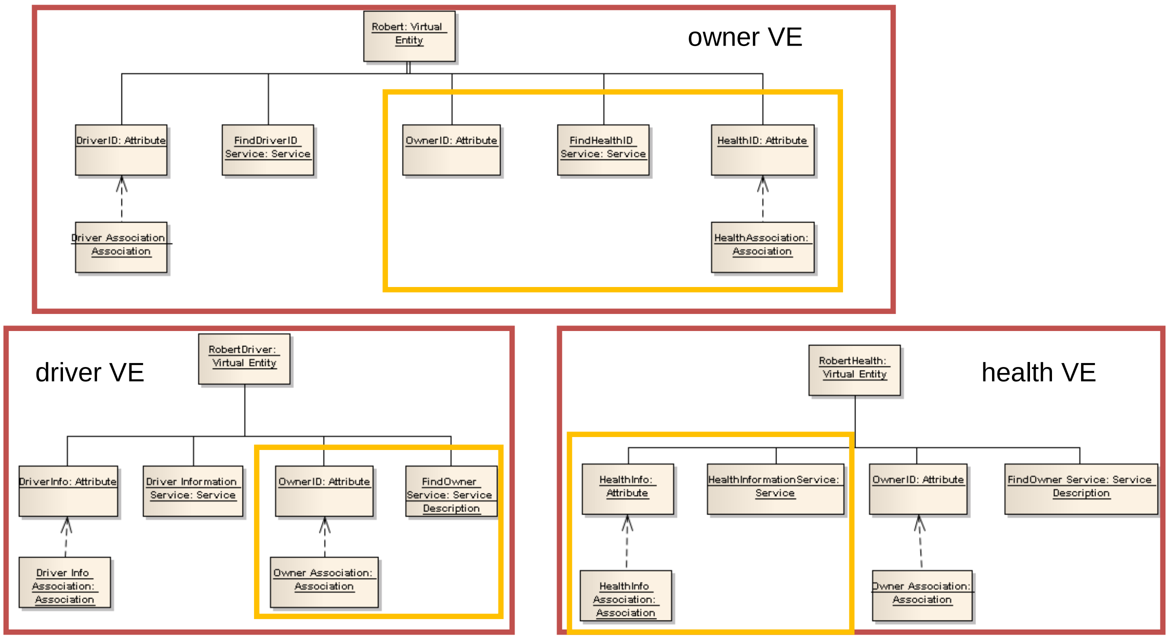

Robert: Virtual Entity
owner VE
DriverID: Attribute
FindDriverID Service: Service
OwnerID: Attribute
FindHealthID Service: Service
HealthID: Attribute
Driver Association: Association
HealthAssociation: Association
driver VE
RobertDriver: Virtual Entity
DriverInfo: Attribute
Driver Information Service: Service
OwnerID: Attribute
FindOwner Service: Service Description
Driver Info Association: Association
Owner Association: Association
RobertHealth: Virtual Entity
health VE
HealthInfo: Attribute
HealthInformationService: Service
OwnerID: Attribute
FindOwner Service: Service Description
HealthInfo Association: Association
Owner Association: Association

Figure 28: Information Model of scene 6, depicting the three Virtual Entities; boxes in Yellow indicate Associations and Services used in the scene

In the information model, there are three virtual entity representations of Robert: RobertDriver VE, Robert VE, and RobertHealth VE. Each Virtual Entity has an identifier (driverID, ownerID and healthID respectively). In RobertDriver VE and RobertHealth VE, both have the ownerID as attributes and a service that exposes this ID. In the Robert VE (which can be thought of as a “master” virtual entity), the identifiers of all other VEs of Robert are known.

Therefore, starting from either an identifier for RobertDriver VE or RobertHealth VE, it is possible to use the Resolution infrastructure to discover the service which exposes the ownerID pointing to Robert VE. In turn, once the ownerID is known, because RobertVE contains all other VE IDs and the services which expose the identifiers, it is also possible to eventually find and execute the services of other VEs.

3.2.6.4 Step 4: Relevant Design Choices for Implementation

For the IoT system described in this scene, there were different design choices possible depending on the view or perspective taken in the architecture. Since the core idea of this demo scene was on the concept of multiple identifiers, rigorous security was not the main focus; we describe the design choices made specifically for the demo, and where appropriate, we note when a different design might be considered for a real world hospital implementation. For ease of visualisation, the overall design choices are presented in Table 11.

- VE Resolution – In a real-world hospital environment, it is likely that the resolution of virtual entities would have some security policies depending on which client is requesting what information about a particular VE. Since these decisions are left to the individual owners of a VE and their associated resource, the use case reflects the design choice “VE Resolution with optional security”; the initial resolution of the driver ID is subject to authentication and authorisation, while other resolution processes in the use case are not.

- IoT Service Resolution – In a real-world deployment, in particular for the domains of driving and health, the services that can be resolved would probably be restricted depending on the security and privacy policies of the service provider; for example, which services that can be found (associated with a particular VE) might depend on which client is making the query. At the time of the demo deployment, such a feature was not available in the IoT service resolution, as such in our scene the IoT Service Resolution has optional security; i.e. for a given service specification, the IoT Service Resolution will discover all services, without filtering results based on the IoT Services' S&P policies.

- IoT Services provides an interface to IoT users by utilising capabilities of IoT Resources. In the demo, security policies for the services are optional. In a real-world deployment, the services which access patient information in particular are recommended to use mandatory security IoT Services.
- Service Access Control – one possible security policy in a real-world implementation would be service access control – i.e. only those authenticated and authorised could use a particular service. We use an authentication based service access once in the demo, when the hospital client makes the first query via the IoT Resolution service to resolve the driver ID of the patient. In a real-world deployment, it would be recommended to combine this with policy-based service access for all services.

Topic	Design Choice	Impact on	Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
VE Resolution	DC 3.1 Resolution with mandatory security		▲▼	●	▲	●
	DC 3.2 VE Resolution with optional security		▼	●	▼	●
	DC 3.3 VE Resolution with QoS		●	●	▲	●
	DC 3.4 VE Resolution domain-orientated		▲	▲	▲	▲
	DC 3.5 VE Resolution location-orientated		▼	▲	▲▼	▲▼
IoT Service Resolution	DC 4.1 IoT Service Resolution with mandatory security		▲▼	●	▲	●
	DC 4.2 IoT Service Resolution with optional security		▼	●	▼	●
IoT Service	DC 4.3 IoT Service with mandatory security		▲▼	●	▲	●
	DC 4.4 IoT Service with optional security		▼	●	▼	●
Service Access Control	DC6.2 Unrestricted access to service		▼	▲	▼	▲
	DC6.2 Authentication based service access		▲▼	▲▼	▲▼	▲▼
	DC6.3 Policy-based service access		▲	▲▼	▲▼	▼

Table 11: Design choices relevant to scene 6. The highlighted choices are those picked for the demonstrator.

3.2.6.5 Step 5: Technical realisation of the Demo

In this scene, a person enters the hospital and wants to check-in. Unfortunately he does not have any health identification available. Instead, we use the IoT-Mouse – a mouse with multiple sensors – to read other identification belonging to the person. The sensor reads a tag on the identification and extracts an identifier. The IoT Resolution Service is called to determine to whom the identifier points to. A separate health service is then called by the hospital client, which requests permission for and obtains related information to check the person in.

The setup, as described above, is represented in Figure 29.

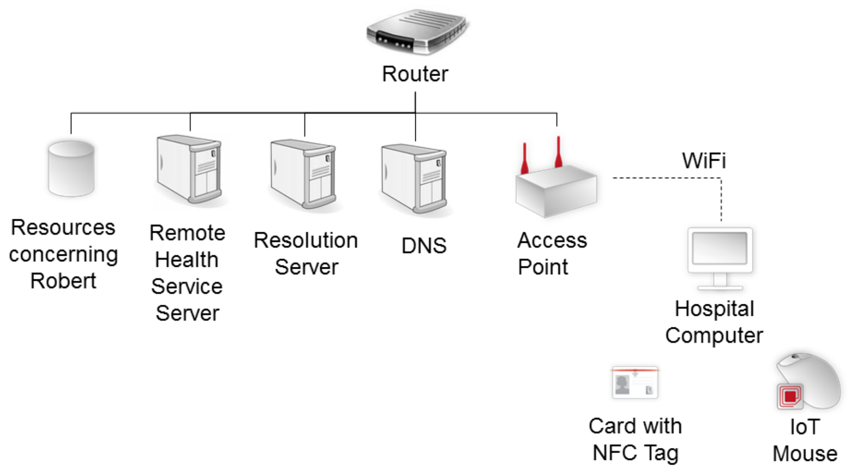


Figure 29: Physical Setup of the Expedited Hospital Check-In

3.2.6.6 Feedback

During IoT Week 2013 in June, in line with our goal discussed earlier, our demonstrator was used to explain the practical applications of the abstract IoT-A concepts – in particular the IoT-A resolution service and concepts of multiple identities of WP4.

Visitors who saw the demo understood intuitively the concepts of multiple identities and the idea of the resolution service. There was interest in the design choices behind the demo, in particularly with respect to security and privacy. To visitors, we explained that as it was a demo, rigorous security was not needed; in discussion with one visitor, we considered what options might be necessary in a real deployment, and concluded that authentication over encrypted channels would be a necessary choice. There were also discussions with visitors about authentication-based service access (our current design choice) and the possibility of policy-based service access. From these discussions, we can infer that the public has a strong interest in the security aspects of IoT systems, and that highly relevant future work after the project should address these aspects.

In conclusion, the demo fulfilled its goal of educating the public about the concepts of IoT-A, starting a dialog about IoT-A activities, and also in eliciting interest in future extensions of IoT-A in real life applications.

3.2.7 Scene #7: Patient safety in the operating theatre

This scene was provided by a stakeholder and expands the defined scenario out of D7.2. The scenario was included by Prof. Christoph Thümmler, who is actively contributing in the eHealth area. The specific application was implemented with help of the MUNICH platform by Celestor, Napier University Edinburgh, Technical University of Munich and Siemens.

3.2.7.1 Application Description

This use case scene is about counting of 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 (the human body). Therefore each towel is fitted with a RFID tag, to be able to track it within the surgery. Figure 30 shows an on-going surgery with the blue stomach towels.

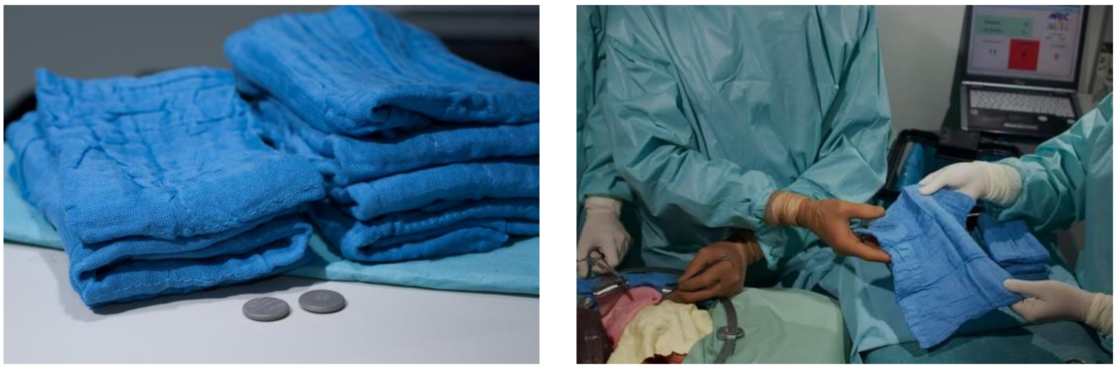


Figure 30: The scene of tracking towels in surgery

The RFID-tagged towels may be tracked by three antennas from different positions in the operational 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 which are put into the patient's body are "in use". Finally, towels which are not needed any more after the surgery are put into the towel container and put into the state "used".

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 waste bin.

From a business perspective up to 100 stomach towels may be used within a single surgery. Towels remaining in the patient's abdomen may cause severe and even fatal infections. As there are no official numbers, e.g. no central databases on overseen towels within a patient's body the numbers differ. Studies state 6.000-9.000 incidents per year.

3.2.7.2 Domain Model Representation of the Demo

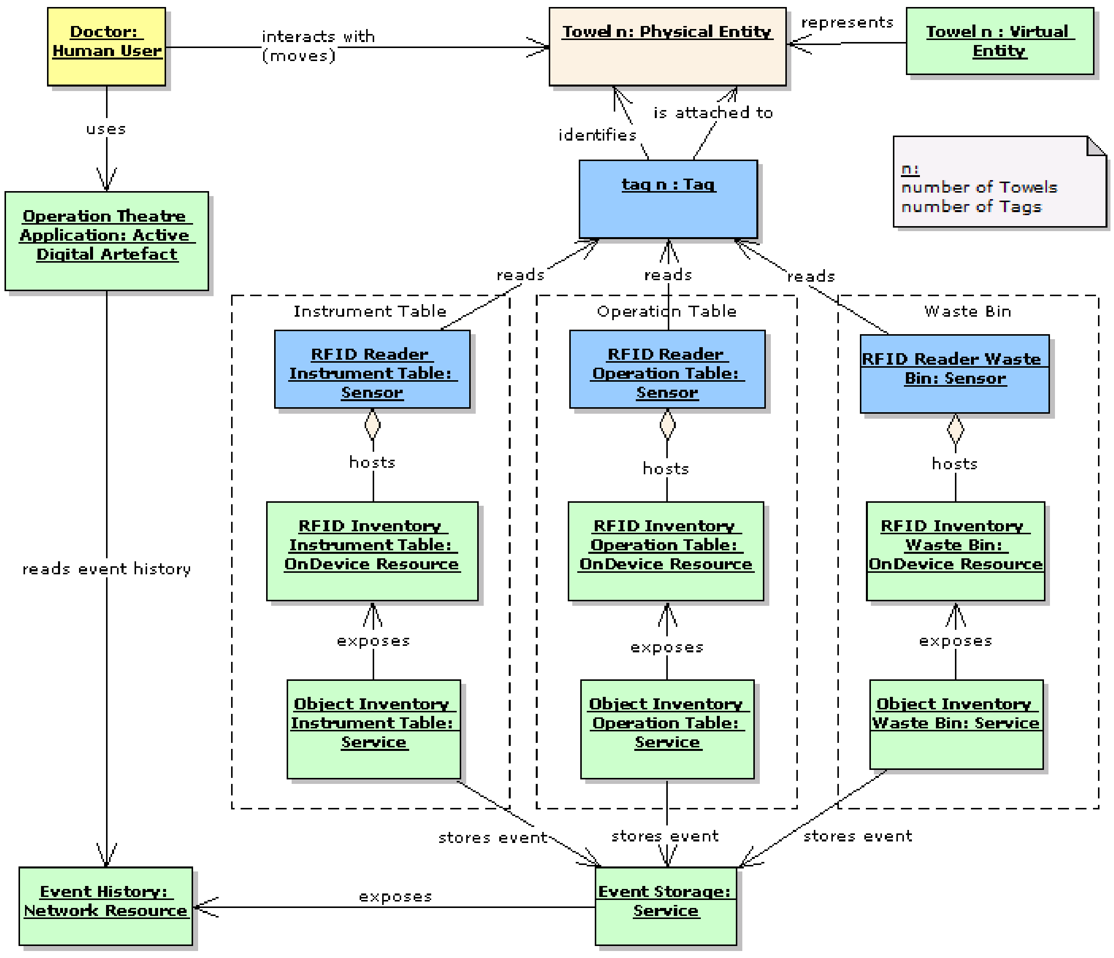

Doctor: Human User
interacts with (moves)
Towel n: Physical Entity
represents
Towel n : Virtual Entity
uses
identifies
is attached to
tag n : Tag
n: number of Towels number of Tags
Operation Theatre Application: Active Digital Artefact
reads
reads
reads
Instrument Table
Operation Table
Waste Bin
RFID Reader Instrument Table: Sensor
RFID Reader Operation Table: Sensor
RFID Reader Waste Bin: Sensor
hosts
hosts
hosts
reads event history
RFID Inventory Instrument Table: OnDevice Resource
RFID Inventory Operation Table: OnDevice Resource
RFID Inventory Waste Bin: OnDevice Resource
exposes
exposes
exposes
Object Inventory Instrument Table: Service
Object Inventory Operation Table: Service
Object Inventory Waste Bin: Service
stores event
stores event
stores event
Event History: Network Resource
exposes
Event Storage: Service

Figure 31: Domain Model of scene 7

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’. The user checks only if no towel is 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 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, Waste Bin) that are modelled as Sensor devices. Each of the Sensors hosts an OnDevice Resource that exposes an ‘Object Inventory Service’. These services store events by invoking the ‘Event Storage Service’ that is exposed by the Network Resource ‘Event History’. This Resource is also exposed to the ‘Operation Theatre Application’ by the Event History.

3.2.7.3 Information Model Representation of the Demo

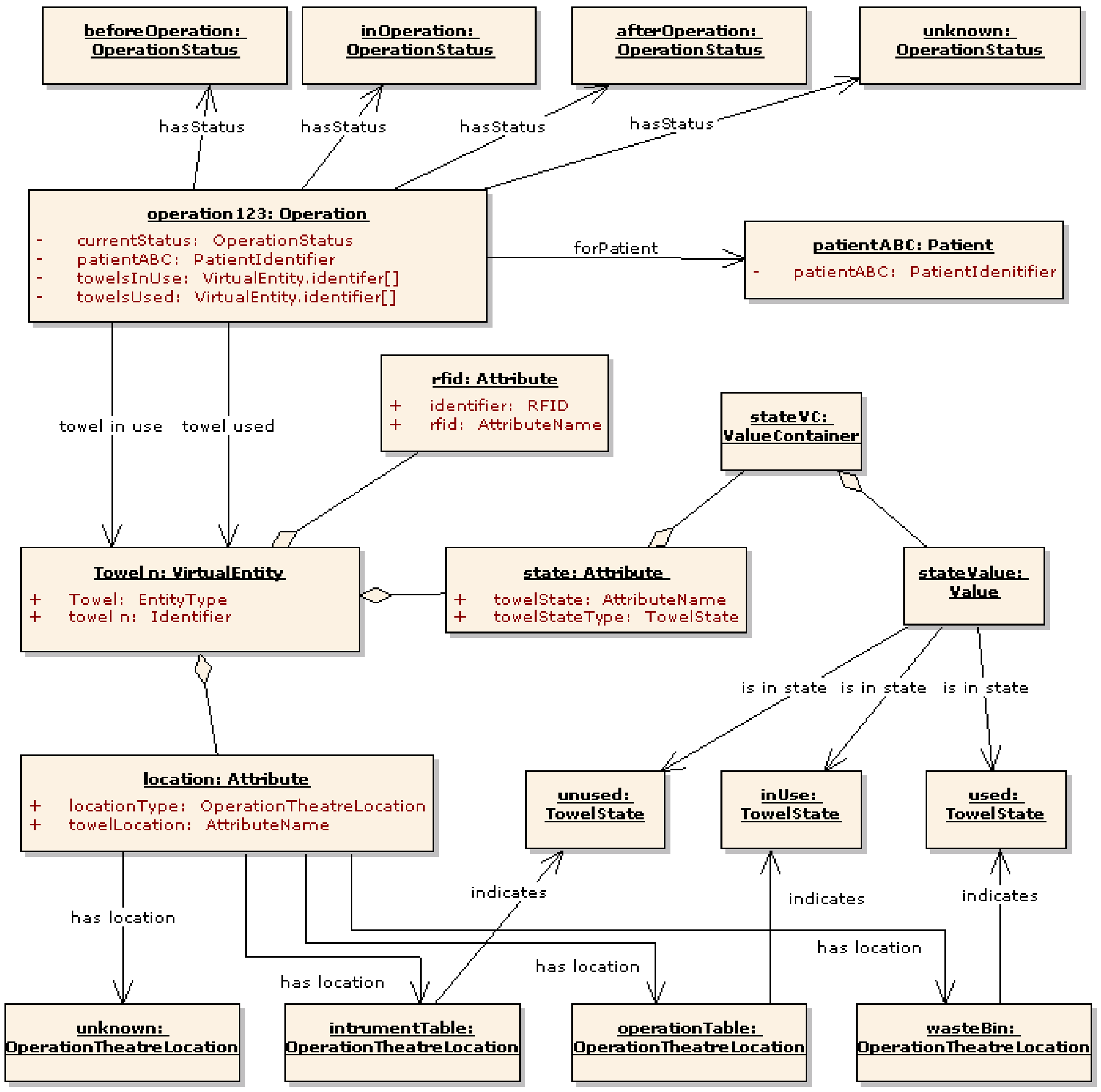

beforeOperation: OperationStatus
inOperation: OperationStatus
afterOperation: OperationStatus
unknown: OperationStatus
hasStatus
hasStatus
hasStatus
hasStatus
operation123: Operation
- currentStatus: OperationStatus
- patientABC: PatientIdentifier
- towelsInUse: VirtualEntity.identifer[]
- towelsUsed: VirtualEntity.identifier[]
forPatient
patientABC: Patient
- patientABC: PatientIdenitifier
rfid: Attribute
+ identifier: RFID
+ rfid: AttributeName
stateVC: ValueContainer
towel in use
towel used
Towel n: VirtualEntity
+ Towel: EntityType
+ towel n: Identifier
state: Attribute
+ towelState: AttributeName
+ towelStateType: TowelState
stateValue: Value
is in state
is in state
is in state
location: Attribute
+ locationType: OperationTheatreLocation
+ towelLocation: AttributeName
unused: TowelState
inUse: TowelState
used: TowelState
indicates
indicates
indicates
has location
has location
has location
has location
unknown: OperationTheatreLocation
intrumentTable: OperationTheatreLocation
operationTable: OperationTheatreLocation
wasteBin: OperationTheatreLocation

Figure 32: Information Model of scene 7

The Information Model specified for this use case (see Figure 32) also addresses relationships between entities not depicted in the Domain Model before but appearing in the Business Process Model. 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 in the Information Model specifying Domain Attributes that are essential for the use case. The towel’s identifier corresponding to the attached 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 operation theatre are reflected in the Information Model as Attribute of the Towel VE. For simplification the allowed values for this attribute {InstrumentTable; OperationTable; WasteBin; unknown} are not visualised as ValueContainer, but should be seen as those. The OperationTheatreApplication is then 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’}.

3.2.7.4 Technical realisation of the Demo

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 running the application. The MUNICH-platform depicted in Figure 33 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 number of towels in their respective range by invoking the ‘Event Service’.

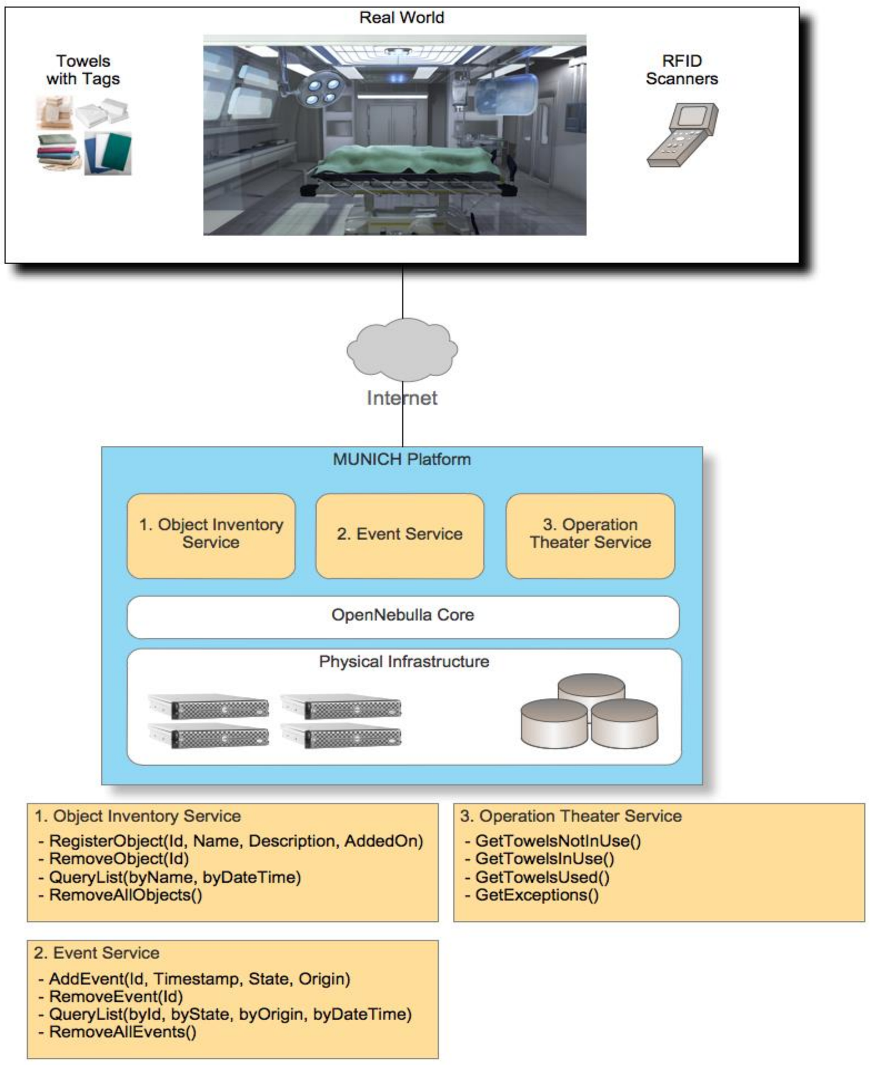


Figure 33: Current Architecture of MUNICH platform, scene 7

The application shown in Figure 34 that monitors the towels being in use and being used 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 to. 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.

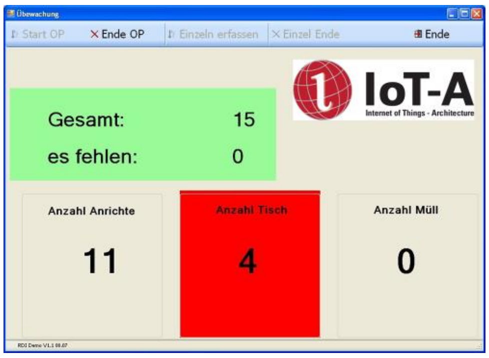


Figure 34: Demonstrator of scene 7 shown at IoT week 2013, Helsinki

3.2.8 Scene #8: Environment and Patient Remote Monitoring (CSD/CATTID)

The implementation of this scene is based on the “Medication Control” health use case scene 8 depicted in D7.2.

3.2.8.1 Step 1: Application Description

Robert is hosted in a twin room with another patient. The room is equipped with sensors to measure the humidity and temperature of the room. When Robert is under care, his body is equipped with body sensors to monitor his body temperature. These sensors enable a continuous monitoring of the patient, which complete the periodic controls done by the nurse. The nurse defines the routine temperature monitoring parameters of Robert like the time, the frequency, temperature limits and she confirms the data through the E-Health HIS.

As defined by the nurse, in the morning, before the nurse starts her visit to the patients, the body temperature measurement is done by the sensors. Unfortunately, the measurement value is a little bit higher than the defined upper threshold. This indication is recognised by the eHealth HIS, which triggers automatic control checks for possible causes. The control of the room temperature shows that the room temperature is lower than the defined value. The responsible hospital staff is informed by HIS about the possible failure in the heating system. The responsible person detects the failure and repairs it. After the repairs have been acknowledged, the eHealth HIS adjusts the room temperature to the defined degree. The nurse is notified that Robert’s body temperature was low during the morning routine and it was due to the failure in the heating system.

3.2.8.2 Step 2: Domain Model Representation of the Demo

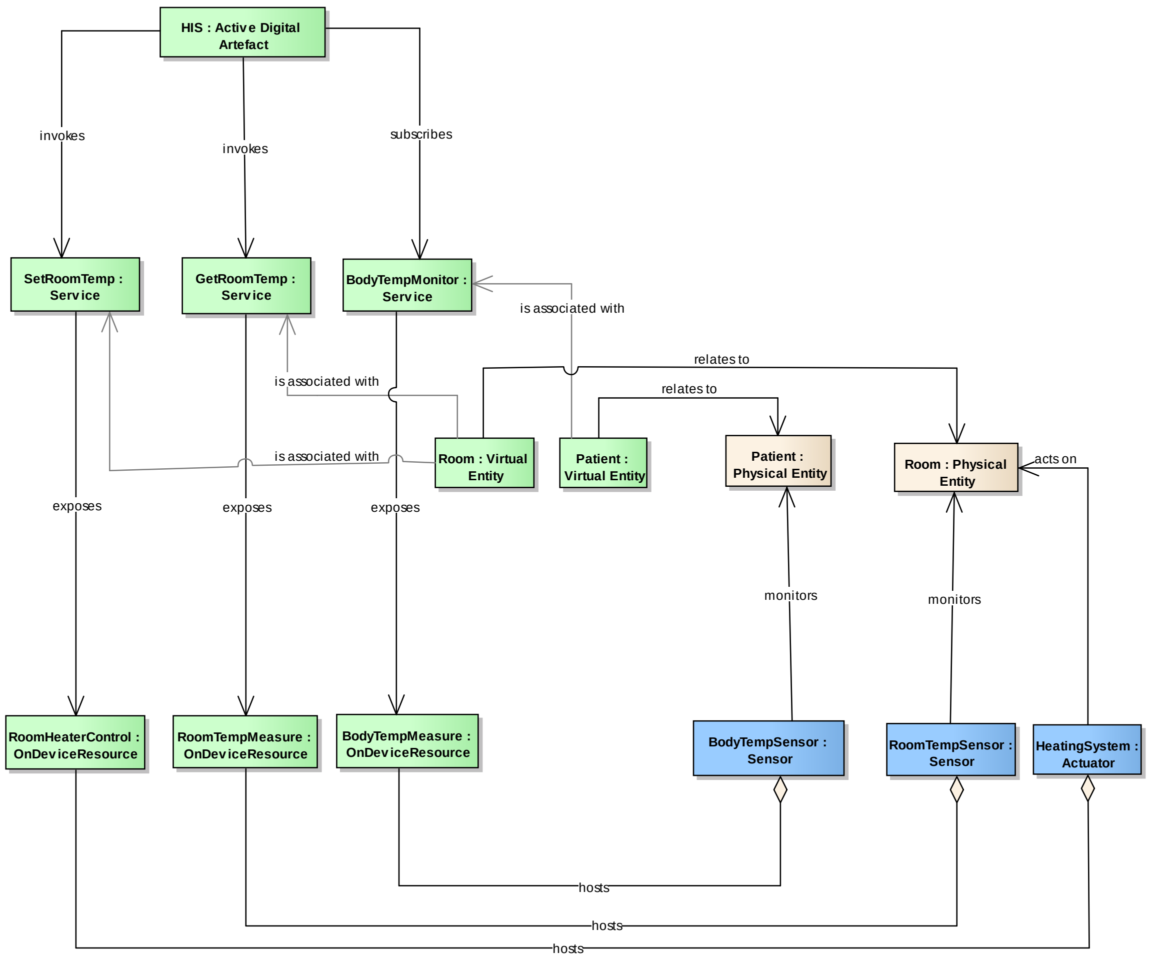


Figure 35: Domain Model of scene 8

The domain model for this scene is represented in Figure 35. The patient and the room he's residing in are modelled as Physical Entities, and the respective and Virtual Entities are related with them. The resources (the sensors) are modelled as OnDeviceResources and they are exposed by the respective services. The HIS, on the other side, is modelled as DigitalArtefact. More in details, the services expose the following resources:

The BodyTempMonitor Service exposes the BodyTemp OnDeviceResource to do the reading of the body temperature sensor value. The HIS subscribes to this service so to be notified every time there is a change of value.

The GetRoomTemp service exposes RoomTemp OnDeviceResource to read the values sensed by the room sensors. This service invoked by HIS whenever there is the need to examine the environmental conditions.

The SetRoomTemp service exposes RoomHeater OnDeviceResource which hosts the actuator HeatingSystem. The actuator sets the room temperature to the default values whenever triggered. The SetRoomTemp service is invoked by the E-Health HIS.

3.2.8.3 Step 3: Information Model Representation of the Demo

Figure 36 provides a graphical representation of the information model specialised to scene 8. In the upper part of the figure there are the attributes of the Virtual Entity Room: temperature and light. These attributes contain single value. The bottom part of the figure, instead, illustrates the services of the Virtual Entity as follows:

- TemperatureMonitoringService: this Service retrieves the value of the temperature in the hospital room using the Resource Temperature through the temperature reader device;
- TemperatureRegulationService: this Service change the value of the Resource temperature in the room through an actuator device;
- LightMonitoringService: this Service retrieves the value of the light in the hospital room using the Resource light through the light reader device;
- LightRegulationService: this Service change the value of the Resource Light in the room through an Actuator device;

The two parts of the Information Model are connected through Associations between the Services and the Virtual Entity Room. These Associations are mapped through the Attributes and the Virtual Entity Room.

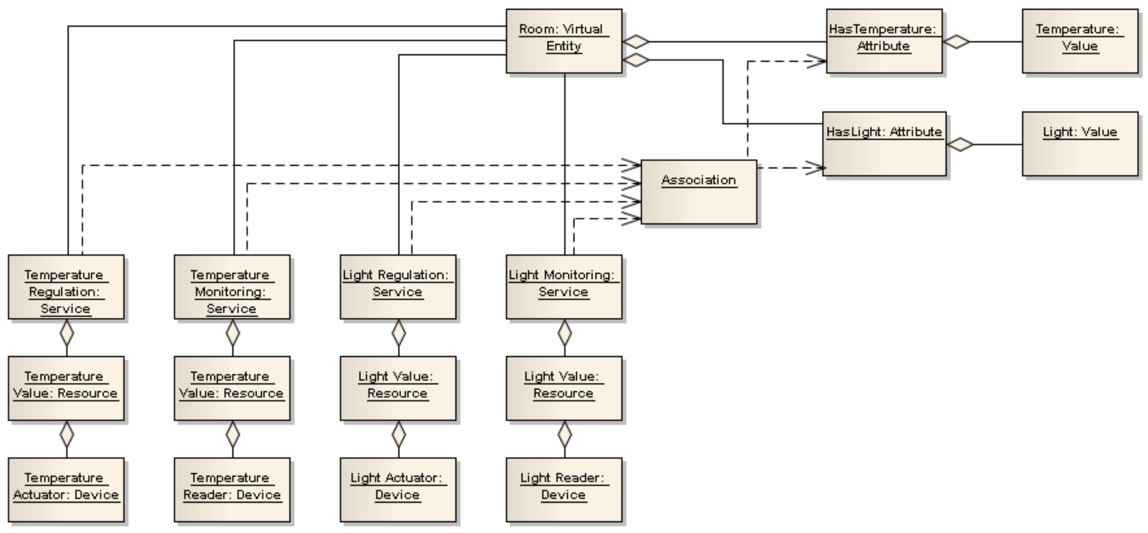

Room: Virtual Entity
HasTemperature: Attribute
Temperature: Value
HasLight: Attribute
Light: Value
Association
Temperature Regulation: Service
Temperature Monitoring: Service
Light Regulation: Service
Light Monitoring: Service
Temperature Value: Resource
Temperature Value: Resource
Light Value: Resource
Light Value: Resource
Temperature Actuator: Device
Temperature Reader: Device
Light Actuator: Device
Light Reader: Device

Figure 36: Information Model of scene 8

3.2.8.4 Step 4: Relevant Design Choices for Implementation

For the IoT system described in this scene, there were different design choices possible depending on the view or perspective taken in the architecture. Since the core idea of this demo scene was on the concept of multiple identifiers, rigorous security was not the main focus; we describe the design choices made specifically for the demo, and where appropriate, we note when a different design might be considered for a real world hospital implementation. For ease of visualisation, the overall design choices are presented in Table 12.

Topic	Design Choice	Impact on			
		Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
Communication Confidentiality	DC9.1 No encryption	▼	▲	●	▲
	DC9.2 End-to-end Encryption	▲	▼	●	▼
	DC9.3 Hop-to-hop Encryption	▲	▼	●	●
	DC9.4 Onion routing-like encryption	▲	▼	●	▼
	DC9.5 Tunnelling	▲	▼	▲▼	▲▼
Bootstrapping	DC12.1 Static key pair	▼	▲	▲	▼
	DC12.2 Updateable key pair	▲	▼	▲	●
	DC 12.3 Static shared secret	●	▲▼	●	▼
	DC 12.4 Updateable shared secret	▲	▲▼	●	●
	DC 12.5 Neighbour keys	●	●	▲	▼
	DC 12.6 Group Key	▼	▲	▲	▲
	DC 12.7 Transitive imprinting	●	▼	●	●
Smart object connectivity	DC16.1 Sensor and actuator netw orks	▼	▲▼	▲▼	▲
	DC16.2 RFID and smart tags	●	●	●	●
	DC16.3 WiFi connectivity	▲	▲	▲	▲
	DC16.4 Cellular netw ork connectivity	●	●	●	●
"Last mile" communication protocols	DC17.1 IoT-A protocol suite	▲	▲	●	▲▼
	DC17.2 Ad hoc proprietary stack	▲	▲	●	▼
	DC17.3 Other standards not in the IoT-A protocol suite	▲▼	▲▼	●	▼

Table 12: Design choices relevant to scene 8. The highlighted choices are those picked for the demonstrator.

3.2.8.5 Step 5: Technical realisation of the Demo

The scene showed how IoT systems can help monitor environmental conditions in a patient's room, along with the medical conditions of the patient. More in details, the hospital is equipped with an eHealth system called Hospital Information System (HIS) which exploits sensors to monitor the aforementioned. The HIS includes also a component running on the hospital personnel's tablets. Whenever the HIS detects an issue, the system automatically performs the pre-defined controls, and, if necessary, the referring personnel member receives a notification on his tablet. The role of the IoT-A in this scene is prominent: It enables the remote monitoring through the sensors (subjects), and the interoperability between them and the IoT-A services which actually detect the anomaly (e.g. high patient's body temperature), fetch the sensing's of the sensors in the room to check if the high temperature of the patient's body is due to environmental conditions. Finally, the wireless communication primitives within IoT-A make it possible for the services to wirelessly communicate to the personnel's tablets a possible alarm.

The IoT-A concepts shown in the demonstrator were WP3 connectivity features and WP4 resolution mechanisms.

The demo setup consisted in a minipc acting as application and database server and as a connectivity gateway between the user IoT phone and the measuring instrument, a sensor node mimicking the behaviour of the measuring instrument and a mobile phone running the health care application through which the user could update her record.

Communications were configured according to WP3 prescription, namely 6LoWPAN and IPv6 were used for the network layer, CoAP and HTTP for the application in the constrained and the unconstrained networks, respectively.

Description	Communication
Blood glucose level reader	802.15.4
IoT phone	WiFi
Gateway (minipc)	WiFi + 802.15.4 + Ethernet

Table 13: Used Devices in implementation for scene 8

The telosb sensor nodes mimicking the blood glucose level reader were connected to the gateway via a 802.15.4 connection. While the IoT phone used a WiFi connection to interact with the gateway.

The personal health care application and the remote server were running on the user mobile phone and the gateway, respectively.

3.2.8.6 Feedback

The demonstrator of this scene has not been shown in Helsinki; comments from similar demonstrator can apply.

In particular, the audience often asked for services capable of instantiating a direct communication between the Nurse and the HIS to allow for the former to trigger changes in the room temperature according to the patient’s needs.

3.2.9 Scene #9: Medication Control (CSD/CATTID)

The implementation of this scene is based on the “Medication Control” health use case scene 9 depicted in D7.2.

3.2.9.1 Step 1: Application Description

The nurse enters the patients room with his/her TazPad device. The TazPad device authenticates wirelessly with the Resolution Infrastructure of the IoT system so to assure that the particular nurse does have the right credentials to further proceed with the medication. After the successful authentication, the nurse reads the patients unique ID by positioning the NFC reader of the TazPad device close to the patient's wristband. Then, the TazPad device pulls from the IoT system (thanks to the Resolution Infrastructure) the patient's EHR. After doing so, the nurse positions the NFC reader of the TazPad device close to the medicine's RFID label. Again, the TazPad device pulls the medicine information from the IoT system. Automatic data analysis takes place to decide whether the medicine is applicable to the user or not, and in case, an alarm is raised.

3.2.9.2 Step 2: Domain Model Representation of the Demo

The domain model fort the demo is depicted in Figure 37.

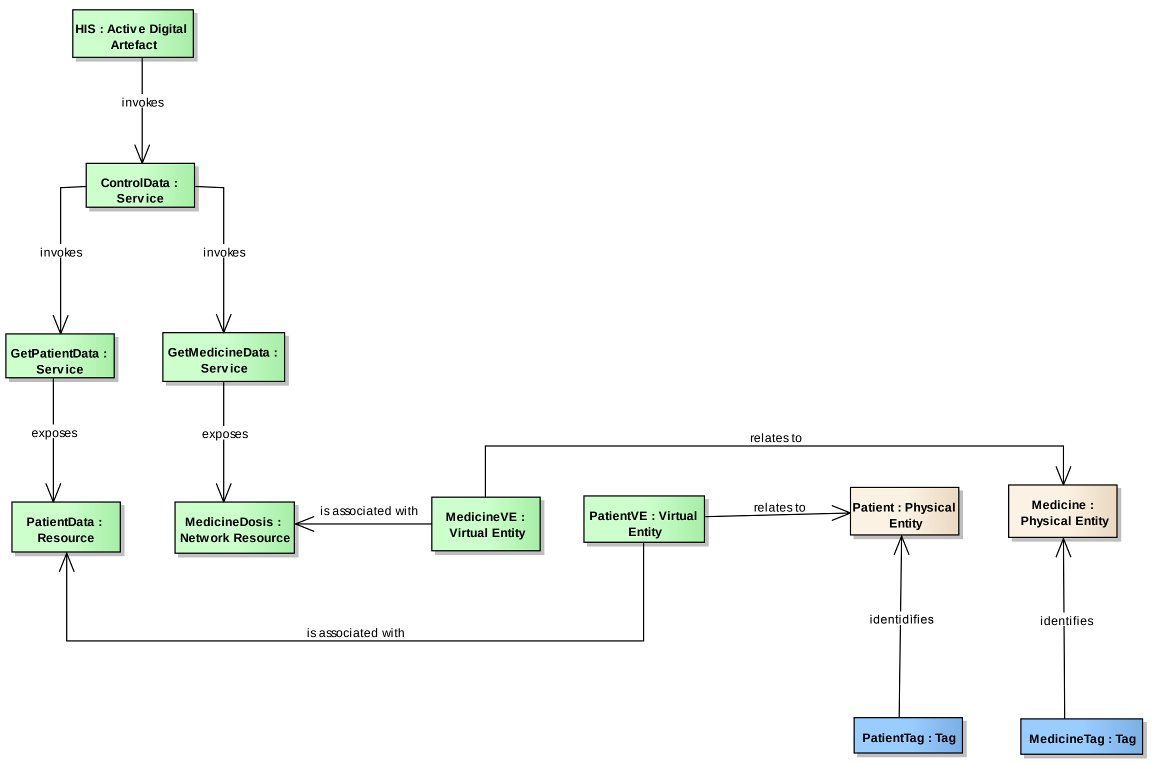


Figure 37: Domain Model of scene 9

The patient and the medicines are modelled as Physical Entities and the respective Virtual Entities are related with them. The RFID wristband and the RFID labels of the medicines provide the unique tags that univocally identify both the patient and the medicines within the HIS system. The services that involved in the domain model are detailed below:

The GetMedicineData service exposes the NetworkResource modelled MedicineDosis which is associated with the Virtual Entity MedicineVE.

The GetPatientData service exposes the PatientData Resource, associated with the Virtual Entity PatientVE.

The ControlData service is invoked by the DigitalArtefact (the HIS). This service invokes, in turn, two other services: GetPatientData and GetMedicineData.

3.2.9.3 Step 3: Information Model Representation of the Demo

Figure 38 provides a graphical representation of the information model specialised to scene 9. In the upper part of the figure there are the attributes of the Virtual Entity E-Health: medicine, patient and allergy.

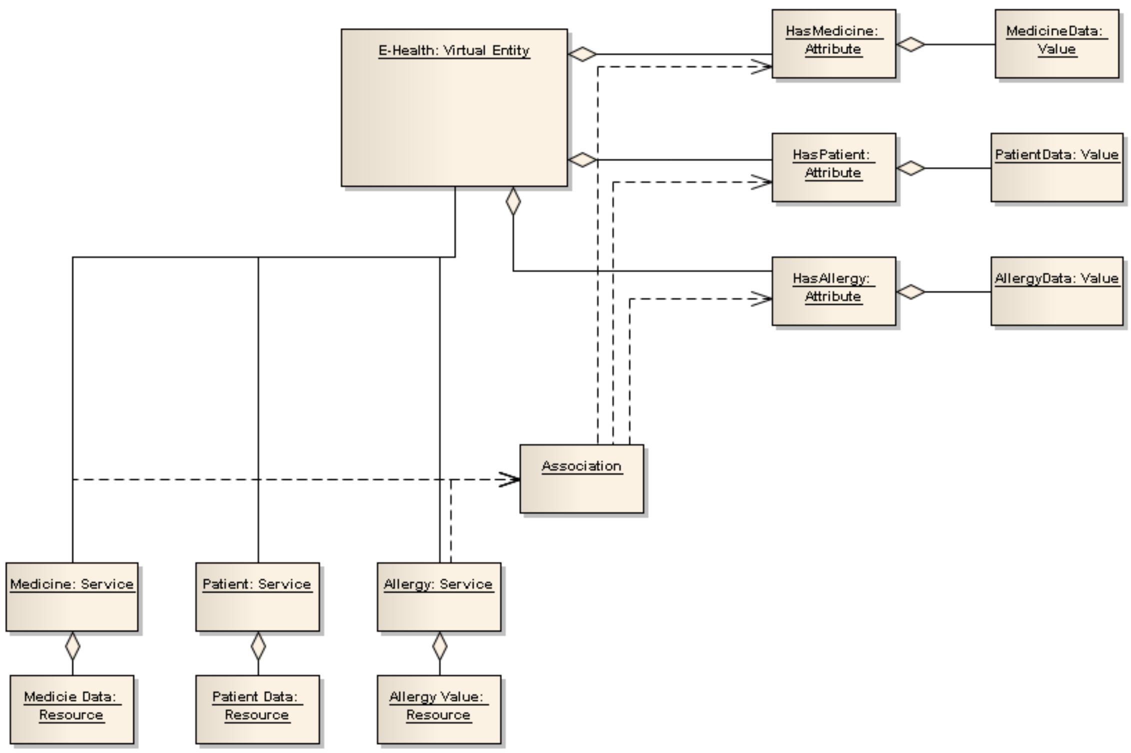


Figure 38: Information Model of scene 9

The Attributes medicine and patient contains multiple values, instead the Attribute allergy contains single value. In the bottom part of the figure are illustrated the following Services of the Virtual Entity:

- Medicine: this Service retrieves the data of a particular medicine through the Resource medicine data;
- Patient: this Service retrieves the data of a particular patient through the Resource patient data;
- Allergy: this Service checks if a given patient is allergic to a particular medicine through the Resource allergy value.

The two parts of the Information Model are connected through Associations between Service and the Virtual Entity eHealth. These Associations are mapped through the Attributes and the Virtual Entity eHealth.

3.2.9.4 Step 4: Relevant Design Choices for Implementation

For the IoT system described in this scene, there were different design choices possible depending on the view or perspective taken in the architecture. Since the core idea of this demo scene was on the concept of multiple identifiers, rigorous security was not the main focus; we describe the design choices made specifically for the demo, and where appropriate, we note when a different design might be considered for a real world hospital implementation. For ease of visualisation, the overall design choices are presented in Table 14.

Topic	Design Choice	Impact on			
		Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
Communication Confidentiality	DC9.1 No encryption	▼	▲	●	▲
	DC9.2 End-to-end Encryption	▲	▼	●	▼
	DC9.3 Hop-to-hop Encryption	▲	▼	●	●
	DC9.4 Onion routing-like encryption	▲	▼	●	▼
	DC9.5 Tunnelling	▲	▼	●	▲▼
Bootstrapping	DC12.1 Static key pair	●	▲	▲	▼
	DC12.2 Updateable key pair	▲	▲	▲	●
	DC 12.3 Static shared secret	▼	▲▼	●	▼
	DC 12.4 Updateable shared secret	▲	▲▼	●	●
	DC 12.5 Neighbour keys	●	●	●	●
	DC 12.6 Group Key	▼	▲	▲	▲
	DC 12.7 Transitive imprinting	●	▼	●	●
Smart object connectivity	DC16.1 Sensor and actuator networks	▼	▲▼	▲▼	▲
	DC16.2 RFID and smart tags	▲	▼	▼	▲
	DC16.3 WiFi connectivity	▲	▲	▲	▲
	DC16.4 Cellular network connectivity	▲	▲▼	▲	▲
"Last mile" communication protocols	DC17.1 IoT-A protocol suite	▲	▲	●	▲▼
	DC17.2 Ad hoc proprietary stack	▲▼	▲	●	▼
	DC17.3 Other standards not in the IoT-A protocol suite	▲▼	▲	●	▼

Table 14: Design choices relevant to scene 9. The highlighted choices are those picked for the demonstrator.

3.2.9.5 Step 5: Technical realisation of the Demo

This demonstrator illustrated how automatic control operations and alarms can be realised through IoT communication and services so to prevent nurses from applying the wrong medicine to patients. During their stay in the hospital recovered patients are given medicines by the nurses. The medicine type and dose is decided by the doctors. The medication type and dose is registered within the Electronic Health Record (EHR) of the patient. When the patient checks in, he is given a wristband equipped with an RFID tag containing the unique ID of the patient's EHR. Similarly, every medicine in the hospital is labelled with an RFID tag which points to the electronic record containing the description of that medicine (Type, amount, expiring date, etc.). Nurses are equipped with TazPad devices, which have a built-in NFC reader through which they read the RFID tags of both patients and medicines.

The demo setup consisted in a mini-pc acting as a connectivity gateway between the nurse's TazPad and the IoT Resolution Infrastructure, the patient's wristband equipped with the RFID tag, and the medicines labelled with RFID tags.

Communications were configured according to WP3 prescription: CoAP and HTTP for the application in the constrained and the unconstrained networks, respectively.

The applications, devices, and functional components are detailed in Table 15.

Applications	Devices	Functional Components
SW.CATTID.4 (HIS Tablet Application)	HW.CATTID.4 (Tablet)	C.CATTID.2 (Application)
	HW.CATTID.5 (RFID Tag)	C.CATTID.3 (Device connectivity and communication)
	HW.CATTID.6 (Gateway)	

Table 15: Details on the applications, devices, and functional components involved in the demo

Figure 39: Demonstration at IoT week 2013, Helsinki

3.2.9.6 Feedback

During the IoT week this scene received many comments related to the role of the patient's personnel (the nurse in this case) within the system. In particular, quite a few people were asking whether it might be a better idea to give more "power" to the Nurse, so that she was able to insert new data in the system about the patient, update the patient's record after the application of the medicine and so on.

3.3 Retail Use Case

3.3.1 Scene #1: NFC-supported check-in and assisted loading (FHG IML)

The implementation of this scene is based on the “NFC-supported check-in and assisted loading” use case scene depicted in D7.2.

3.3.1.1 Step 1: Application Description

The scene shows how NFC devices can be used to allocate new transport orders to truck drivers and furthermore how each driver can be supplied with additional information which is needed to perform his task. The transport orders are allocated using the manufacturer’s ERP system.

Ted, the truck driver, arrives at the gardener’s production site to pick up some goods he has to transport. Up until now he does not know anything about his task for this day. To get more information he checks in at the manufacturer’s registration office by holding his IoT-Phone above the check in terminal which is located in front of the entrance barrier. The NFC reader inside the terminal receives Ted's identifier and language information located in his IoT-Phone and sends a notification to the manufacturer’s ERP system. This retrieves the transport order which has been assigned to Ted from the transport order database and sends all information including the gate number to Ted using the notification service running on his IoT-Phone; Ted will be delivering orchids today.

After he gets the information, Ted drives to the appropriate gate and starts loading the intelligent load carriers containing the orchids into his truck. All load carriers are equipped with sensor nodes which measure temperature, humidity and acceleration. Every time Ted puts a carrier into the truck, he uses his IoT-Phone to scan the load carrier's barcode to mark it as loaded and signs up for sensor events of the attached sensor node. After he finished loading, Ted confirms it to the manufacturer, receives his shipping order, and starts driving away.

3.3.1.2 Step 2: Domain Model Representation of the Demo

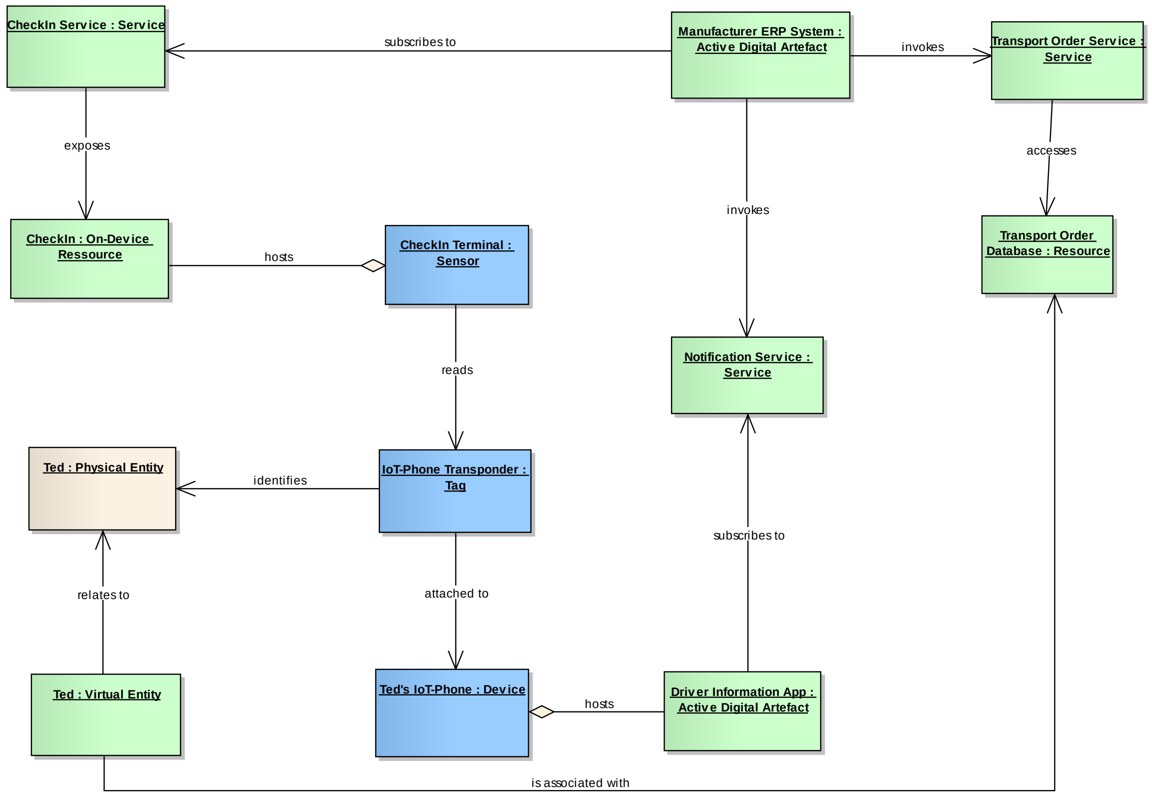

CheckIn Service : Service
subscribes to
Manufacturer ERP System : Active Digital Artefact
invokes
Transport Order Service : Service
exposes
accesses
invokes
CheckIn : On-Device Ressource
hosts
CheckIn Terminal : Sensor
Transport Order Database : Resource
Notification Service : Service
reads
Ted : Physical Entity
identifies
IoT-Phone Transponder : Tag
subscribes to
relates to
attached to
Ted : Virtual Entity
Ted's IoT-Phone : Device
hosts
Driver Information App : Active Digital Artefact
is associated with

Figure 40: Domain Model of scene 1 – Arrival at the production site

Figure 40 shows the modeling of the different components used in arrival part of scene 2. The CheckIn Terminal *Sensor* reads the virtual transponder *Tag* which is attached to Ted's IoT-Phone *Device*. Over the embedded CheckIn *Service* the notification data is sent to the Manufacturer ERP System which is defined as a *Digital Artefact*. This invokes the Transport Order *Service* for searching the transport orders which are associated with Ted's *Virtual Entity*. In the next step the found transport order information is sent to the Notification *Service* which has been subscribed by the Driver Information App an *Active Digital Artefact* running on Ted's IoT-Phone.

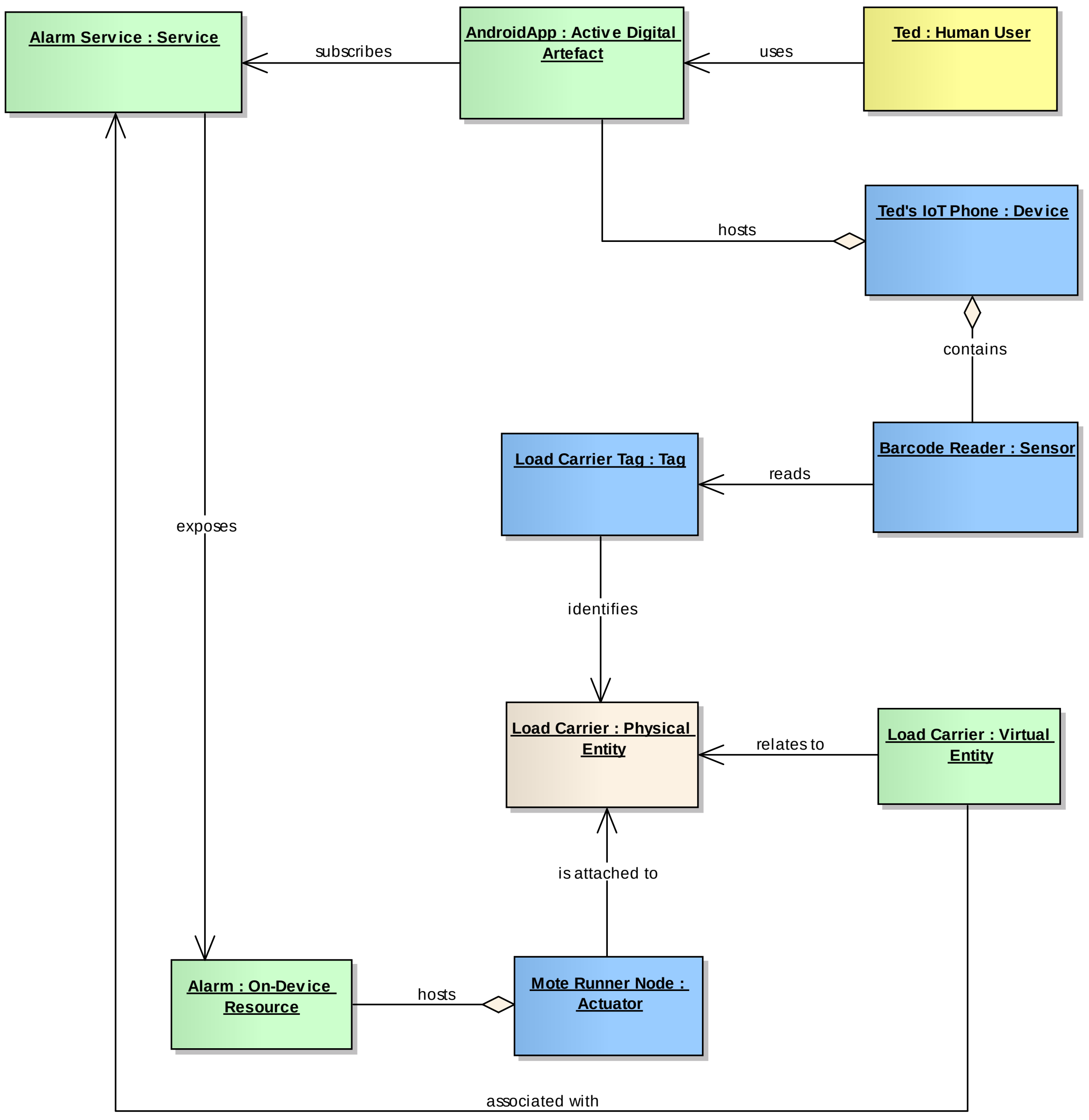


Figure 41: Domain Model of scene 1 – Loading

In Figure 41 the class instance diagram of the loading process is shown. The sensor node *Device* which is attached to a load carrier *Physical Entity* hosts a measurement *Service* as an *On-Device Resource* which will be subscribed by the AndroidApp (*Active Digital Artefact*) for getting live measurement data. Scanning the Load Carrier *Tag* with the embedded RFID-Reader *Sensor* of Ted's IoT-Phone *Device* identifies the *Physical Entity* of the Load Carrier, thus allowing the AndroidApp to get access to the Alarm *Service* (which corresponds to the *Virtual Entity* of the Load Carrier).

3.3.1.3 Step 3: Information Model Representation of the Demo

The information model representation of scene 1 is divided into two parts. The first part affects the arrival process at the check-in terminal. The second part concerned with the loading process of the smart load carriers.

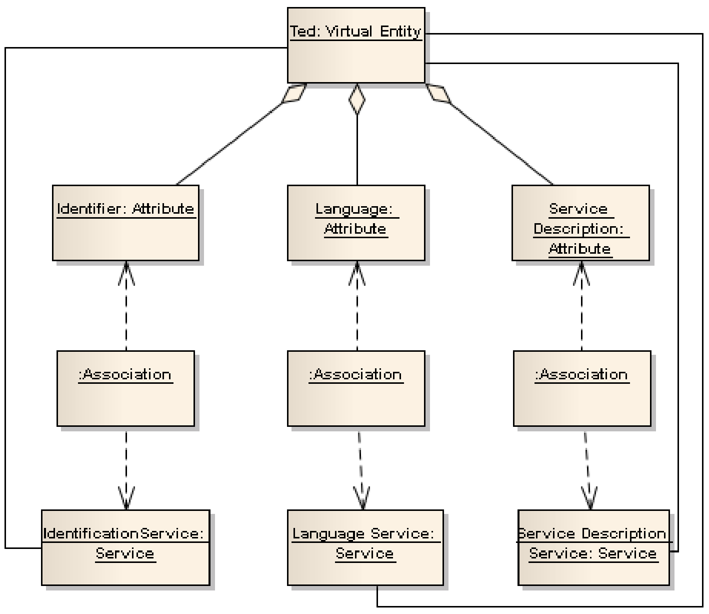

Ted: Virtual Entity
Identifier: Attribute
Language: Attribute
Service Description: Attribute
:Association
:Association
:Association
IdentificationService: Service
Language Service: Service
Service Description Service: Service

Figure 42: Information Model of scene 1 – Terminal

There is one Virtual Entity pictured in Figure 42. Ted the truck driver has three attributes used for the terminal process. The Identifier consists of a number which is required for the service lookup. The Language helps setting the language of the user interface Ted is using. For adding the services, Ted offers to the production site the third attribute, which is the service description.

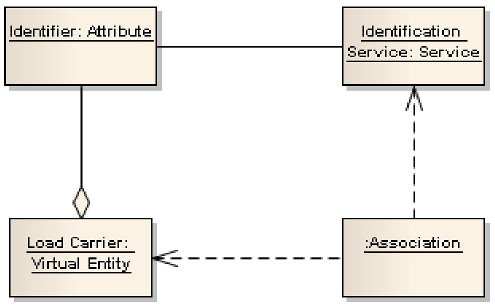


Figure 43: Information Model of scene 1 – Loading

In Figure 43 the loading process consists of one virtual entity, namely the load carrier which should be loaded. In this scene, the load carrier has an attribute for the identification which is used by the truck driver for doing the signup for sensor alarms.

3.3.1.4 Step 4: Relevant Design Choices for Implementation

In order to build a demonstrator for the scene we took into consideration many aspects that do not have straightforward solutions, in fact under many implementation fields it is possible to find issues that can be solved in different ways and, while a given solution may result to be the best under a particular view, it is also possible that other solutions outperform the former adopting different views.

All that said, we thought that in this particular scene the design choices that impacted the most on the final result are the following (see also Table 16):

- VE Resolution with mandatory security – handles functions needed for handling with resolution, monitoring, and storage of history of the virtual entity. In our case, the truck driver scans the load carriers upon loading, where the NFC tags attached to each load carrier contain the VE IDs. The VE Resolution shall only be allowed for registered drivers.
- IoT Service Resolution with mandatory security – Upon loading the IoT Service Resolution shall only be used by registered drivers. A driver registers for the AlarmService offered by the load carrier.
- IoT Service with mandatory security – The driver's NotificationService shall only be used by the authorized backend system. The driver will receive the loading list and transport destination over this service.
- Crypto-based authentication over open channel – The NFC-based terminal authenticates a driver over an open channel to initiate a session key.
- Cellular network connectivity – this design choice is due to the connectivity problem of the truck driver, who, arriving at a location needs to get into contact with the backend systems. Upon check-in all further communication with the driver is done over cellular networks, as a local and publically available installation of WLAN is not feasible.

		Impact on			
Topic	Design Choice	Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
VE Resolution	DC3.1 VE Resolution with mandatory security	▲▼	●	▲	●
	DC3.2 VE Resolution with optional security	▼	●	▼	●
	DC3.3 VE Resolution with QoS	●	●	▲	●
	DC3.4 VE Resolution domain-oriented	▲	▲	▲	▲
	DC3.5 VE Resolution location-oriented	▼	▲	▲▼	▲▼
	DC3.6 Resolution semantic Web-oriented	●	●	▲	▲▼
	DC3.7 Resolution Peer-to-Peer-oriented	●	▲	▲	●
	DC3.8 Resolution Federation-based	●	▲	▲	●
IoT Service Resolution	DC4.1 IoT Service Resolution with mandatory security	▲▼	●	▲	●
	DC4.2 IoT Service Resolution with optional security	▼	●	▼	●
IoT Service	DC4.3 IoT Service with mandatory security	▲▼	●	▲	●
	DC4.4 IoT Service with optional security	▼	●	▼	●
Identification and Authentication	DC5.1 Identifier based identification	▼	▲	▲▼	●
	DC5.2 Crypto-based authentication over open channel	▲▼	▼	▲▼	▼
	DC5.3 Authentication over encrypted channel	▲	▼	▲	▼
Smart object connectivity	DC16.1 Sensor and actuator networks	▼	▲▼	▲▼	▲
	DC16.2 RFID and smart tags	▲	▼	▼	▲
	DC16.3 WiFi connectivity	▲	▲	▲	▲
	DC16.4 Cellular network connectivity	▲	▲▼	▲	▲

Table 16: Design choices relevant to scene 1. The highlighted choices are those picked for the demonstrator.

3.3.1.5 Step 5: Technical realisation of the Demo

Figure 44 shows a structured view of the technical realisation of the NFC-supported check-in use case in scene 1. In the authentication process the truck driver Ted and the Terminal (1) a secure channel between both is set up.

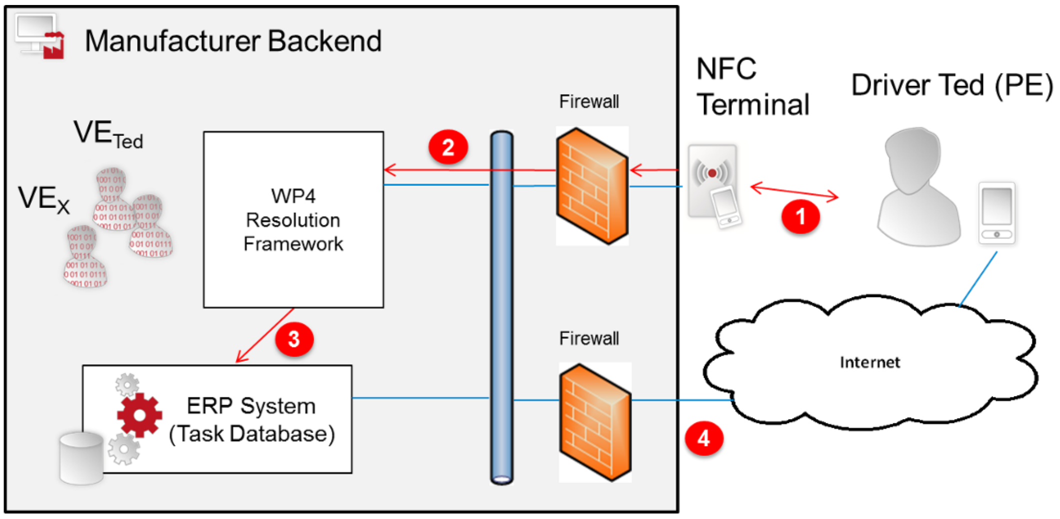


Figure 44: Implementation of scene 1

After this Ted is added with his Virtual Entity ID and the associated services to the WP4 Resolution Framework (2) and the ERP system gets informed about the arrival of a new truck driver (3). This prompts the ERP system to check for tasks or specific transport orders of Ted. If a task was found, then the ERP system does a lookup of Ted's ID to pick the associated Notification-Service for sending the related task information to Ted (4).

The implemented demonstrator is shown in Figure 45.

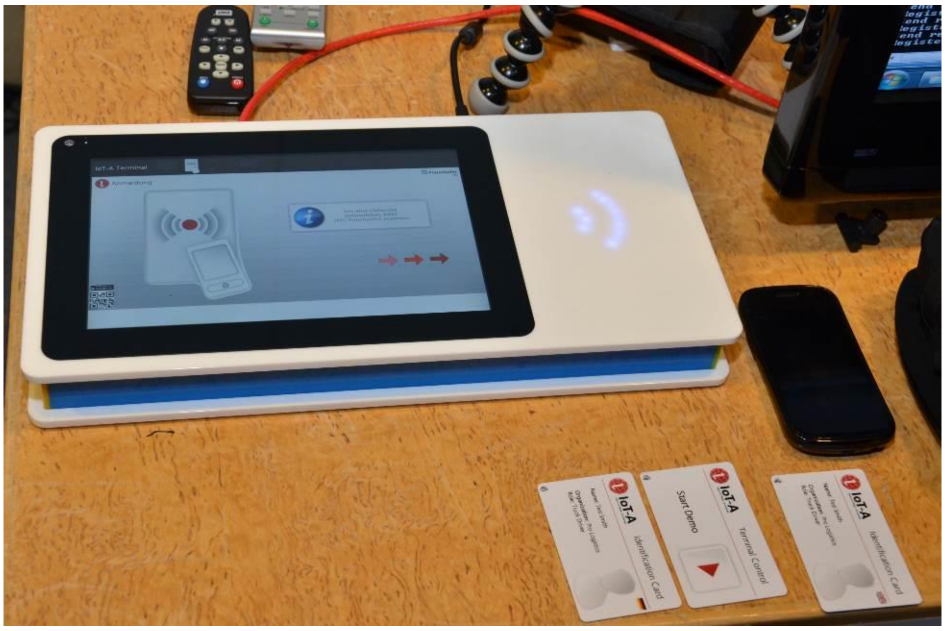


Figure 45: Demonstrator at IoT week 2013, Helsinki

3.3.1.6 Feedback

The feedback for this use case is described in the feedback chapter of scene 3.

3.3.2 Scene #2: Transport Monitoring with Smart Load Carriers (FHG IML)

The implementation of this scene is based on the “Transport monitoring with Smart Load Carriers” use case scene depicted in D7.2.

3.3.2.1 Step 1: Application Description

This scene shows how the Internet-of-Things-Architecture can be used with intelligent load carriers, equipped with humidity, temperature and shock sensors to ensure a better and easily traceable way of monitoring and maintaining product quality during transport. This reduces the probability of quality loss during transportation and the data saved in the sensor history can later be used to reject goods of unsatisfying quality or adjust prices in further retail.

In this scene Ted, a truck driver, transports fruits and vegetables from a gardener to a retail shop. The goods reside within smart load carriers, equipped with sensors which monitor the load carriers’ environment and can communicate with Ted’s IoT-Phone. A user-friendly IoT-Phone Application handles the communication between the sensor nodes and the IoT-Phone.

To illustrate how the IoT-A and the sensors help ascertain product quality, an example of the automatic communication between the sensor node and Ted’s IoT-Phone is given: When Ted stops at a rest area, he forgets to leave the load’s air conditioner turned on and the temperature inside the truck starts to rise. When the sensors measure a temperature that exceeds the goods’ critical limit, the IoT Resolution Service is called by the sensor to get Ted’s virtual entity and the corresponding alarm service. Then, after this alarm service is called by the sensor, Ted gets a temperature alarm on his IoT-Phone. Immediately Ted returns to the truck to turn the air conditioner back on.

3.3.2.2 Step 2: Domain Model Representation of the Demo

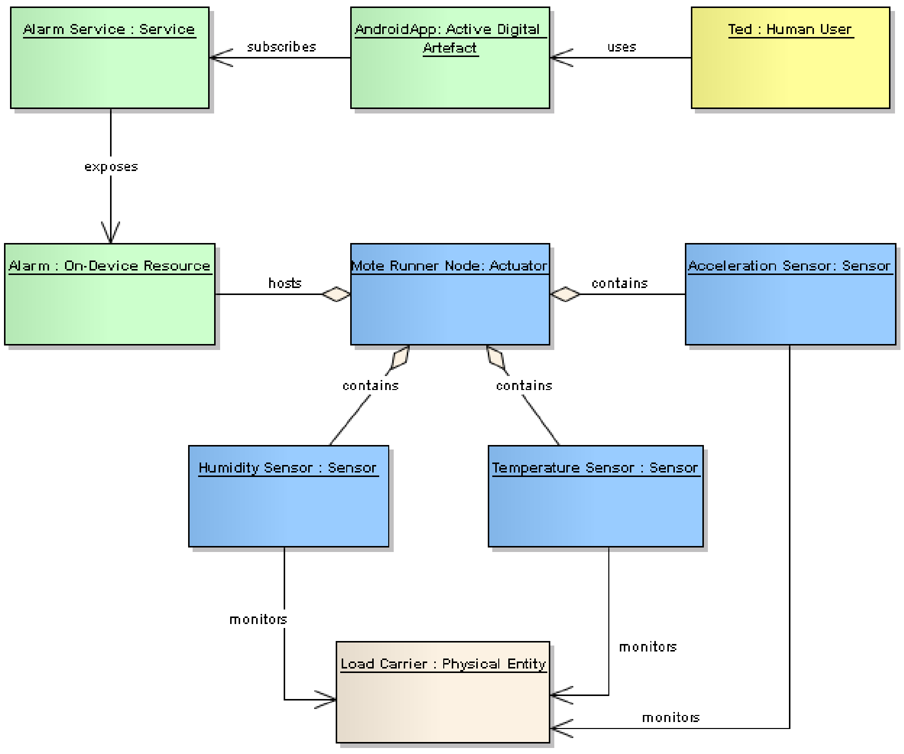

Alarm Service : Service
subscribes
AndroidApp: Active Digital Artefact
uses
Ted : Human User
exposes
Alarm : On-Device Resource
hosts
Mote Runner Node: Actuator
contains
Acceleration Sensor: Sensor
contains
contains
Humidity Sensor : Sensor
Temperature Sensor : Sensor
monitors
monitors
Load Carrier : Physical Entity
monitors

Figure 46: Domain Model of scene 2 – Transport monitoring

The modeling of scene 2 is shown in the above Figure 46. The sensor node *Device*, which is connected to temperature, humidity and acceleration *Sensors*, monitors the corresponding load carrier *Physical Entity*. The measured data is sent to the Android application which is an *Active Digital Artefact* over the integrated measurement *Service* as an *On-Device Resource*.

3.3.2.3 Step 3: Information Model Representation of the Demo

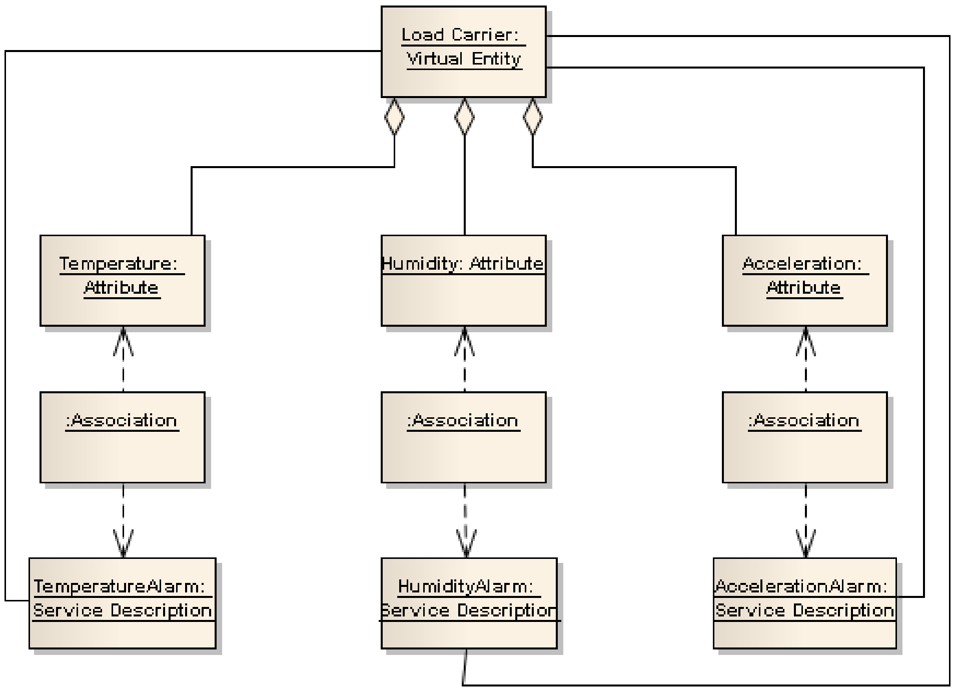


Figure 47: Information Model of scene 2 – Transport monitoring

In Figure 47 the transport monitoring process consists of one virtual entity of the load carrier which has one attribute for every sensor equipped to it. The temperature, humidity and acceleration attributes are associated to related services which are used by the truck driver for getting sensor alarms.

3.3.2.4 Step 4: Relevant Design Choices for Implementation

In order to build a demonstrator for the scene we took into consideration many aspects that do not have straightforward solutions, in fact under many implementation fields it is possible to find issues that can be solved in different ways and, while a given solution may result to be the best under a particular view, it is also possible that other solutions outperform the former adopting different views.

All that said, we thought that in this particular scene the design choices that impacted the most on the final result are the following (see also Table 17):

- VE Resolution with mandatory security – handles functions needed for handling with resolution, monitoring, and storage of history of the virtual entity. In this scene the load carrier knows the VE ID of the driver and is using the VE resolution to find the Notification Service where alarms may be sent to.
- IoT Service Resolution with mandatory security – Upon loading the IoT Service Resolution shall only be used by registered drivers. The load carrier is able to resolve the NotificationService offered by the driver.

- IoT Service with mandatory security – The driver's NotificationService shall only be used by the authorized backend system. The driver will receive the alarms and current sensor values of the load carrier over this service.

- Storage of History Locally and Remotely – Parts of the sensor data is stored locally on the used gateway; other parts are sent to a remote destination, e.g. alarms of critical situations and real-time tracking data.

- Service Hosting on Gateways – An installed gateway hosts the services to get in contact with a load carrier, due to energy and complexity restrictions.

Topic	Design Choice	Impact on			
		Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
VE Resolution	DC3.1 VE Resolution with mandatory security	▲▼	●	▲	●
	DC3.2 VE Resolution with optional security	▼	●	▼	●
	DC3.3 VE Resolution with QoS	●	●	▲	●
	DC3.4 VE Resolution domain-oriented	▲	▲	▲	▲
	DC3.5 VE Resolution location-oriented	▼	▲	▲▼	▲▼
	DC3.6 Resolution semantic Web-oriented	●	●	▲	▲▼
	DC3.7 Resolution Peer-to-Peer-oriented	●	▲	▲	●
	DC3.8 Resolution Federation-based	●	▲	▲	●
IoT Service Resolution	DC4.1 IoT Service Resolution with mandatory security	▲▼	●	▲	●
	DC4.2 IoT Service Resolution with optional security	▼	●	▼	●
IoT Service	DC4.3 IoT Service with mandatory security	▲▼	●	▲	●
	DC4.4 IoT Service with optional security	▼	●	▼	●
Storage of Information History	DC14.1 Storage of History locally	▲	▼	▼	●
	DC14.2 Storage of History remotely	▲▼	▲	▲	●
	DC14.3 Storage of History locally and remotely	▲▼	▲	▲	●
Service hosting	DC18.1 on smart objects	▲▼	▲	▲	●
	DC18.2 on gateways	▲	▼	▲▼	●
	DC18.3 in the cloud	▲	▲▼	▲	●

Table 17: Design choices relevant to scene 2. The highlighted choices are those picked for the demonstrator.

3.3.2.5 Step 5: Technical realisation of the Demo

In Figure 48 the structured view of the technical implementation of scene 2 is shown. The smart pallets owned by the manufacturer measure the ambient conditions like temperature, humidity and the current acceleration.

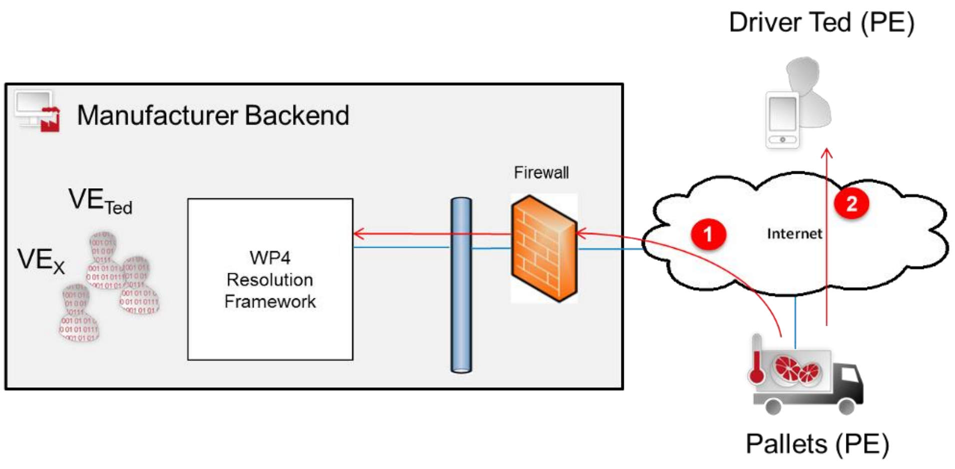
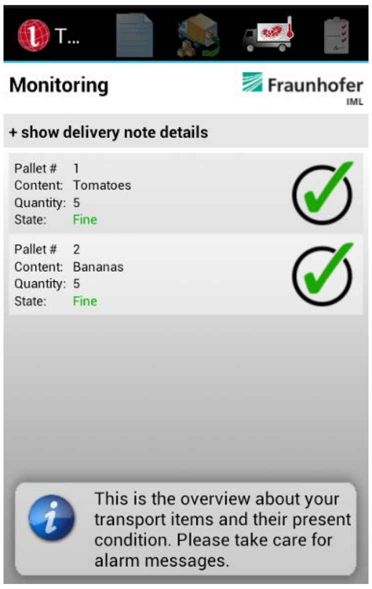


Figure 48: Implementation of scene 2

For sending them to the truck driver, they do a lookup in the resolution framework of the manufacturer's backend system (1). By doing this, the pallets get the URL to the alarm service associated with the current truck driver Ted. After this, they send their measurements to Ted (2).


T...

Monitoring

Fraunhofer IML

+ show delivery note details

Pallet # 1
Content: Tomatoes
Quantity: 5
State: Fine

Pallet # 2
Content: Bananas
Quantity: 5
State: Fine

This is the overview about your transport items and their present condition. Please take care for alarm messages.

Figure 49: Smartphone App „Transport Monitoring“

In Figure 49 the transport monitoring GUI of the truck driver smartphone app is shown. The truck driver Ted gets the measured sensor information on his phone and is informed about sensor alarms.

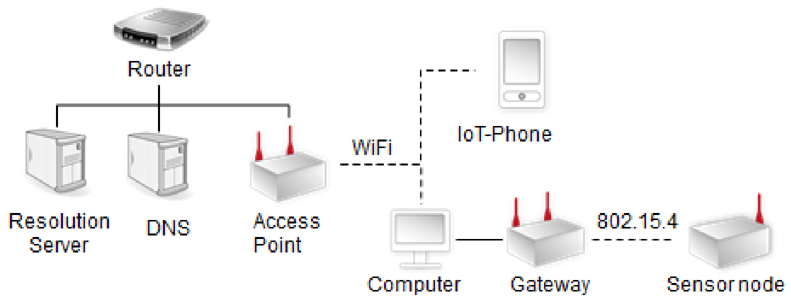


Figure 50: Physical Setup of the Transport Monitoring

Figure 50 shows the physical setup of this demo which consists of sensor nodes communicating with a central gateway which is equipped to a computer to enable the communication with the resolution server and the IoT-Phone of the truck driver Ted.

Figure 51 shows the implemented demonstrator shown at both IoT week 2012 and IoT week 2013. Here the Moterunner sensor node simulator is shown which part of the transport monitoring scenario is.

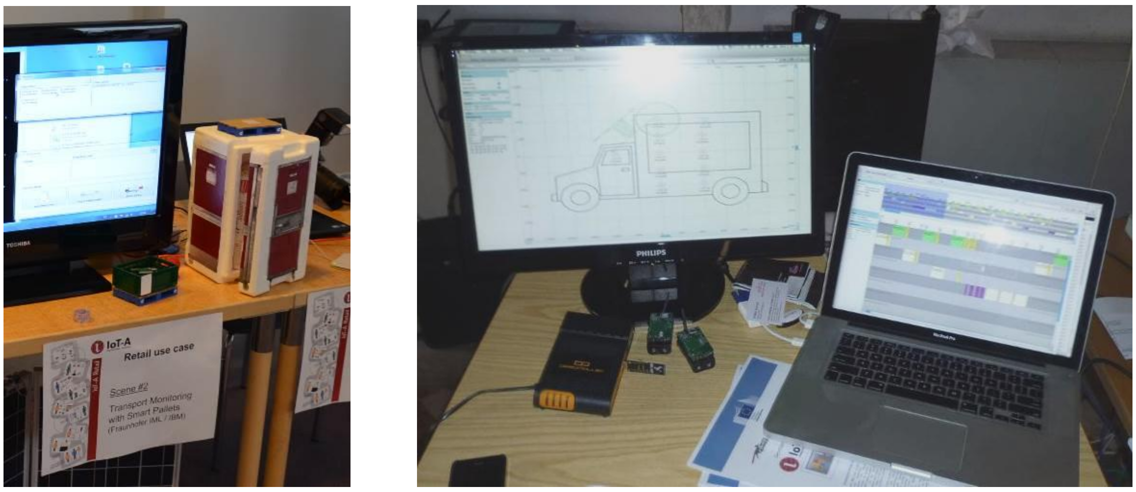


Figure 51: Implementation of demonstrator

3.3.2.6 Feedback

The feedback for this use case is described in the feedback chapter of scene 3.

3.3.3 Scene #3: Handover, assisted quality check and digital signature (FHG IML)

The implementation of this scene is based on the “Assisted quality check and digital signature” use case scene depicted in D7.2.

3.3.3.1 Step 1: Application Description

When Ted arrives at the store, he gets access to the local WLAN. There he does a geo discovery for the nearest responsible shop staff member who is able to do the handover. After that Ted sends the delivery note and the associated sensor history to John the determined staff member. John gets informed about the delivery by his IoT-Phone which shows the delivery note on the display. He now is able to check the sensor history on his IoT-Phone for unacceptable violations of environmental limits critical for the goods’ quality and decide whether he wants to reject parts of the delivery or not. During the quality check, John’s IoT-Phone draws his attention to a recorded rise in temperature during transportation of one load carrier, so John does a visual inspection of it. Once he has assured himself that all the goods are still in good quality, he confirms the delivery and holds up his IoT-Phone to authorize Ted to receive the signed delivery note by using the Peer2Peer functionality of John’s and Ted’s IoT-Phones.

3.3.3.2 Step 2: Domain Model Representation of the Demo

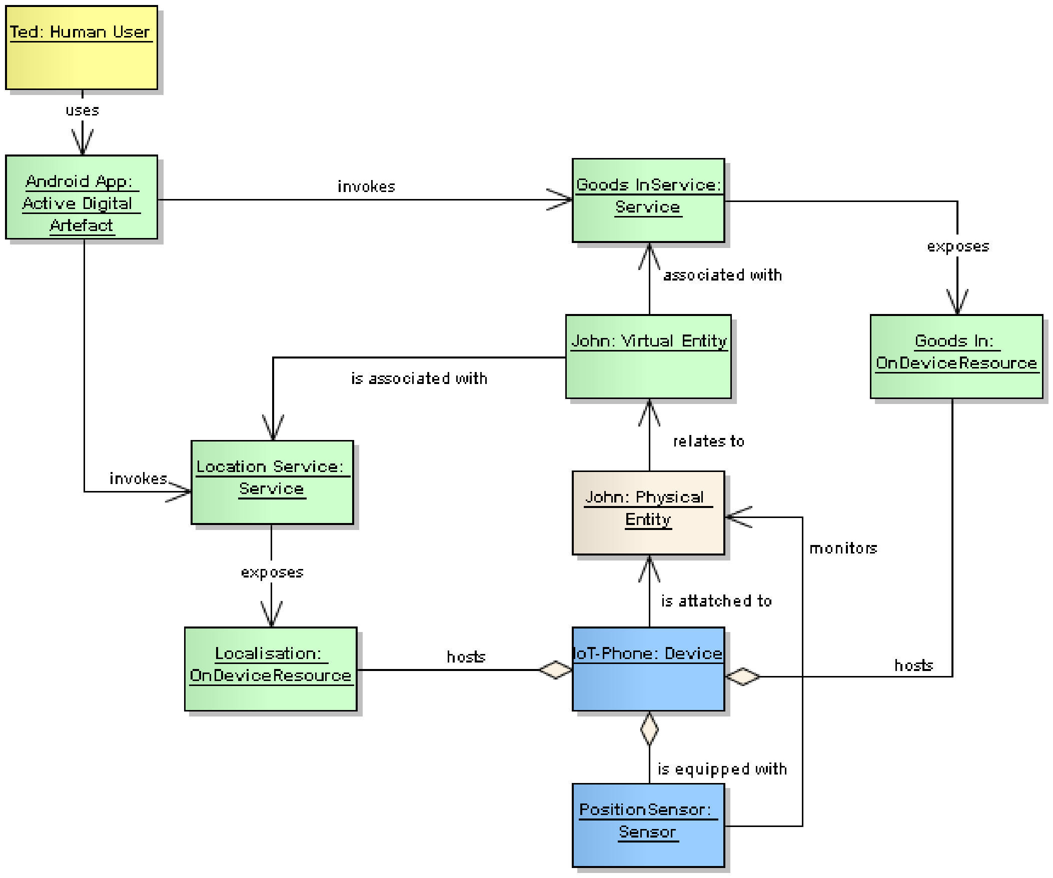


Figure 52: Domain Model of scene 3 – Handover

In Figure 52 the domain model of the handover process is shown. John as a *Physical Entity* is monitored by the Position *Sensor* equipped to his IoT-Phone *Device*. It tracks John's position and offers the corresponding location information over the Location *Service* which is hosted on the IoT-Phone and associated to the *Virtual Entity* of John. A second service which is also hosted on John's IoT-Phone is the Goods-In *Service* which is used to inform John about new deliveries. Ted a *Human User* invokes these services by using his Android App (*Active Digital Artefact*).

3.3.3.3 Step 3: Information Model Representation of the Demo

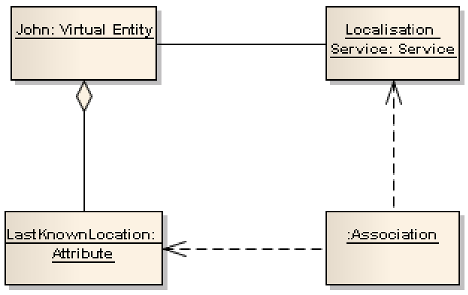


Figure 53: Information Model of scene 3 – Handover

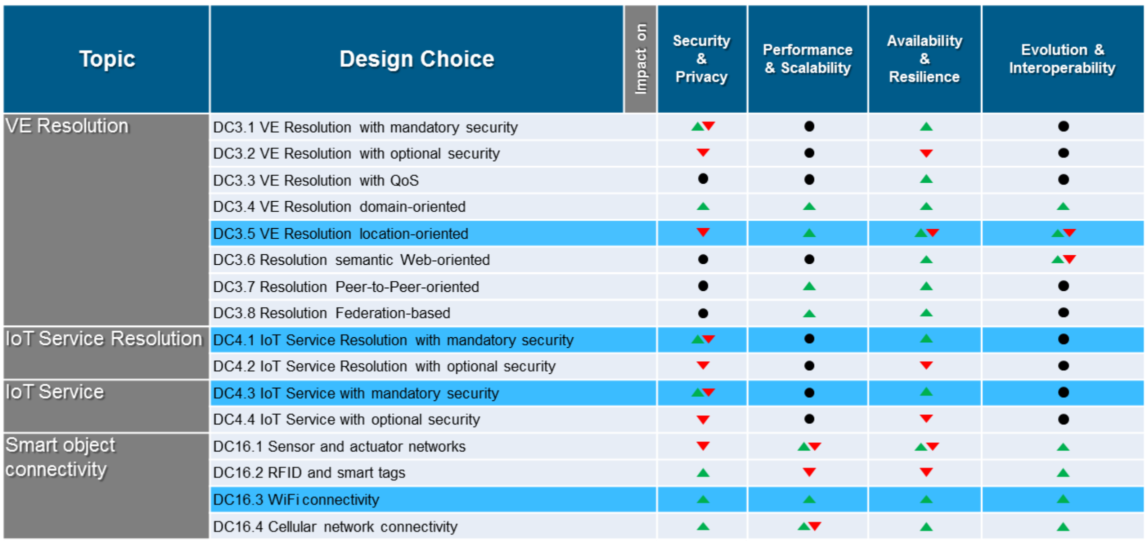
In Figure 53 the handover process consists of one virtual entity of John which has one attribute keeping information about John's last known location. This information is used for the geo discovery functionality in this scene.

3.3.3.4 Step 4: Relevant Design Choices for Implementation

In order to build a demonstrator for the scene we took into consideration many aspects that do not have straightforward solutions, in fact under many implementation fields it is possible to find issues that can be solved in different ways and, while a given solution may result to be the best under a particular view, it is also possible that other solutions outperform the former adopting different views.

All that said, we thought that in this particular scene the design choices that impacted the most on the final result are the following (see also Table 18):

- VE Resolution location-oriented – handles functions needed for handling with resolution, monitoring, and storage of history of the virtual entity. In our case, the truck driver arriving at the store needs to find the responsible partner for a handover. He is using a location-based search to find the store manager.
- IoT Service Resolution with mandatory security – Upon loading the IoT Service Resolution shall only be used by registered drivers. The driver sends his delivery note to the GoodsInService of the responsible person's smartphone.
- IoT Service with mandatory security – The store manager's GoodsInService shall only be used by authorized drivers.
- WiFi connectivity – We chose local WiFi connectivity in the store, as this is usually available. The driver is granted access to the local WiFi by scanning a NFC tag which contains the access code.



Topic	Design Choice	Impact on			
		Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
VE Resolution	DC3.1 VE Resolution with mandatory security	▲▼	●	▲	●
	DC3.2 VE Resolution with optional security	▼	●	▼	●
	DC3.3 VE Resolution with QoS	●	●	▲	●
	DC3.4 VE Resolution domain-oriented	▲	▲	▲	▲
	DC3.5 VE Resolution location-oriented	▼	▲	▲▼	▲▼
	DC3.6 Resolution semantic Web-oriented	●	●	▲	▲▼
	DC3.7 Resolution Peer-to-Peer-oriented	●	▲	▲	●
	DC3.8 Resolution Federation-based	●	▲	▲	●
IoT Service Resolution	DC4.1 IoT Service Resolution with mandatory security	▲▼	●	▲	●
	DC4.2 IoT Service Resolution with optional security	▼	●	▼	●
IoT Service	DC4.3 IoT Service with mandatory security	▲▼	●	▲	●
	DC4.4 IoT Service with optional security	▼	●	▼	●
Smart object connectivity	DC16.1 Sensor and actuator networks	▼	▲▼	▲▼	▲
	DC16.2 RFID and smart tags	▲	▼	▼	▲
	DC16.3 WiFi connectivity	▲	▲	▲	▲
	DC16.4 Cellular network connectivity	▲	▲▼	▲	▲

Table 18: Design choices relevant to scene 3. The highlighted choices are those picked for the demonstrator.

3.3.3.5 Step 5: Technical realisation of the Demo

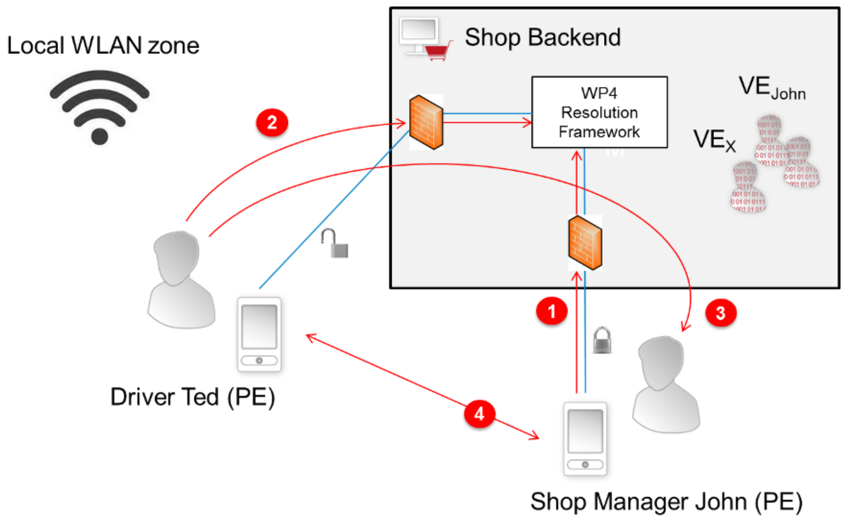


Figure 54: Implementation of scene 3

Figure 54 shows the setup of the demonstrator. John the shop manager creates a secure WLAN connection to the shop for getting access to the shop's backend system (1). Ted the truck driver gets access to the local client WLAN of the shop by scanning the access NFC tag inside the shop. After that he does a Geo-Discovery for a responsible shop member and gets back the Goods-In-Service of the shop manager John (2). Over this service he sends the delivery note and the corresponding sensor history to John who gets the information on his IoT-Phone (3). After the quality check is done by John he sends the signed delivery note back to Ted over the Peer2Peer functionality of Ted's and John's IoT-Phone (4).

The implemented Smartphone application is shown in Figure 55.

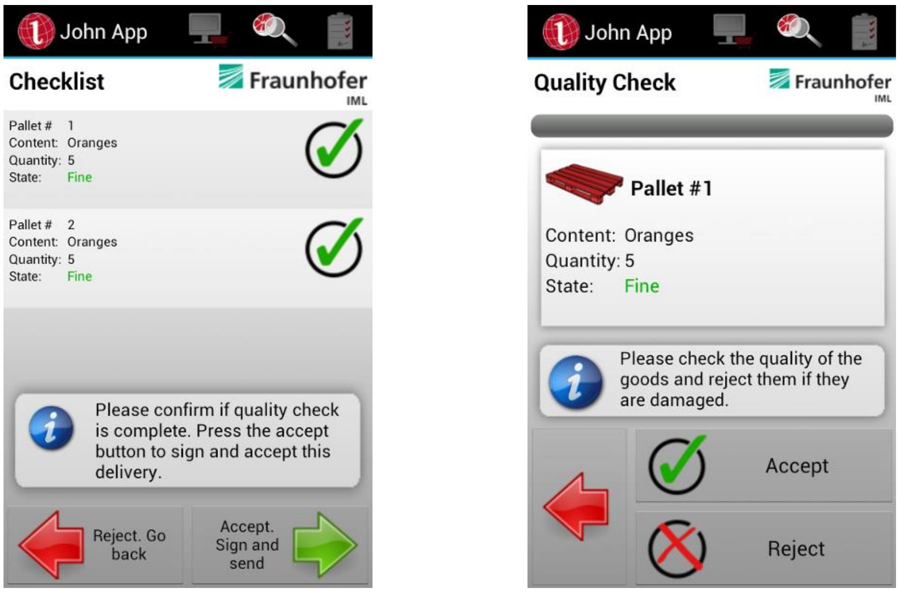


Figure 55: Smartphone App „Goods-In Assistant“

3.3.3.6 Feedback

Overall the three demonstrators of scene 1 to 3, which build a consistent logistics scenario, received a lot of feedback on the different exhibition occasions. During the demonstration of IoT week 2012 in Venice the prototypes of the transport monitoring and handover scenes were first shown. A. Greenfield saw a huge improvement in transparency for customers, which want to follow the transport chain and be sure the product is still safe to consume. The SmartAgriFood project (FI-PPP phase 1) was very impressed by the demonstrator as the scenario fits to one of their defined use cases. Further talks are ongoing on how to work on this topic within FI-PPP phase 2/3. Groupe Casino evinced large interest on creating a pilot for the shown monitoring use case. Further talks continued and a pilot is ongoing which may be seen in Section 6.1. Prof. Tangorra and S. Leonardi from University of Milan were interested to see how the use case may be adapted to RFID-based tracking in a slaughtering scenario.

The demonstrator for NFC-assisted check-in & loading was first shown at IoT week 2013 in Helsinki, where all three scenarios were shown. The biggest improvement regarding to the demonstration at IoT week 2012 was the integration of the resolution framework of WP4 and hardware components from WP5. The resolution framework showed to be a flexible tool, where we were able to realize scenarios which were not planned before. We managed to integrate NFC readers and sensor nodes with the Moterunner operating system.

3.3.4 Scene #4: Sensor Based Quality Control (SAP)

The implementation of this scene is based on the “Sensor Based Quality Control” use case scene 7 depicted in D7.2.

3.3.4.1 Step 1: Application Description

Within the store we support two important concepts: Dynamic pricing and quality control of perishable goods. Dynamic pricing as a real-time tool for price optimization strategies has always been crucial for profit maximization. In contrast to traditional systems, dynamic pricing is not performed on static information such as best before end dates in the transaction data of the backend ERP system, but it is based on real time IoT data gathered from a sensor infrastructure. Up to 20% of perishable goods never reach the consumer, but are disposed of before, either in the store or in the supply chain. The utilization of IoT sensors is therefore an interesting concept to implement quality control of perishables and thus reduce waste and increase profits at the same time.

The sensor based quality control business process estimates the future quality of the goods based on the luminance, humidity, and temperature of the environment. The dynamic pricing is then able to reduce prices, even before a perceivable degradation of quality occurs. By applying this sensor based quality control and combining it with dynamic pricing, it is ensured that the goods are sold before quality degradation is likely to occur.

3.3.4.2 Step 2: Domain Model Representation of the Demo

The domain model for this scene is shown in Figure 56. Hardware parts are shown in blue, software artefacts in green. Users are yellow, which in this example consist only of Ted the truck driver.

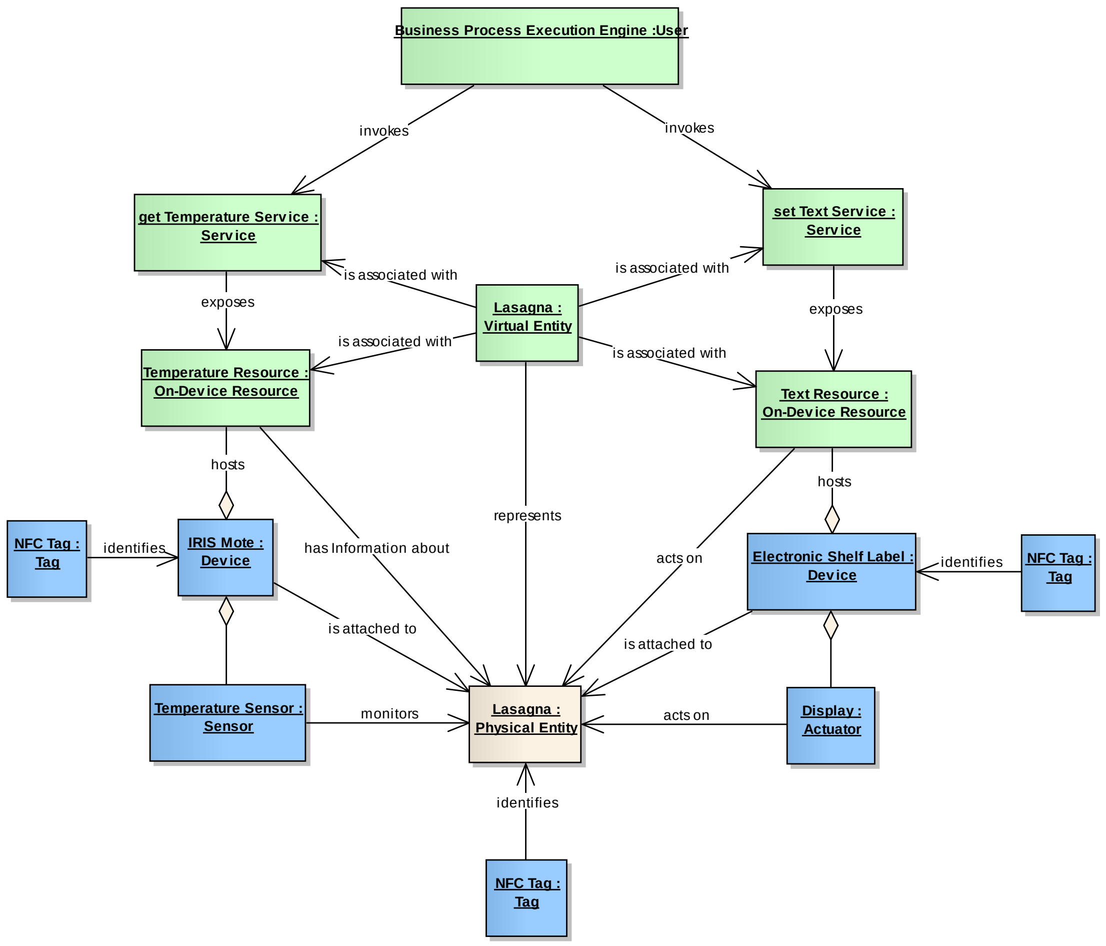


Figure 56: Domain Model of scene 4 – Sensor Based Quality Control

3.3.4.3 Step 3: Information Model Representation of the Demo

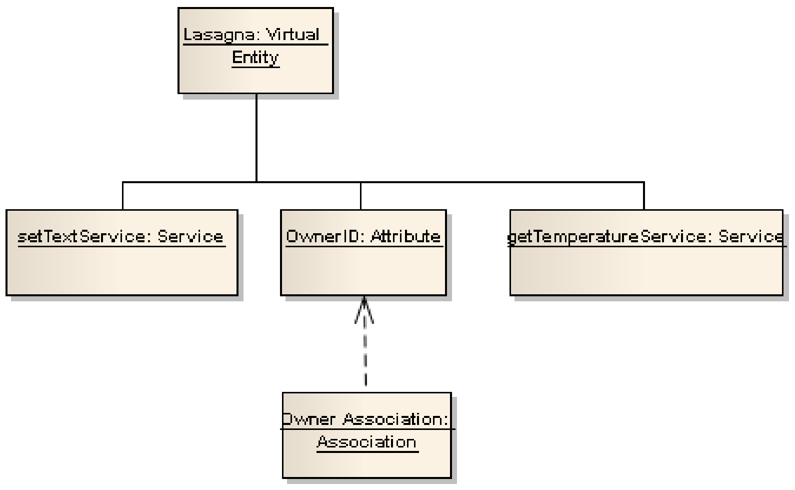
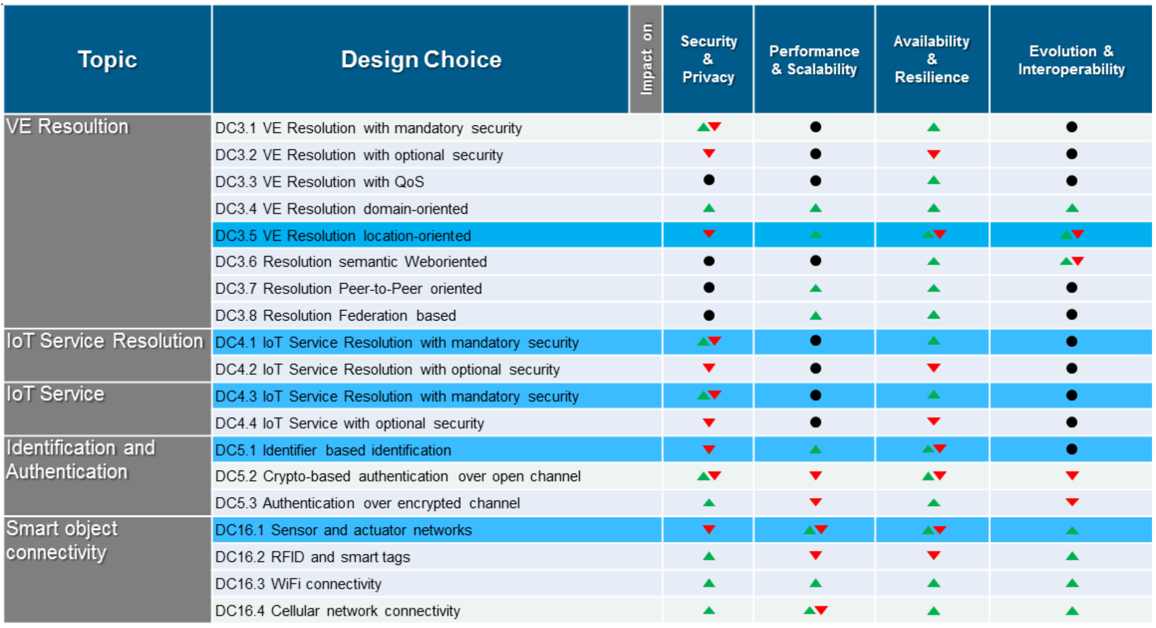


Figure 57: Information Model of scene 4 – Sensor Based Quality Control

3.3.4.4 Step 4: Relevant Design Choices for Implementation

Within the retailer's infrastructure, a location-oriented VE resolution was chosen as a design choice due to the distributed nature of multiple retail stores dealing with similar object types. While the IP based service functionalities can easily afford mandatory security, we do have a potential issue in the smart object space, as profitability demands the use of constrained sensor and actuator networks for which we have not implemented security measures. This is certainly an issue, but the only realistic choice. Accordingly, security measures / plausibility checks need to be performed above the IoT Service layer. Table 19 shows the picked design choices.



Topic	Design Choice	Impact on			
		Security & Privacy	Performance & Scalability	Availability & Resilience	Evolution & Interoperability
VE Resoultion	DC3.1 VE Resolution with mandatory security	▲▼	●	▲	●
	DC3.2 VE Resolution with optional security	▼	●	▼	●
	DC3.3 VE Resolution with QoS	●	●	▲	●
	DC3.4 VE Resolution domain-oriented	▲	▲	▲	▲
	DC3.5 VE Resolution location-oriented	▼	▲	▲▼	▲▼
	DC3.6 Resolution semantic Weboriented	●	●	▲	▲▼
	DC3.7 Resolution Peer-to-Peer oriented	●	▲	▲	●
	DC3.8 Resolution Federation based	●	▲	▲	●
IoT Service Resolution	DC4.1 IoT Service Resolution with mandatory security	▲▼	●	▲	●
	DC4.2 IoT Service Resolution with optional security	▼	●	▼	●
IoT Service	DC4.3 IoT Service Resolution with mandatory security	▲▼	●	▲	●
	DC4.4 IoT Service with optional security	▼	●	▼	●
Identification and Authentication	DC5.1 Identifier based identification	▼	▲	▲▼	●
	DC5.2 Crypto-based authentication over open channel	▲▼	▼	▲▼	▼
	DC5.3 Authentication over encrypted channel	▲	▼	▲	▼
Smart object connectivity	DC16.1 Sensor and actuator networks	▼	▲▼	▲▼	▲
	DC16.2 RFID and smart tags	▲	▼	▼	▲
	DC16.3 WiFi connectivity	▲	▲	▲	▲
	DC16.4 Cellular network connectivity	▲	▲▼	▲	▲

Table 19: Design choices relevant to scene 4. The highlighted choices are those picked for the demonstrator.

3.3.4.5 Step 5: Technical realisation of the Demo

In this section we present the actual integration of our monitoring and dynamic pricing business process into our living lab. The technical realisation of the sensor monitoring system in the store is shown in Figure 58. In the following we will shortly present these components and how they interact.

In order to make the wireless sensor nodes and the electronic shelf labels accessible, we use SAP's "Real World Integration Platform" (RWIP). The Site Manager is used for the configuration of the RWIP agents and the Node Runtime loads the configuration from the Central Instance for execution. The Mote Sensor Agent queries the motes for the environmental parameters and the Electronic Shelf Labels (ESL) Actuator Agent utilizes the Bounce WebService to display prices on the shelf labels. The REST Interface Agent provides a standardized interface to the motes and the ESLs. All IoT-related services and associations between services and entities are handled by the IoT-A Service Resolution Infrastructure. The central component of our setup is an IoT-aware Business Process Execution Engine, which is coupled with an SAP Retail System via a bridging component.

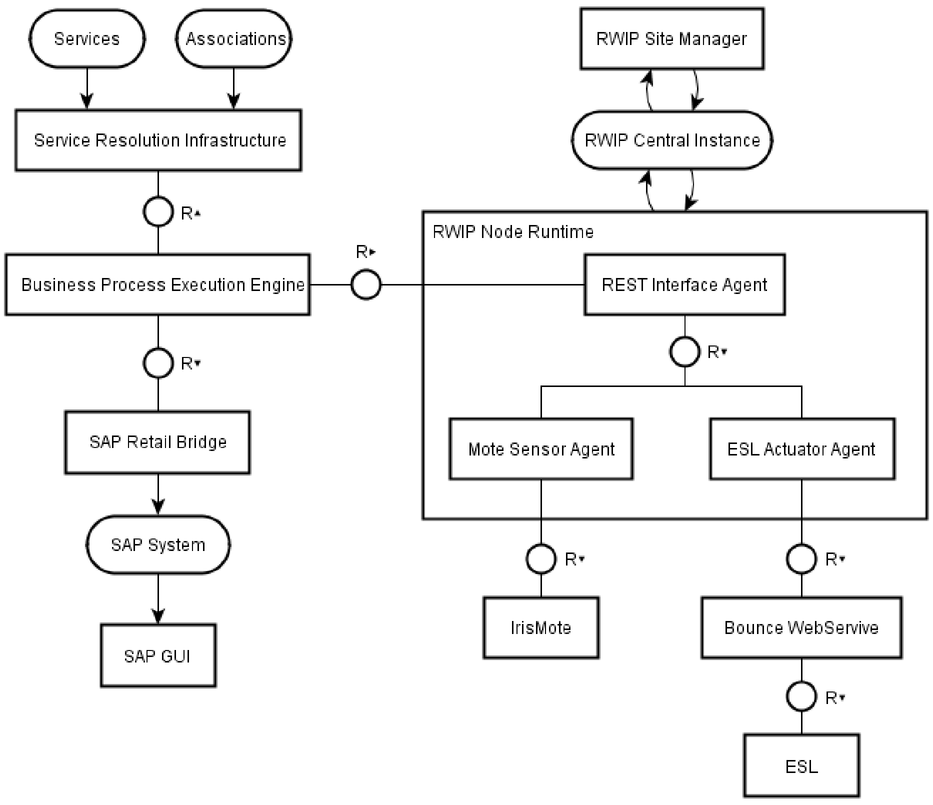


Figure 58: Technical Realisation in the Retail Domain

The insertion of IoT services into the Resolution Infrastructure is performed by the REST Interface Agent upon the appearance of new motes or ESLs. The association of motes and ESLs to the perishable goods in the store is performed using a mobile app. The store personnel scans the NFC tags attached to the mote (resp. ESL) and the one attached to the crate containing the goods and the mobile app inserts the association between the entity and the IoT service into the Resolution Infrastructure.

The adaption of the retail price is handled by an IoT-aware business process, which measures the environmental parameters and potential changes in the price. The process model contains declarative descriptions of the particular good for which the model should be executed and of the involved IoT-related operations. The execution engine generates a SPARQL query from these descriptions and therewith queries the resolution infrastructure for the implementing IoT services. Upon the execution of the process, the engine accesses the service endpoints, which are provided by the REST Interface Agent. Additionally, the execution engine writes the environmental parameters and the prices into the SAP Retail System.

Hardware-wise, we use Iris motes (see Figure 60), with technical details as is shown in Table 15. All motes are running on IBM Research’s Moterunner platform, with all business process steps encoded in Java. Our on-mote services are fully configurable through a RESTful interface. The actual communication is done with the constrained application protocol (CoAP) over 6LoWPAN.

All entities used in the system are semantically described to solve the interoperability and integration issues, which usually arises when combining artefacts from different vendors. We use the following ontologies:

- The sensor network landscape (W3C SSN) is used for modeling relationship between Entities and the monitoring motes

- Services running on the motes are described in Linked USDL

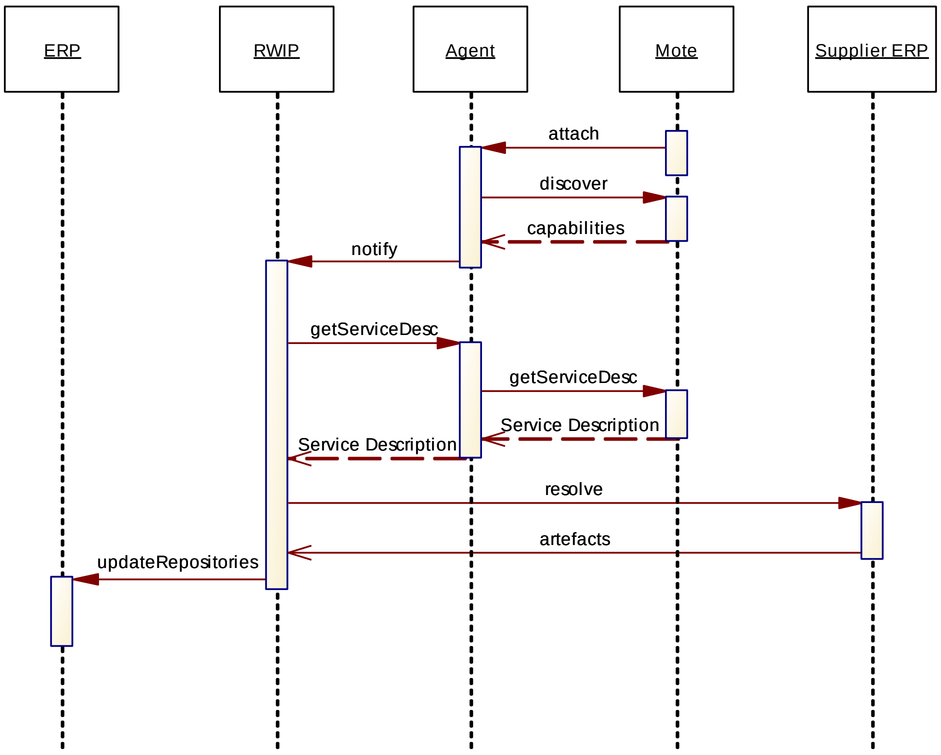


Figure 59: Automatic recognition of Motes

We support extendibility and scalability of the system through self-descriptive motes. Motes joining our system have a Linked USDL service description on them. This service description might be incomplete, and need to be completed with cloud support, but contain all the necessary semantic information for the system to automatically recognize the mote, determine its capabilities and add it to the retail system. Figure 59 shows the process of adding a mote to the system. First the joining mote notifies the edge mote and thus the Mote Agent about its appearance. In our implementation, this notification is through a 6LoWPAN TDMA beacon based protocol, which then triggers a custom discovery protocol. The discovery will identify the mote as includable and determine its capabilities. Next the mote agent will notify the RWIP, which in turn will ask for the services on the mote. The mote transfers the service description which is resolved on the RWIP-Agent, if necessary, and then the ERPs repositories are updated accordingly.

The used motes are shown in Figure 60, technical details may be seen in Table 20.

Figure 60: Iris Mote

CPU	ATMega 1281@8Mhz
Memory	8B RAM
Program Flash	128KB
Serial Flash	512KB
Current Draw	
Active	8mA
Sleep	8µA
Receive	16mA
TX	10mA(-17dBm), 12mA(-2dBm) 17mA(3 dBm)

Table 20: Technical details of an Iris Mote

The running demonstrator is shown in Figure 61.

Figure 61: Demonstrator at IoT week 2013, Helsinki

3.3.4.6 Feedback

The version presented in this document is technically radically different from the versions presented earlier, e.g. at the IoT-A year 2 review. The original feedback we received from the reviewers was that it was questionable in how far the system would scale to millions of devices if the respective business processes would not be able to fully automate the integration of additional devices. In that respect we have introduced complete self-description capabilities of the devices by using USDL descriptions stored on the motes themselves. This approach was presented e.g. at IoT-Week 2013 to the FI-WARE consortium and will be a foundation of SAP's contributions to the FI-WARE project. In that respect we have succeeded in transferring valuable project results towards a public private partnership. Other, important feedback we got was the question of choosing Moterunner and not e.g. tinyOS as a platform. We believe that Moterunner is the stronger platform, but probably we cannot completely disregard the factual market penetration of tinyOS in the future.

3.3.5 Scene #5: Low Insulin Supply (link to Health UC) (ALU-BE)

This scene is part of the Health use case scene #4. All the implementation specific details are found in Section 3.2.4.

4 Validation by Requirements

4.1 Background on Requirements

IoT-A aims at defining a Reference Architecture for IoT systems – that is "a matrix that eventually gives birth ideally to all concrete architectures [for IoT systems]" – the IoT Architecture Reference Model (ARM) [Carrez 2013].

The IoT ARM was derived from requirements, and as such, one dimension for validating the presence of the IoT ARM in the WP7 use cases would be to check to what extent the unified requirements are present. This link between the IoT ARM, requirements and the WP7 use cases are depicted in Figure 62**.** A more detailed description on these relations can be found in the Guidance Chapter of [Carrez 2013]).

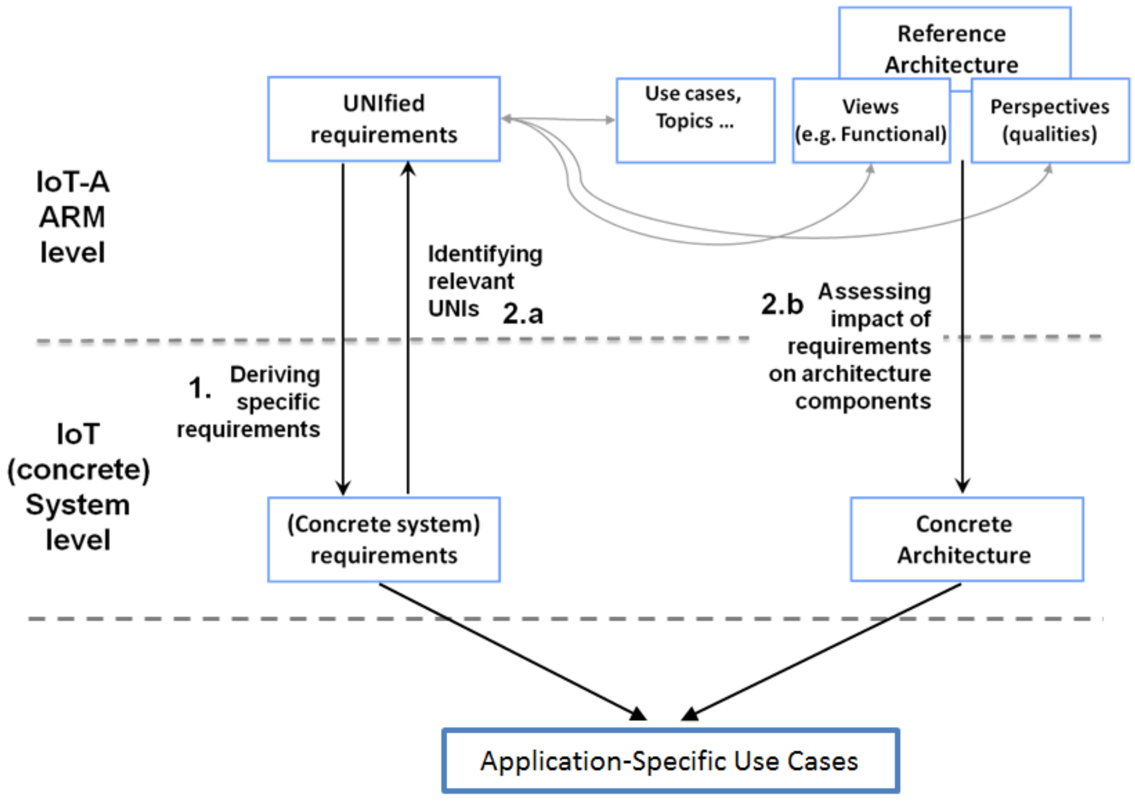

Reference Architecture
UNIfied requirements
Use cases, Topics ...
Views (e.g. Functional)
Perspectives (qualities)
IoT-A ARM level
Identifying relevant UNIs 2.a
2.b Assessing impact of requirements on architecture components
1. Deriving specific requirements
IoT (concrete) System level
(Concrete system) requirements
Concrete Architecture
Application-Specific Use Cases

Figure 62: Concrete system requirement and architecture vs. Unified Requirements/Reference Architecture

It should be noted that the definition and usage of requirements to drive such reference architecture work is significantly different from traditional requirement engineering practice. Just as the IoT Reference Architecture tries to abstract IoT concrete architectures by generalising common traits, the requirements in IoT-A (known as the "Unified Requirements") ***generalize solution-specific requirements*** and aim at providing the grounding for i) helping to define and validate the Reference Architecture, and ii) support the usage of the IoT ARM by a (concrete) system designer when deriving domain-specific architecture and requirements.

As a result, the requirements are derived both from existing solutions characteristics, i.e. state of the art, and from discussion with internal and external stakeholders. Since the requirements come from these two very different sources, the requirements appear in different granularities; some at a high abstraction level, while others are specific to a domain or system. Due to the diversity, it follows that not all of them can be fully represented in any given use case, including those in WP7.

Nonetheless, the WP7 use cases aim to instantiate as much as possible the core ideas of the IoT ARM and the underlying. In doing so, we validate that the IoT ARM can be applied to derive real world architectures and use cases.

Figure 63 shows the time line of the project and the periodic updates of the requirements and use cases. The requirements were updated at three time periods; these were given as inputs into the use case development. Since the use cases reached a stable state in “D7.2 Exact Definition of the Use Cases”, the core addressed requirements are those available at the time, found in D6.2. The few requirements which were subsequently defined in D6.3 were addressed where possible. For more details on the requirement gathering process, the interested reader should look at D6.3, the Final Unified Requirement List [IoT-A D6.3 2013]).

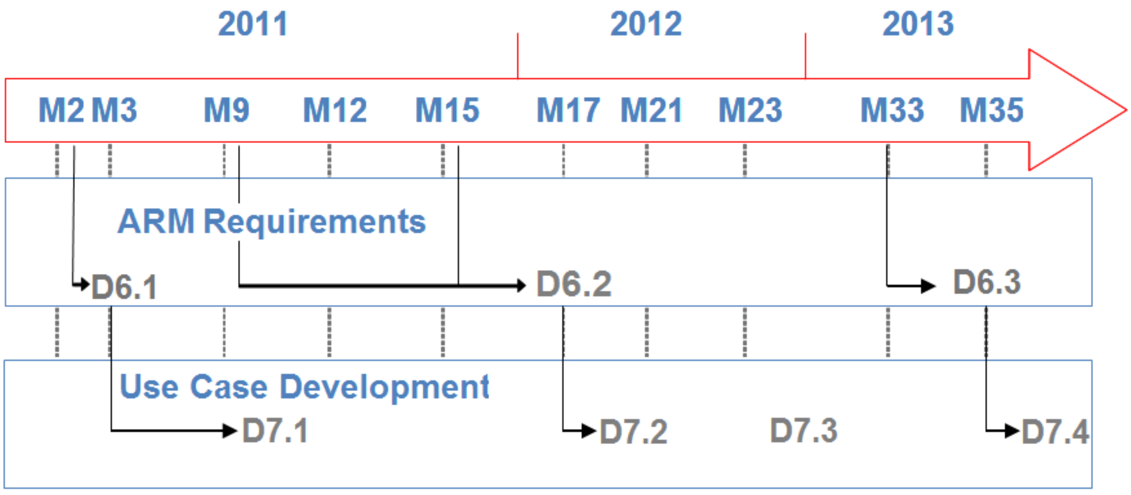


Figure 63: Timeline of the project and the cyclic updates of the requirements and use cases

4.2 Validating the Requirements in the Use Case

Upon completion of D7.4, we conducted an exercise to check to what extent the requirements are present in the use cases. In each of the use case scenes, the partners went through all 184 unified requirements, and indicated for each one:

- Whether the requirement is applicable to their scene (yes/no)
- And if it is appropriate, then indicate to what extent the requirement was implemented in the scene (0 = not at all, 1 = implemented)

In doing this exercise, we could evaluate which requirements were important to our use case scenes, and which ones were implemented. Although the space of possible use cases is infinite, this exercise done in the space of our two use cases allows us nonetheless to get a qualitative picture of which requirements are important in an IoT system, and which ones are readily applicable to health and retail.

The actual requirements and their flagged status are in Appendix A; a summary breakdown of the initial result is shown in Figure 64**:**

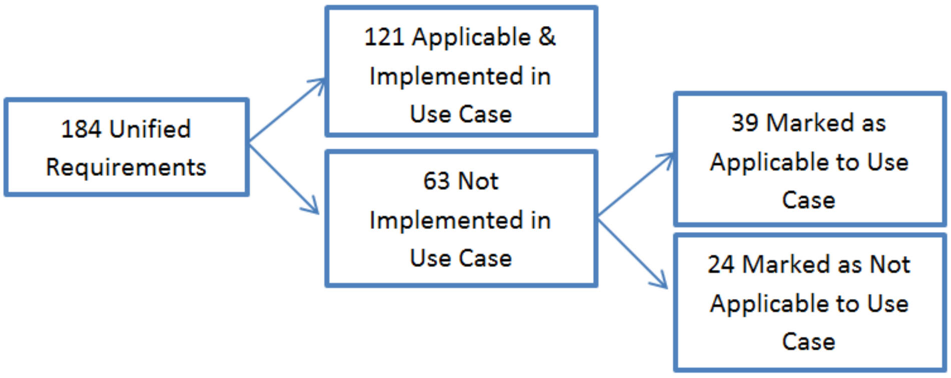

184 Unified Requirements
121 Applicable & Implemented in Use Case
63 Not Implemented in Use Case
39 Marked as Applicable to Use Case
24 Marked as Not Applicable to Use Case

Figure 64: Break down of the requirements and their applicability and implementation in use cases

From the results, one sees that only 13% of the requirements (24 requirements) were considered not applicable by the consortium partners to their particular use case scenes. This means the majority of the requirements were relevant to our WP7 IoT system, a healthy reflection on the breadth and relevance of the requirements.

For the non-implemented requirements, by inspection we see that many of them describe specific functionalities which were not necessary in our particular use cases (ex. UNI.251 “The service organization shall provide a feedback to the user who sent a composition request” or UNI.405 “A system built using the ARM shall allow programmers to add new coordinate reference systems and shall support the transformation of coordinates among them”) or were out of scope for a demo (ex. UNI.601 “A system built using the ARM shall guarantee infrastructure availability” or UNI.702 “A system built using the ARM development shall support iterative approaches (e.g. spiral model)”). As such, it did not make sense to include them.

About 21% (39 requirements) were considered important to the use cases but were not implemented. We see that many of these requirements reflect design choices on robustness (ex. UNI.089 “A system built using the ARM shall support reliable time synchronization” or UNI.099 “A system built using the ARM shall guarantee correctness of resolutions”) service quality (ex. UNI.237 “A system built using the ARM shall offer services for the retrieval of quality of information related to virtual entities”) or security (ex. UNI.625 “A system built using the ARM shall provide a device security and privacy measurement”). Although important, as each of these very diverse requirements each imply complex implementation in of themselves, not all of these could be feasibly implemented in a demo given the resources of the project. Additionally, the timing of the requirements – particularly the security (UNI.6XX series) and management requirements (UNI.7XX series) – late in the project (month 33) prevented a full implementation of each of these requirements as by then the use cases had already been strongly developed.

Thus, the overall coverage of requirements is very positive: we see that a majority of the requirements (66%) were considered applicable to the WP7 use cases and therefore implemented. Figure 65 gives an overview on the distribution of implemented and not implemented requirements.

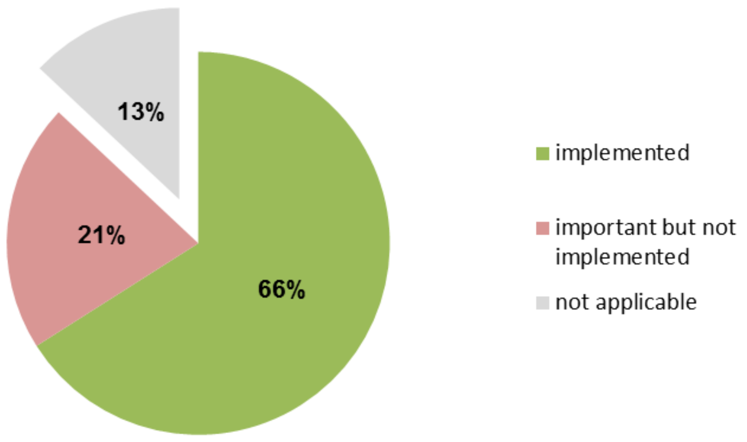


Figure 65: Overview of implemented and not implemented requirements in demonstrators

The high coverage of requirements also showed that the choice of retail/logistics and healthcare as domains for demonstrating the ARM was appropriate, as it covers both the technical requirements and the diverse spread of aspirations from heterogeneous stakeholders.

5 Business Analysis of Use Case

5.1 Introduction

This section summarises the core results of the business case presented in D6.4. It is mainly interesting as these results base on use cases partly developed in WP7 and thus have a relevance for the validation of the use cases from a financial perspective.

We take two approaches in validating the business value of the ARM. In the first approach – the inductive forward development approach – taken for the retail business case, we look at use cases that were developed from the ground up and had used the ARM explicitly as guidance. Accordingly, we select several use case scenes from WP7 and evaluate their business value. In doing so, we show that the ARM can assist development of IoT use cases which lead to value, and establish internal validity.

In the second approach – the reverse mapping approach - taken for the health care business case, we focus on an already implemented IoT system, the MUNICH IoT platform. We first note note that in D1.5, a reverse mapping exercise was conducted on the MUNICH IoT platform to show that the ARM could describe and help realize such a system. We then show in this section the benefits of the MUNICH IoT platform. Combining the reverse mapping and the cost-benefit analysis conducted here, it then follows that the ARM can help realize IoT systems of value, and not necessarily systems internal to the IoT consortium, thus establishing external validity.

Cleary, not every instantiation of an ARM-based system would necessarily be a system of real world value, but this exercise would show that it sufficiently describes core concepts that can lead to real world value, thereby demonstrating the ARM's relevance.

To be able to evaluate the performance of the two use cases, it is necessary to define a business case framework. This framework builds the structure, which guided the overall business case process (see Figure 66). The goal of the framework is to deliver a decision support tool and highlight opportunities and potential risks [Bruegger, 2009].

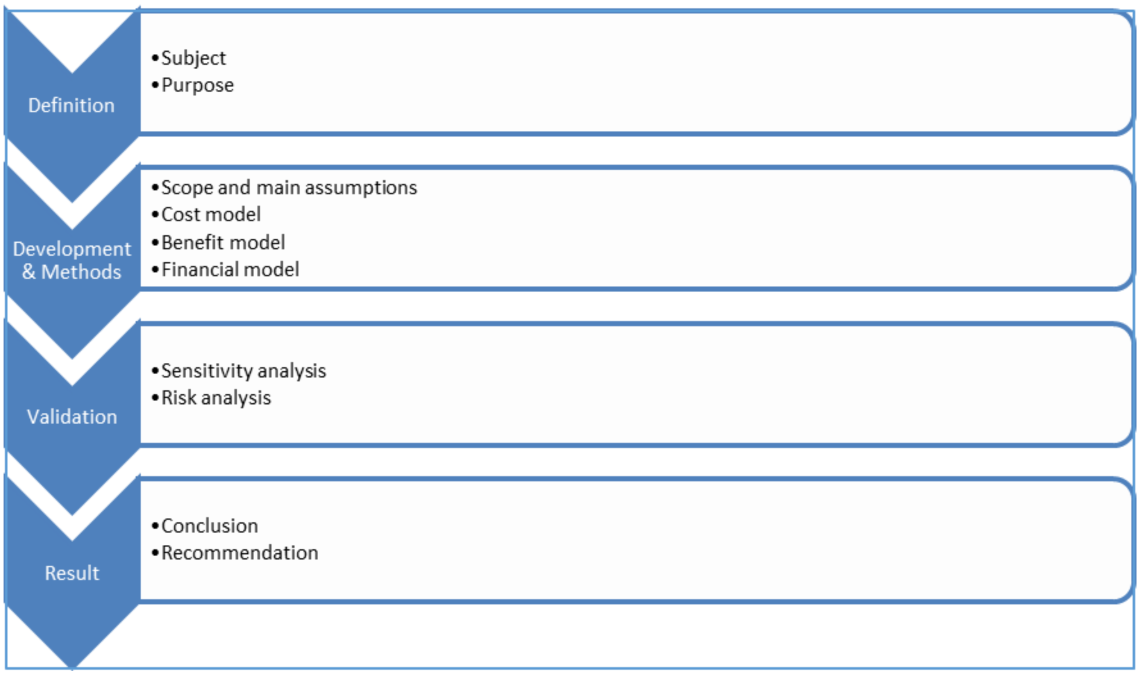


Figure 66: Business case process

In the first phase – the definition – a basic understanding, the purpose and goals of the business case must be defined. This step lays the foundation for the following steps and is always checked against the intermediate results. Subsequently, the development of the individual models and the corresponding methods are conducted. This step encompasses a definition of the scope and main assumptions of the business case which ensures that the business case on the one hand has a clear frame in which it is calculated and on the other hand the main reasonable assumptions which lead to the end result. The three models – cost, benefit and financial – are parts of the business case to define all the relevant information corresponding to each of the models. In the cost model the main cost drivers are described and implemented while the same holds true for the benefit model with respect to the main benefit drivers. The financial model combines both the cost and benefit model and calculates the financial impact based on common performance measures such as return on investment (ROI) or Net Present Value (NPV). As these calculations are built upon certain assumptions the validation step highlights the deviations if some of the assumptions with high impact are changed. Therefore, the sensitivity analysis shows a range which is framed by the best and worst case scenario. As a preceding step in the risk analysis the influencing factors which constitute major risks are identified and form input for the sensitivity analysis. The final step finishes with a conclusion with recommendations based on the results of the business case calculation.

5.2 The Retail Business Case

5.2.1 Background

The business case was calculated in the context of a virtual supply chain which bases on three scenes of the retail use case, namely the following scenes:

- NFC supported check in and assisted loading (scene 2)
- Transport monitoring with Smart Load Carriers (scene 3)
- Sensor Based Quality Control (scene 4)

The main problem statements for the retail business case can be summarized into three major groups. The three main groups are:

- Software development
- Transport of perishable goods
- Customer satisfaction

The software development is particularly important as it is the process before the introduction of the IoT system. The costs of developing such a system impact the final result of the business case significantly and in this phase the IoT ARM has its biggest value as we will compare the final result for two cases – the development with and without the IoT ARM. Thus, the value of having a common ground like the IoT ARM to build interoperable systems which will be easier to maintain will be revealed in this comparison. Furthermore the transport of perishable goods is a good example for a suitable use case as in this scenario IoT technologies such as temperature sensors can lead to a better performance in terms of less waste because of temperature issues during the transport. The former will then lead to a higher customer satisfaction as at the point of sale the temperature-controlled environment is able to estimate the actual quality of the perishable goods which can be on top combined with a dynamic pricing system.

Besides short-, medium-term objectives and long-term objectives can be accomplished [Bruegger, 2009]. The focus of this business case is on the short- and medium-objectives, which are directly linked to the investment project. However, the usage of a standardised architecture generates long-term objectives as well.

The short- and medium-term objectives of the business case are aggregated into four main categories as depicted in Figure 67. In the following each will be explained in more detail.

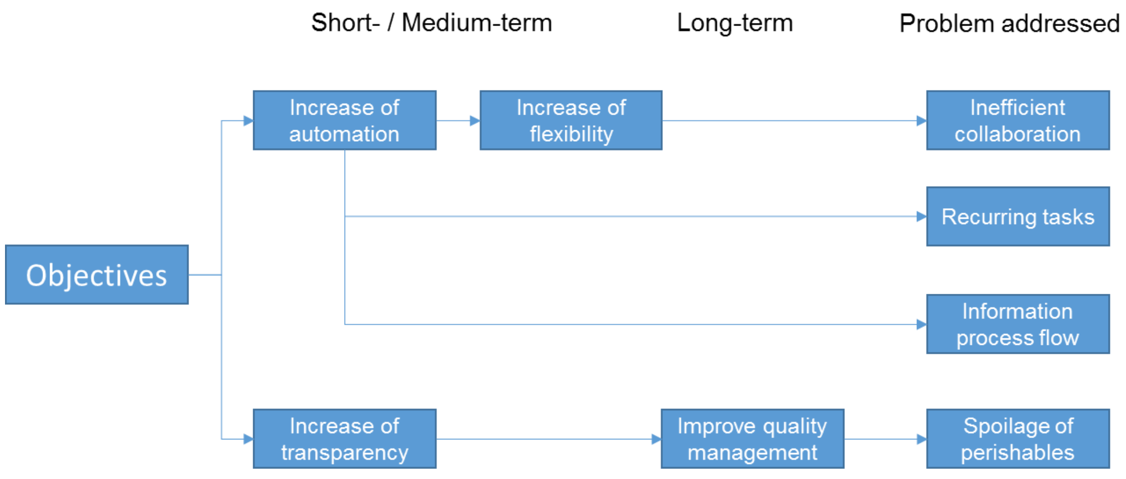


Figure 67: Short-, medium-, and long-term objectives

The short-term objective “Increase automation” addresses two problems in the cold chain. First, novel IoT technologies automate the information process flow in the cold chain and distribution. Second, the reduction of recurring tasks such as price labelling, quality control and load/unload control with support of IoT technologies increase the rate of automation.

Based on the increase of automation the objective “flexibility” emerges. The goal is on the one hand to increase the flexibility in the cold chain based on more and better information. That allows flexible order scheduling and planning as well as the integration of new partners in the supply and delivery chain. A further goal is the increase of transparency in the cold chain as a short-term objective. This mainly concerns the synchronisation between physical entities and their virtual representations, namely virtual entities, in the information systems. In this case not only real-time data of one parameter, e.g. temperature, increases the transparency of the cold chain rather than a combination of different parameters, e.g. temperature and humidity.

The last objective is related to quality management. The requirements of the cold chain take into account environmental conditions during transports. The target is to improve the quality control along the chain with constant and real-time monitoring enabling the participants to react and solve problems instantly with the objective to reduce spoilage and loss of perishables.

Apart from short- and medium-term objectives companies involved in a cold chain follow long-term objectives. In a sensor-based cold chain the sensors and other IoT technologies produce a lot of data. The provided information is particularly useful if it is shared between all partners.

5.2.2 The Business Case

5.2.2.1 Cost and Benefit Models

As input factors for the business case a cost and benefit model were defined. The costs can be divided into non-recurring and recurring cost. The former are mainly investment costs incurred in the beginning of a project, e.g. software development and initial hardware costs. The recurring costs can be interpreted as the operating costs of the system. The biggest share of the operating costs are the maintenance and service cost for software applications and different hardware devices.

The benefits can be split in two main categories, these are the tangible and non-tangible benefits. Subsequently, both are explained in more detail.

Tangible benefits are typically those benefits one can be directly or indirectly measured and monetarised. Directly measurable benefits are those which are directly generated due to an investment, e.g. lower costs for support or maintenance. Indirectly measurable benefits are those which are only indirectly generated, that means the financial impact is not immediately identifiable, e.g. higher customer satisfaction leads to higher revenues.

Non-tangible benefits are mainly based on subjective and thus not measureable benefits which often includes hypothetical assumptions, e.g. higher flexibility (how can one use this flexibility?). As an example one can regard smart things which are spread across the process and generate new data and information. These have to be analysed to which extent they cause business value [Bucherer, 2011]. New warehouse management and distribution scheduling based on better quality information can also contribute to long-term goals such as acting as a sustainable company by reducing the volume of disposals [Dada, 2008]. There exist further effects in customer relation management if data is enhanced with meta-information a fully automatic reporting and analysis is possible. Trends and customer wishes can be discovered and individual offers can be provided to improve the customer satisfaction [Bucherer, 2011].

The tangible benefits are calculated on two different ways. The first method considers the benefit equal to the saved cost:

$$Benefit\ (B) = C_{saved\ cost}$$

The second method compares the cost in the state of the art process with the cost for the new process.

$$Benefit\ (B) = C_{goal\ process} - C_{state\ of\ the\ art\ process}$$

The start of the benefit realisation depends on the used software development estimation, either with or without the support of IoT ARM. The differences between with and without IoT ARM is in the first year around 570,000 € and in the second year around 370,000 €. After the second period both models achieve the same amount of yearly benefits as the underlying assumption is that both solutions perform similarly. Overall the project using IoT ARM has higher cumulative benefits of nearly 938,000 € or 13% after all considered business case periods.

5.2.2.2 Cost-benefit anaylsis

Figure 68 shows the cash flow development over the business period of six years. The implementation without IoT ARM has a lower negative cash flow in the first period, but after the second investment year the case with IoT ARM demonstrate a superior business performance for the remaining analysis periods. From the third period onwards the cash flow in both development scenarios are approximating with minor advantage for the scenario with IoT ARM.

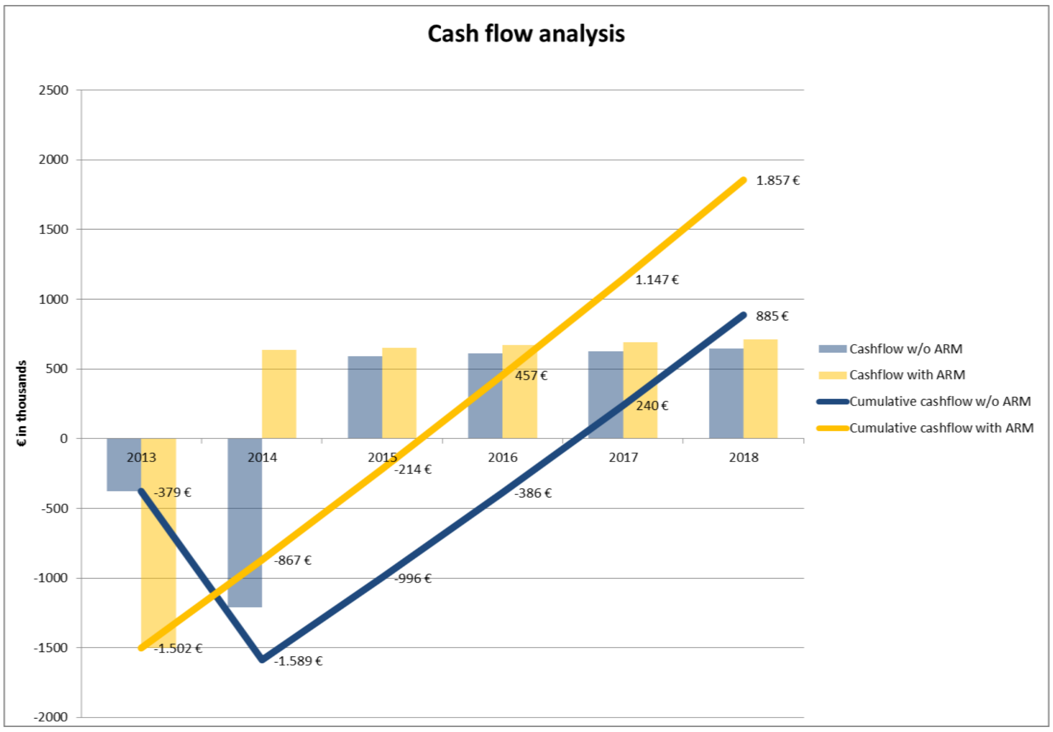


Figure 68: Cash flow analysis

5.2.2.3 Sensitivity analysis

The positive result of the business case calculation shows a clear benefit for the usage of IoT ARM. In this chapter, the robustness of the business case model will be analysed in respect of possible changes of main influencing factors. The sensitivity analysis includes the following aspects:

- Higher and lower internal interest rate (6%, 10% 12%)
- Raise of the risk factors (+10%, +20%, -10%)
- Longer software development time (20%, 50%)
- Benefit robustness (-10%, +10%)

In the best case scenario all critical risk factors are decreased by 10% and additionally we assume the project can be realized 10% cheaper. Likewise, the discount factor is reduced from 8% to 6%. The results show positive increase for the business performance. The best case scenario is driven by two assumptions. First, the cost for hardware will decrease over the time. Second, the cost savings can be achieved, due to standards for the architecture and technologies.

The worst case scenario simulates the situation that the stakeholders demand a higher safety margin, which increases the discount factor to 10%. Further the software development time increases by 25% and leads to a longer total project time. All benefits are reduced by 10%. The results are displayed in Figure 69. In this worst case scenario only the case with IoT ARM can save a positive net present value around 268,000 €. In the case without the IoT ARM the result shows a negative net present value of -409,000 €. The worst case scenario is based on the assumption that software development is still a risk intensive project. Additionally it is assumed that the benefit realisations do not behave as calculated, due to smaller margins.

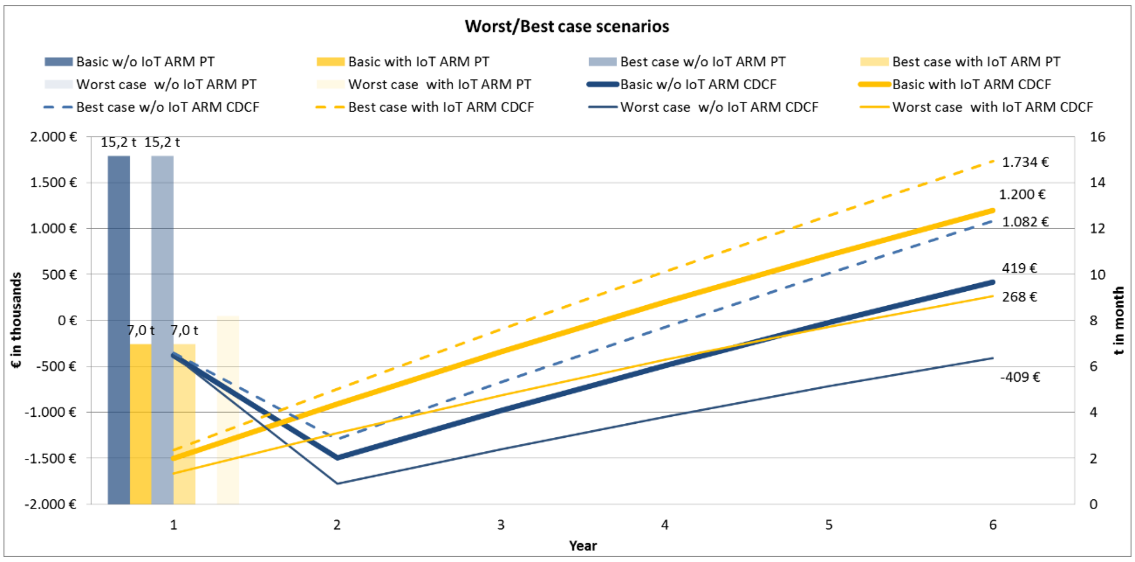


Figure 69: Worst/best case scenarios

5.3 The Healthcare Business Case

5.3.1 Background

In the health care use case, we focus on an already implemented IoT system to show the real-world value of the ARM. In combination with the reverse mapping in D1.5, we show that not only can the IoT ARM describe existing IoT systems (and by extension, help realize 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, namely debris left in the human body after surgery and time consuming process steps without added value (“non productive time”). A third auxiliary problem is the ongoing integration of software and solutions from 3rd party providers, which the IoT-A ARM would address.

Regarding the debris problem, in spite of already implemented safety checks debris (tools, towels, consumables) left in the body still occurs in 1:10.000 cases [Kranzfelder et al. 2001] during surgical procedures. 70% of the debris come from surgical towels, and 30% come from remaining surgical equipment [Kranzfelder et al. 2001]. The consequences for the patient are 40% morbidity rates and 5% is the mortality rates [Kranzfelder et al. 2001]. Regarding non-productive time, this refers to steps like documenting and registering towels in pre-operation, subsequent counting of towels during operation, and searching for towels when something is amiss; none of these steps add value, but instead address a problem created from the process itself.

Accordingly, a solution that addresses the tracking of surgical towels would sharply mitigate these problems. Given these problems, the MUNICH platform’s objectives and solutions can be mapped as shown in Figure 70:

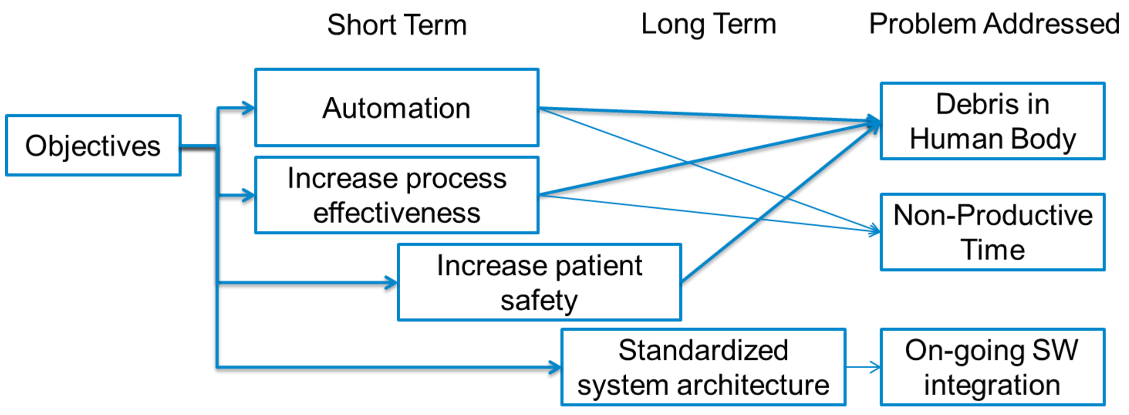


Figure 70: Objectives of the health care use case and the problems addressed

Real-time monitoring and location of all towels reduces the risk of debris in the human body, because of manual error prone counting and searching is avoided [MUWS 2013]. Therefore, the automation reduces manual errors. The process improvement raises the transparency of the process and reduces the risk of documentation errors which also can lead to debris in the human body. The experts estimate that a 100% failure protection is possible with this solution [Kranzfelder et al. 2001]. 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 removes the error-prone steps of documenting and registering towels in pre-operation, subsequent counting of towels during operation, and searching for towels when something is amiss.

For the long term problem of integrating new software developments from the hospital and their 3rd party solution providers, the IoT-A ARM provides a standardized reference architecture. This would simplify the complexity of the architecture and make integration of new components into the system easier.

5.3.2 The Business Case

5.3.2.1 Cost and Benefit Models

The inputs to our analysis consisted of a cost model and benefit model. The cost model factored in non-occurring costs (NRC) such as the RFID antenna and readers. The main cost driver is the hardware investment for the RFID antennas, which total 49,500€: 58% of the total non-recurring cost (85,600€). Beyond this initial investment, the cost model also factored in reoccurring 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 important impact of the cost model and counts for 98% of the yearly RC of 1,034,000€. A price change of 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 5.3.2.2 Cost-Benefit Analysis).

The benefit model is composed of three benefits; the calculated yearly benefits are in brackets:

- RFID supported surgery (815,000€) – the main benefit arises from controlling for and searching for towels, saving about 1000 towels per year

- Cost savings from prevention of surgical errors (370,000€), the main benefit arises from averting death and non-fatal incidences from leaving a surgical towel in a patient
- RFID supported surgery preparation (104,000€): The MUNICH expert team estimate a time saving of 5 minutes per surgery, which totals to 1667 hour

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 per 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 5.3.2.2 Cost-Benefit Analysis).

5.3.2.2 Cost-Benefit Analysis

In Figure 71, the yearly and cumulative cash flows are presented. The cost-benefit analysis demonstrates a positive investment result. The discount factor is assumed with 8% and the net present value is 805,000€. The payback period is below one year. Within Germany, according to healthcare experts, this would meet the requirement of a one year payback period for new investments in a German hospital.

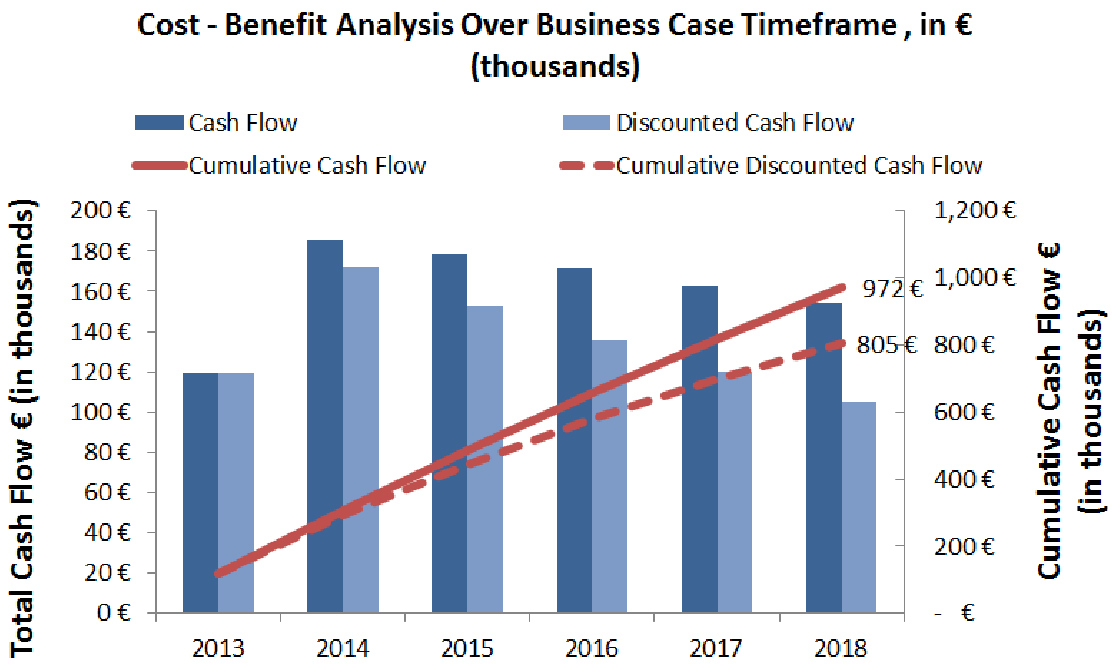


Figure 71: Cost-benefit analysis over business case timeframe (Health care case)

5.3.2.3 Sensitivity analysis

With the sensitivity analysis, we can investigate the impact of changing the major calculation variables. The following impacts shown in Table 21 will be discussed:

Model Tested	Model element changed	Change	Resulting discounted cumulative cash flow (original: 805,000€)	Remark
Cost Model	Critical risk factors: • software risk = SR • hardware risk = HR • personnel risk = PR, • maintenance risk = MR)	**CRF -10%**	**1,187,000€**	**Sensitive. Although the results are sensitive to the fluctuations in the reoccurring and non-reocurring costs, a 20% increase seems unlikely.**
		CRF +10%	**423,000€**	
		CRF +20%	**41,000€**	
	System service fee (SFS)	**SFS + 10%**	**270,000€**	**Very Sensitive. Moving from a SFS of 20€ per surgery to 23€ (15%) removes all benefit.**
		SFS + 15%	**0€**	
Benefit Model	Benefit variation factor (BSF)	**BSF +10%**	**1,453,000€**	**Sensitive.** The results are sensitive to the fluctuations in the benefit.
		BSF – 10%	**158,000€**	
		BSF -12.4%	**0€**	
General Assumpt-ions	Discount Rate (DF)	**DF +10%**	**772,000€**	**Insensitive.** The results are insensitive to the general assumptions of the models and do not pose a threat to the overall cash flow.
	Frequency of surgeries (TAoS)	**TaOS -25%**	**524,000€**	
		TaOS + 25%	**1,087,000€**	
		TaOS -75%	**0€**	

Table 21: Models and parameters varied in sensitivity analysis of the health care case

We note that the overall benefit is insensitive to the general assumptions, but sensitive to changes in cost and the benefits. For the critical risk factors, it is unlikely that it would rise as abruptly as 20% at once, and notably the cash flow still remains positive even in such a case. Therefore this is not a great threat. The system service fee is very sensitive, but can likely be managed and negotiated between the hospital and the service provider. Therefore, the only remaining threat is the benefit variation factor. Out of the three components of the benefit model - RFID supported surgery, cost savings from prevention of surgical errors, and RFID supported surgery preparation – the component that has the strongest potential to lowering the overall benefit was the cost savings from prevention of surgical errors; if the baseline of errors today were less, then the relative benefit of having this IoT system would be less.

Best -/ Worst case scenario

By combining cost and benefit variation in the sensitivity analysis, best- and worst case scenarios can be elaborated. For example in case the system service cost is reduced by 1 €/surgery (= -5%) and the hospital performs 25% more surgeries annually than the net present value raises 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 cost, additional development support 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 efficiency, and accordingly, the number of surgeries per year rise.

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 Figure 72).

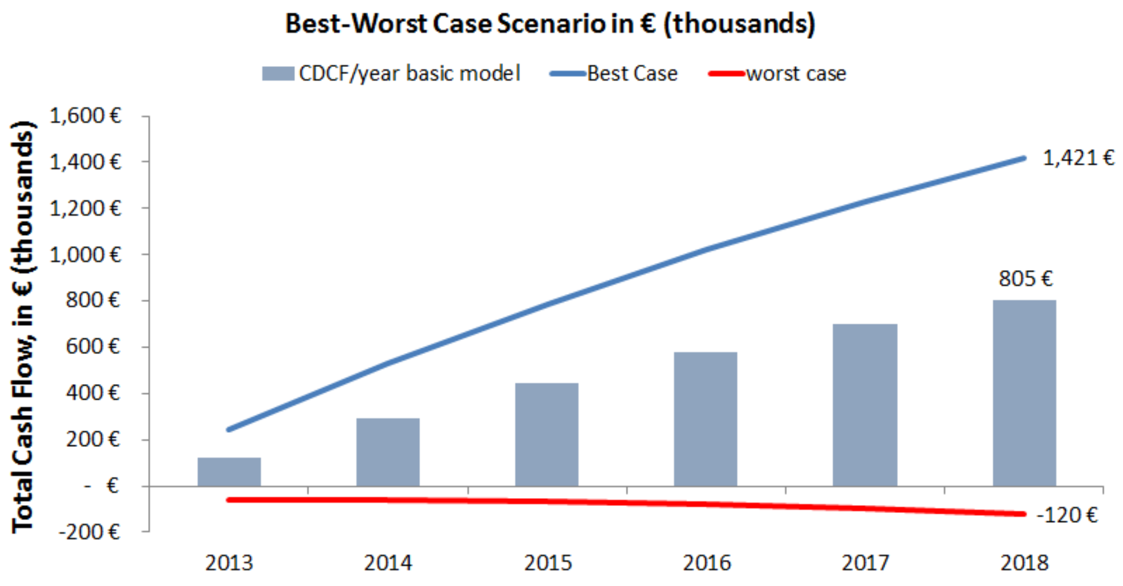


Figure 72: Best and worst case scenario (Health care case)

We observe that the economic feasibility of the case depends on a high degree of 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 are 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 are possible. The service provider would then also have better information for the calculation of the cost for service fee.

5.4 Conclusion

The business case for the retail / logistics use case shows that the usage of the IoT ARM in a project developing an IoT solution can be advantageous. Even though in the first period of the project the initial costs are higher than without the IoT ARM, this fact goes into reverse from the second period on so that the case with IoT ARM generates higher profits. As most of the assumptions in the business case are subject to variation we conducted a sensitivity analysis to reveal the range in which the financial results are included. In a nutshell one can claim that an IoT solution based on the IoT ARM not only serves in terms of technical advantages such as higher interoperability but also might generate higher profits.

We showed in the health case that the overall benefit is positive for the hospital, within the bounds of probable parameters in a sensitivity analysis. Beyond the monetary benefit, the use of an IoT system like MUNICH also averts death and non-fatal complications arising from leaving a towel in a patient, is likely to improve the reputation of the hospital due to a better track record, and improve efficiency of processes.

6 Exploitation by stakeholder group

One of the strongest means of validation for the IoT-A project is the application of IoT-A results to real-world problems and issues of stakeholders outside the project. We are currently evaluating the IoT ARM and the concrete technologies developed in the project with stakeholders in both of the major use case domains.

6.1 Groupe Casino: IoT-A Retail Exploitation

Within retail, we have been in contact with Groupe Casino since the beginning of the project, when Groupe Casino participated in Stakeholder Workshop 1. Groupe Casino is a French mass-retailer with its head office in Saint-Étienne. It operates in eight countries, generating net sales of €41.971 billion and employing 318,600 people. In 2012, Groupe Casino had more than 12,000 outlets (126 hypermarkets, 446 supermarkets, 2,476 discount stores, 6,457 convenience stores and 299 restaurants) in France and abroad, often through interests held in local retailers.[1] Groupe Casino distributes fresh and frozen goods through several of its own distribution centres (DC) all over the world.

One of the core problems of Groupe Casino relates to continuous and real-time monitoring of the cool chain, a problem that IoT-A has been working on intensively. Consequently, we have discussed the application of technologies based on the joint IoT Week 2012 demonstration of Fraunhofer, SAP, and IBM with Groupe Casino in several phone conferences that followed a joint meeting at IoT Week 2012. On July 23, 2013, a delegation of IoT-A (Martin Fiedler (FhG), Guenter Kuelzhammer (VDI-VDE), Carsten Magerkurth (SAP), and Rob van Kranenburg (Stakeholder Coordinator) have finally visited a distribution centre of Groupe Casino in Auxerre, France (see Figure 73), in order to evaluate the feasibility of a real-world pilot installation. The result is a concrete project plan that will be presented in the following section.

Figure 73: IoT-A representatives at the Auxerre Distribution Centre of Groupe Casino

[1] Source: Wikipedia, Groupe Casino

6.1.1 Background and Goal

The Cold Chain DC for the IoT-A trial installation is based in Auxerre, France. The DC has four different warehouses containing cooled/frozen goods for Hyper, Super, Proxy and D.P.G. markets. Easydis as a 100% daughter of Groupe Casino is the logistics provider and offering storage and transport services.

The goal of the Groupe Casino Cool Chain Monitoring pilot is to utilize IoT-A technology in a real world setting at an industrial partner who is not itself part of the project consortium. Based on the cool chain monitoring use case developed in work package 7, we will work on solving the problem of continuous and real-time monitoring of the cool chain from the distribution centre of Groupe Casino in Auxerre, France, to the delivery at a retail store location.

6.1.2 Scope

The pilot will involve installing a sensor device for measuring the temperature of frozen food inside a load carrier. The sensor will continuously measure the temperature and will establish a wireless connection to a GSM device / mobile phone that will forward the measurements to a monitoring server via GSM. The monitoring server will allow for tracking the history of sensor measurements for any given sensor device via a web interface accessible from any Internet connected computing device. Additionally, an interruption of the cool chain will trigger an alert both at the mobile phone and the monitoring server, so that appropriate measures can be taken in real time.

Figure 74 shows the adapted technical setup from the IoT-A “Transport Monitoring” scenario.

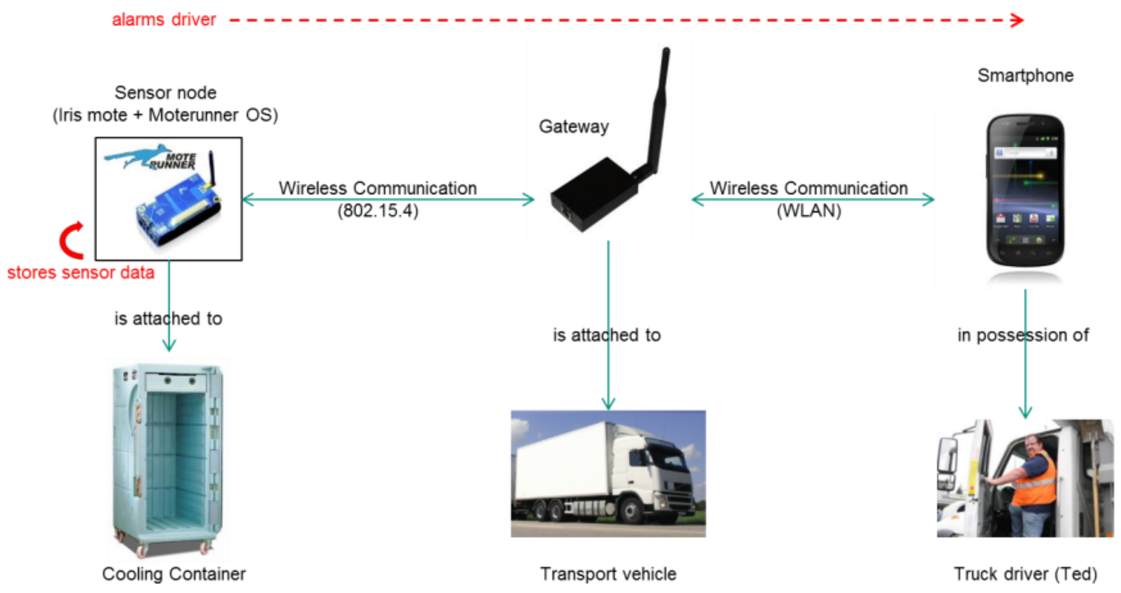


Figure 74: Adapted Transport Monitoring Scenario (retail UC scene 2)

6.1.3 Implementation

The project will involve the following implementation activities:

- physical installation of a sensor device at the load carrier / cooling container
- testing connectivity and reliability under real conditions
- development and installation of a mobile phone software component (FhG)
- development and deployment of a backend server component (SAP)

6.1.4 Schedule

The project will run from August to November 2013. The rough schedule is:

Work Package	August				September					October				November				
	31	32	33	34	35	36	37	38	39	40	41	42	43	44	45	46	47	48
WP1: Proof of concept and pilot definition	▨	▨	▨	▨	▨	▨												
WP2: Software development					▨	▨	▨	▨	▨	▨	▨							
WP3: Installation of hardware and software components										▨	▨	▨	▨	▨				
WP4: Test and documentation														▨	▨	▨	▨	▨

Table 22: Schedule for Groupe Casino Pilot

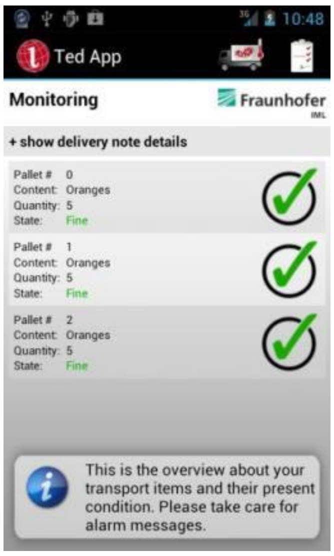
6.1.4.1 WP1: Proof of concept and pilot definition

In a first step the pilot with regard to needed hardware and software components and physical setup (e.g. attachment of sensors to the cooling container, gateway use) has to be specified. Depending on the defined scenario, a prior test of the general wireless connectivity at the distribution centre is needed. As an example a sensor node that is put inside a cooling container, which is filled with CO_2, may have problems to contact the gateway. On the other side, the mobile phone needs to be able to communicate with the gateway and/or sensor nodes from the front of a truck to the trucks cargo area.

6.1.4.2 WP2: Software development

This work package contains all the efforts regarding software and hardware development. The respective software components, meaning a frontend application in the form of a smartphone app and a backend application are developed. The backend application gathers the sensor and alarm data generated by the sensor nodes contained in the cooling containers.

Figure 75 shows a possible GUI of the smartphone monitoring app.



Pallet # 0
Content: Oranges
Quantity: 5
State: Fine

Pallet # 1
Content: Oranges
Quantity: 5
State: Fine

Pallet # 2
Content: Oranges
Quantity: 5
State: Fine

This is the overview about your transport items and their present condition. Please take care for alarm messages.

Figure 75: Smartphone Monitoring App

6.1.4.3 WP3: Installation of hardware and software components

In this step the developed software and hardware components are installed in the cooling container and a chosen truck. Here, specific locations need to be found to put e.g. the gateway component in a reasonable place. The smartphone app will be distributed to drivers who will participate in the pilot.

6.1.4.4 WP4: Test and documentation

The final step consists of a test of the installed software and hardware. Finally documentation will be generated and a discussion of potential next steps will follow.

6.1.5 Conclusion

The pilot installation at the DC of Groupe Casino is a great opportunity for the consortium. It shows the relevance of the IoT-A technologies for external stakeholders, as the investment for Groupe Casino is not insignificant: Groupe Casino will provide a load carrier and potentially other devices / products at the distribution centre as well as personnel and access to the centre for partners from the IoT-A consortium. Likewise, we also see this pilot installation as a first step towards exploring exploitation venues for future products based on the IoT ARM, such as the planned M2M platform of SAP. Correspondingly, FhG's consulting expertise in logistics and IoT technologies might also lead to a further cooperation following the pilot installation at Groupe Casino.

6.2 Reverse Mapping of IoT ARM to MUNICH demonstrator

Besides the Groupe Casino pilot, work package 7 was able to include a stakeholder proposed scene into the use cases. Beginning with the initial demonstration of prototypes at IoT week 2012 in Venice, valuable stakeholder feedback out of the Stakeholder Workshop 4 (SW4) on the health use case was considered. Professor Christoph Thümmler, who is actively contributing in the eHealth area, proposed several real-life scenarios to strengthen the defined health storyline of D7.2 [Fiedler 2012] on which the IoT week 2012 demos were based. Several options of matching scenario were discussed. Finally, we chose a new scene that fits to the proposed use case "Tracking my Things" of D7.1 [Hagedorn 2011] from Telefónica (TID) who left the IoT-A project early and therefore did not implement a demonstrator. A project plan was defined to have the implementation of the new demonstrator ready by IoT week 2013 in Helsinki. The planning and implementation process was supported with a reverse-mapping of the IoT ARM to the scenario, which was covered in parts from work package 1.

Storyline-wise the new scene "Patient safety in the operating theatre" fits as a new element after the defined accident and hospitalization scenes. As we already included the EHR in previous scenes the general scenario of tracking medical items, which may get contaminated and get in contact with a patient in the operational theatre, was seen as an improvement. Besides that, the tracking of towels used in abdominal surgery proves a real business case, which is further explained in Section 5.3. Figure 76 shows the planning meeting of work package 7 with Christoph Thümmler in Munich.

Figure 76: MUNICH demonstrator

The demonstrator was shown at IoT week 2013 in Helsinki, Finland. The specific application was implemented with help of the MUNICH platform by Celestor, Napier University Edinburgh, Technical University of Munich and Siemens.

In section 5.6.4 of D1.5 [Carrez 2013] the complete reverse mapping process of the MUNICH demonstrator to the IoT ARM is explained. The detailed description of the demo implementation can be found in Section 3.2.7.

7 Conclusion and outlook

In this deliverable we have presented the final implementations of the two central use cases of the IoT-A project. For both the health and retail use case we have discussed in depth the architecture and implementation as it has evolved throughout the project with a special focus on the demonstrated concepts from the IoT ARM and the technical work packages.

For each of the scenes we have "eaten our own food" and used the IoT ARM as a tool for designing, modeling, and presenting the respective scenes including central IoT ARM parts such as the IoT Domain Model, the Information Model, applicable design choices as well as exhaustive information about the feedback we gathered when presenting the scenes at various occasions. This lavish discussion of the implementation of the use cases clearly demonstrates the utility of the IoT ARM for building IoT use cases, does all of the discussed scenes where not only successfully implemented and demonstrated, but the unified and standardized description formats of the scenes facilitate an understanding of the core issues of the use cases significantly.

The approach of validation by implementation and demonstration, as presented and discussed in chapter 3, is certainly valuable. We did, however, not stop there, but include both a relation to the requirements process as well as a business analysis of the use cases. The core rationale behind the requirements analysis is to judge the relevance of the implemented scenes for the requirements of our stakeholders and technical work packages. Naturally, as we explicitly built our scenes upon technical components of the IoT-A work packages and the IoT ARM itself, we anticipated a high correlation between the requirements for the IoT ARM and the technical results on the one hand, and the use case is on the other hand. As we have discussed in chapter 4, we managed to reach an overlap of requirements to our scenes of almost 90%. From a requirements perspective, we can thus clearly claim that our use cases are relevant for our stakeholders and provide a high correlation with the technical and conceptual results of the IoT-A work packages.

As most of the IoT-A stakeholders and consortium members come from a technical background, it is crucial to not only validate technical aspects of the use cases, but also evaluate the business side, in order to prove the relevance of IoT-A for commercial exploitation. Consequently, we have provided a business analysis of the use cases that regards each individual use case and performs both a cost-benefit analysis and a respective sensitivity analysis including calculations for different predictions of technical and economic developments in the future. For both of the use case we could demonstrate that solutions based on the IoT ARM are potentially more profitable than applications that are not based on our sophisticated and detailed IoT-A approach. It must also be noted that we anticipate several secondary benefits that are not directly measured in profitability, but issues like the reputation of an organization that utilizes standardized processes such as the ones depicted in the IoT ARM might also be beneficial on different levels and have an indirect impact on the business side.

From our perspective, the most valuable activities related to technical validation address the interest of and interaction with external stakeholders that wish to exploit and utilize the demonstrators discussed in this deliverable. For both of the domains we have managed to engage significantly with real stakeholders outside to the project consortium. In the health domain, we have successfully mapped the IoT ARM to the MUNICH system and have exhibited a joint demonstrator on several occasions. Naturally, a real life deployment in the health domain is not realistic due to medical regulations. However, for the retail part, it is not unrealistic to engage and prototype with partners outside the consortium. As the French retailer Groupe Casino has expressed strong interest in our retail solutions, we have already planned a pilot installation in one of their distribution centers for the remaining period of the project and potentially even afterwards. As for instance SAP is commercializing an M2M platform, we have a strong interest of potentially continuing the interaction with Groupe Casino even after the project ends. But for now we have provided in the document our detailed planning for the trial execution at the Groupe Casino distribution center and focus our efforts on a successful pilot installation of IoT-A results.

References

[Bruegger 2009]	Bruegger, R., „Der IT Business Case – Kosten erfassen, Nutzen erkennen, Wirtschaftlichkeit nachweisen und realisieren“, Springer Verlag, 2009.
[BSI 2011]	BSI, “Privacy Impact Assessment Guideline for RFID Applications”, 2011
[Carrez 2013]	Carrez, F. (ed.) et. al. (2013), *Project Deliverable D1.5 - Final architectural reference model for the IoT v3.0*
[Fiedler 2012]	Fiedler, M., Bui, N., De Loof, J., Ho, E., Magerkurth, C., Mättig, B., Romero, G.M., Savry, O., Serbanati, A., Zeybek, E. (2012). *Project Deliverable D7.2 - Exact definition use case 1 and use case 2.*
[Hagedorn 2011]	Hagedorn, P., Magerkurth, C., Meyer, S., Haller, S., Sperner, K., Ho, E., et al. (2011). *Project Deliverable D7.1 - Initial definition of Use Cases 1/2.*
[Kranzfelder et al. 2001]	Kranzfelder, M.; Zywitza, D.; Jell, T.; Schneider, A.; Gillen, S.; Friess, H. Feussner, H. (2001): Real-Time Monitoring for Detection of Retained Surgical Sponages and Team Motion in the Surgical Operation Room Using RFID Technology: A Preclincial Evaluation. In: Journal of Surgucal Research, S. 1-8.
[MUWS 2013]	Anonymous, Anonymous, and Anonymous. "MUNICH Platform Workshop." Personal interview. 28 Jan. 2013
[Presser 2012]	Presser, M.: *IoT Comic Book Special Edition*, 2012, In: http://www.e-pages.dk/alexandra/14/
[Salinas Segura 2013]	Salinas Segura, A. (ed.) et. al. (2013), *Project Deliverable D6.4 - Final validation report*
[Zocher 2013]	Zocher, W.: RFID in OP. In: https://www.youtube.com/watch?v=8PK-jpds2qk&feature=youtu.be zugegriffen am 25.05.2013

A Appendix A: Requirements

A.1 121 Requirements Implemented, Sorted by Applicability

UNI ID	Requirement Type	Category	Description	Rationale	Fit Criterion	Total Scenes Applicable (out of 11 Scenes)
UNI.022	Functional Requirements	Security, Usage, Access Control	A system built using the ARM shall provide end users with secure access to resources	Patients are able to initiate communication to the providers Electronic Medical Record (EMR) or health database application using the secure messaging tool for a variety of purposes. Examples include providing manually gathered information on existing self-monitoring and/or chronic care regiments.	Access to resources and system components is secure, e.g. through access control or encryption	11

UNI.071	Design constraints	Data handling & communication, Semantics, Interoperability	A system built using the ARM shall provide standardized and semantic communication between services	"Standard communications between objects, from a communication channel point of view but also from a semantic point of view. (Standardization of object semantic is somehow similar to the standardization of MIB (Management Information Base) of telecommunication equipments)."	Services descriptions and service interfaces shall adhere to standards	11
UNI.093	Non-functional Requirements	Interoperability, Extensibility	A system built using the ARM shall be extensible for future technologies.	"The reference architecture shall provide an integral approach that combines legacy aspects as well as an imaginating vision on the Internet of Things."	The system makes little or no assumptions on protocols, interfaces, and communication styles of its components, although the use of widespread standards should be encouraged.	11

UNI.240	Functional Requirement	Interoperability, Self-Description	A system built using the ARM shall provide unified interfaces to access and query the resource/entity meta data	This will enable WP4 discovery and identification and also reasoning mechanisms to access the required descriptions	A unique IoT service and Virtual Entity dicovery and resolution mechanism is implemented	11
UNI.002	Non-functional Requirements	Privacy, Usage	Users have control how their data is exposed to other users	"Citizens want to protect their private data"	The system lets users select which personal data is accessible to other users	10
UNI.003	Non-functional Requirements	Self-description, Semantics	A system built using the ARM shall enable the provision and exchange of semantics between services in order to support the design of new applications	"I would like a way to create and exchange semantics between objects in order to design new applications"	Semantic descriptions of services in the system are available	10

UNI.004	Non-functional Requirements	Self-description, Semantics	A system built using the ARM shall enable the semantic description of physical entities	"I would like a way to create and exchange semantics between objects in order to design new applications"	Semantic descriptions of physical entities registered in the system are available	10
UNI.008	Non-functional Requirements	Interoperability	A system built using the ARM shall be able to run applications and services in an interoperable manner	"The problem is to provide a framework, a set of scenarios where these applications could be developed in harmony, in an interoperable way and in a way that responses to the real needs of organization and people"	The system should consistently use standardized or at least known interfaces	10
UNI.023	Non-functional Requirements	Interoperability, Enterprise Integration	A system built using the ARM shall provide access to external information sources, e.g. health databases	"Patients are able to initiate communication to the providers Electronic Medical Record (EMR) or health database application using the secure messaging tool for a variety of purposes. Examples include providing manually gathered information on existing self-monitoring and/or chronic care regiments."	The system can use external information sources	10

UNI.030	Functional Requirements	Discovery & lookup, Naming, Addressing	A system built using the ARM shall provide a resolution infrastructure for naming, addressing and assignment of virtual entities and services	"A system may be provided which is operable to determine a routing node for a data object. The system can comprise an identifier generator operable to generate an identifier for the data object on the basis of data content thereof, and a lookup engine operable to compare the identifier for the data object to a routing table to determine a routing node for the data element."	The system includes components that support such resolution	10
UNI.036	Functional Requirements	Self-description, Usage, Semantics	A system built using the ARM shall enable the retrieval of the self-description of things	"My wish is to retrieve the capacity of a thing. Thus, I can plan a change maintenance of all my bulbs if they can say when they should be changed"	It is possible to retrieve a description of a Virtual Entity	10

UNI.041	Functional Requirements	Data handling & communication	A system built using the ARM shall provide historical information about the physical entity	"A method for clarification whether the Cold/Hot Chain has been violated or not is required. To be able to do this, the continuous context information (e.g., temperature) of the things needs to be collected. This is for example of major importance to avoid any damage to the pharmaceutics during the transport and storage process."	There exists a system component that allows for retrieval of historical information	10
UNI.047	Non-functional Requirements	Interoperability	The system must ensure interoperability between objects or between applications	"As an example, CCTV system could inform traffic management of the length of the waiting queue at a crossroad. Having smart traffic lights receiving such input from the CCTV system could, could help changing the schedule of green/red light to optimize the traffic."	It is possible to exchange information between any two service or application (VE, IoT Service, application), provided they are granted access to each other	10

UNI.048	Functional Requirements	Interoperability, Discovery & lookup, Naming, Addressing	A system built using the ARM shall provide interoperable naming and addressing	"IoT-A will play a role in terms of providing a kind of novel resolution infrastructure. We need to understand how best IoT could be served by scheme regarding the naming of objects, the addressing and assigning problems."	Naming and addressing is made interoperable (e.g. by provisioning of dedicated component, adherence to standards, etc).	10
UNI.062	Design constraints	Security, Trust, Data handling & communication Availability, Integrity	A system built using the ARM shall provide trusted and secure communication and information management	"A method for clarification whether the Cold/Hot Chain has been violated or not is required. To be able to do this, the detailed context information (e.g., temperature) of the things, which have been collected in some database need to be easily made available. This is for example of major importance to avoid any damage to the pharmaceutics during the transport and storage process."	Information is available and securely communicated (e.g. by means of encryption, integrity enforcement, access control, etc)	10

UNI.092	Non-functional Requirements	Usage	Remote services shall be accessible by human users	"The mobile phone of the consumer can and should be used for interacting with product centric services"	It is possible for a end-user to access remote VE/IoT Services	10
UNI.095	Design constraints	Data handling & communication, Addressing	A system built using the ARM shall include an interface to IP communication protocols.	"The reference architecture shall consider that we have gateways to IP everywhere, so we must have a global addressing system with protocol and so on. That would be an evolution of IPv6. Or we need an integration package for existing addressing systems."	The system should at least provide a gateway to IP, or use consistently IP	10
UNI.312	Functional Requirement	Data handling & communication, Interoperability	A system built using the ARM shall use/support common-addressing-schemes such as IPv6.	The usage of IPv6 is common practice in IoT systems.Reference: N. Kushalnagar, G. Montenegro, C. Schumacher, IPv6 Over Low-Power Wireless Personal Area Networks (6LoWPANs): Overview, Assumptions, Problem Statement, and Goals, IETF RFC 4919, August 2007.	This function has to be supported by all the component involved in the communication including intermediate network equipment (e.g. gateways) and/or devices.	10

UNI.313	Functional Requirement	Data handling & communication, Interoperability	A system built using the ARM shall support mapping from different addressing schemes to IP v6 (Compatible-addressing-scheme support).	Communication with non-IPv6 networks must be possible (IPv4, others).	This function has to be supported by all the component involved in the communication including intermediate network equipment (e.g. gateways) and/or devices.	10

A.2 39 Requirements Marked as Important in the Use Cases but not Implemented

UNI ID	Requirement Type	Category	Description	Rationale	Fit Criterion
UNI.021	Functional Requirements	Radio-awareness, Usage, Energy-awareness	The user shall be able to control the radio activity of the system	"The application can control the radio transmission"	The user of the system is capable of controlling the radio usage of the system components e.g. through configuring an error detection & correction component
UNI.060	Non-functional Requirements	QoS, Usage	The system shall support different SLA	"Communication blackouts are not accepted from client side and particularly if they are paying for premium services"	It is possible to express and enforce SLA for components or services of the system

UNI.073	Functional Requirements	Semantics, Usage	A system built using the ARM shall allow the semantic description of physical entities and services by a user	"I would like a way to create and exchange semantics between objects in order to design new applications"	Semantic descriptions of services and associations to virtual entities in the system can be added by the user
UNI.089	Non-functional Requirements	N/A	A system built using the ARM shall support reliable time synchronization	"Services which depend on a precise time need a guarantee that the devices they are communicating to have the right time."	The system includes mechanisms to provide time synchronization
UNI.099	Non-functional Requirements	N/A	A system built using the ARM shall guarantee correctness of resolutions.	"When searching for a certain object you need an implemented system that actually gives you the correct result."	Under proper conditions, the resolution functionality can guarantee correctness of their results
UNI.232	Functional Requirement	Interoperability	The process-execution engine shall support the integration with a complex-event-processing (CEP) component.	One WP central process execution engine including the CEP enables a bigger research contribution.	One process execution engine is used in task 2.2 as well as in task 2.4

UNI.236	Functional Requirement	QoS, Data Handling and communication, Usage	A system built using the ARM shall offer services for the retrieval of quality of information related to virtual entities.	Different devices provide information with varying quality. An application may have certain quality requirements.	Quality of information related to virtual entities can be retrieved from the system by aid of a service
UNI.237	Functional Requirement	QoS, Data Handling and communication	A system built using the ARM shall offer data types for describing the quality of information related to virtual entities.	Different devices provide information with varying quality. An application may have certain quality requirements.	Quality of information related to virtual entities can be described in the system
UNI.247	Functional Requirement	Service composition & programmability	The service organization shall support flexible composition	Services involved in compositions can fail and need to be replaced by some serving equal needs. Reference: Kephart, J. O., & Chess, D. M. (2003). The vision of autonomic computing. Computer, 36(1), 41-50.	A service orchestration component is implemented.
UNI.322	Functional Requirement	Data handling & communication, Privacy	A system built using the ARM shall support anonymous communication between devices (Anonymity support).	In some cases, IoT devices may need to communicate without disclosing its identity	This function has to be supported by all the component involved in the communication including intermediate network equipment (e.g. gateways) and/or devices.

UNI.404	Functional Requirement	Self-description, Geolocation	A system built using the ARM shall support a hybrid location model, that is, it shall support symbolic coordinates as well as local and global geometric coordinates	Derived from SP requirement "Smart products shall support a hybrid location model, that is, it shall support symbolic coordinates as well as local and global geometric coordinates" Reference: [SmartProduct Deliverable: "D6.3.1 & D6.4.1 & D6.5.1 Initial Smart Products Communication Middleware, Initial Sensor and Actuator Integration Framework & Initial Context and Environment Model Framework". http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D6.345.1_Final.pdf]	Feature is incorporated in the system

UNI.419	Functional Requirement	Discovery & lookup, Autonomicity	A system built using the ARM shall be able to track dynamic associations between a virtual entity and services related to the virtual entity. This needs to be done in order to determine whether they are still valid.	Due to the mobility of things, as well as devices whose resources are accessible through services, changing services may provide information, allow actuation, or enable interaction with things. In order to provide the currently relevant services for a thing, dynamic associations must be tracked to determine whether they are still valid.	Associations are tracked and automatically removed if it is determined that the association is no longer valid
UNI.501	Non-functional Requirement	Security	A system built using the ARM shall make it difficult to spy on communicated messages.	The confidentiality of messages must be ensured.	Secure channels between the devices can be realized. Is the eavesdropping on the communication easy?
UNI.502	Non-functional Requirement	Security	A system built using the ARM shall prevent a device (contactless card for example) from being activated without the consent of the owner.	The unsolicited scanning of people shall be avoided. A device is always owned by a person or an entity. For example, in a retail use case, the owner of an RFID tag can be a retailer and after the checkout the new owner should be the client. The aim is to avoid skimming attacks	A validation by the owner must be performed or only the owner must be able to read his own devices (tags for example). Is somebody able to read the content of a device without the consent of the owner of the device?

UNI.503	Functional Requirement	Usage, Security, Integrity, Non-Repudiation, Privacy	A system built using the ARM shall make it be possible to change the owner of a device (tag for example).	A device is always owned by a person or an entity. For example, in a retail use case, the owner of an RFID tag can be a retailer and after the checkout the new owner should be the client. The aim is to avoid skimming attacks. Privacy preserving solutions in RFID require sharing a secret key between tag and reader (or owner since in this case, the owner enters his key in the reader). It must be possible to change this key in tag and reader (and even in the databases where the data related to the device is stored) if the owner has changed.	Since only the owner of the device must be able to read the data it contains. Password or strong authentication must be used (requirement just below). It must be possible to change the password or cryptographic keys when the owner is changing. Only the new owner must be able to read the content of his devices or to give the authorization to read it.
UNI.504	Non-functional Requirement	Security, Privacy, Access Control	A system built using the ARM shall prevent tracking of the identifier of the device (ID of an RFID tag for example) by unauthorised entities.	The tracking of items and then people raise the problem of privacy. To preserve privacy, only the owner of the tag shall be able to read it. So, authorized persons are the owner and the persons who are authorized by the owner. The "unauthorized entities" are all the other people.	Only the owner of the device must be able to read the data it contains. Password or strong authentication must be used. Can everyone read the unique identifier of a device? If yes, there is a pb of privacy.

UNI.505	Functional Requirement	Energy-awareness	A system built using the ARM shall support connecting devices able to do energy harvesting.	Maintain operation in environments where power supply is not possible.	Include in the final solution devices able to perform energy harvesting
UNI.507	Non-functional Requirement	Security, Privacy, Data handling & communication	A system built using the ARM shall support data security & privacy at atomic level	Security in end-to-end communication does not address security issues pertaining to the device itself.	Security/Privacy should be guaranteed across the whole communication path between the devices and the user applications
UNI.509	Non-functional Requirement	Self-description	Each IoT device shall possess a universal ID, part of it read only and part of it read/write.	Enable object recognition and setup/configuration in the context of applications development	Enable visibility of objects in the context of user applications

UNI.510	Functional Requirement	Security	Atomic-level protocols must implement only functions related to data acquisition (e.g. DSP-level), crypto and security	Atomic-level protocols are the protocols realised to carry out a particular task related to device internal functions. E.g. how data are acquired from the environment. How they are encoded/encrypted for transportation of unreliable networks, etc. This requirement is needed to avoid overlap with user-level communication protocols.	Atomic-level protocols only implement functions related to data acquisition, crypto, and security.
UNI.602	Non-functional Requirement	Privacy, Trust	The "infrastructure services" of a system built using the ARM (i.e. resolution services, security services, management services) shall be trustable	The services provided by such "infrastructure services" should be trustworthy.	The system contains a trust and reputation component

UNI.603	Functional Requirement	Integrity	The "infrastructure services" of a system built using the ARM (i.e. resolution services, security services, management services) shall comply with the infrastructure service design and operate accordingly	Such "infrastructure services" should operate properly according to their design.	The system requires a vulnerability assessment to prevent the installation and execution of malicious code

UNI.605	Non-functional Requirement	Privacy, Trust	A system built using the ARM shall support the reversing of the pseudonymization processes in order to guarantee mutual accountability	Some scenarios require Subjects to take responsibility for their actions. Some Services could be classified or critical for their provider and could require Users to take responsibility of their action. On the other hand Users might need providers to take responsibility for the Services they provide, because relying on such Services is critical for them. The IoT should support the reversing of the Pseudonymization processes.	The system supports reversing of pseudonymization between user and service provider
UNI.606	Non-functional Requirement	Privacy, Non-repudiation	A system built using the ARM shall make the traceability of digital activities impossible	Subjects should not be able to track the digital activities of other subjects	The system supports non-traceability of subjects

UNI.609	Non-functional Requirement	Security, Integrity	A system built using the ARM shall ensure Data Freshness	The system should be protected from replay attacks (message replays at Service level, packet replay at network and link layer level).	The system supports anti-replay protection (e.g. Cryptographic nonce)
UNI.610	Non-functional Requirement	Security, Availability	A system built using the ARM shall provide IoT-Service availability	Services providing access to Resources must be reachable by the Users who might need to rely on them. This requirement has a specific IoT declination as the resources of many nodes will be constrained and specific ways to protect from DoS or exhaustion attacks will be needed.	The system requires a monitoring and planning for IoT services
UNI.614	Functional Requirement	Security, QoS	A system built using the ARM shall provide Quality of Service	In networks where nodes are constrained devices with limited communication capabilities, QoS might have a new (or extended) meaning compared to the current meaning. For example, real-time, event-triggered data with high time resolution, needs to be delivered with a higher priority than other and might need to ignore the need to sleep of some devices in the network.	The system supports QoS enabling service prioritization and communication enforcement

UNI.616	Non-functional Requirement	Security, Availability	A system built using the ARM shall ensure network availability	The network functions should be available to network endpoints. Appropriate measures should be taken to avoid network disruption.	The system requires a monitoring of IoT network
UNI.617	Non-functional Requirement	Security, Integrity	A system built using the ARM shall enforce correct routing	Packet routing over underlying Link Layer should be efficient and should not be subject to disruption by malicious subjects. Disruption could lead to worm/blackhole, exhaustion and DoS attacks.	The system supports correct and invulnerable routing
UNI.618	Non-functional Requirement	Security, Access control	A system built using the ARM shall have a communication control for restricted usage	In some cases hop by hop communication should only be available to authenticated devices.	The system supports restricted access control for communication
UNI.620	Non-functional Requirement	Security, Integrity	A system built using the ARM shall provide Software Integrity	The software execution environment should preserve software integrity.	The system provides a mechanism to define the software integrity level
UNI.623	Non-functional Requirement	Privacy, Geolocation	A system built using the ARM shall support location privacy	The Location of a Subject should only be available to authorized Subjects. Specific methods for obscuring both network and physical location should be available.	The system supports a Location-Privacy Protection Mechanism (LPPM) such as a Location-Privacy Meter

UNI.625	Functional Requirement	Security, Privacy, Security management	A system built using the ARM shall provide a device security and privacy measurement	Users should be able to monitor and control the security and privacy settings of all the devices that they own.	The system supports a device security and privacy measurement

UNI.701	Non-functional Requirement	Evolvability	A system built using the ARM shall accommodate fast developmental changes in applications and network	"New applications can change traffic characteristics in a few months. In the past decade several applications dramatically changed the way how the Internet is used. Nobody has actually foreseen the success of P2P networks, and especially Youtube and Facebook. Thus, the question is whether it is possible to design a Future Internet without having any ideas what the "next big things" could be. If thus the traffic changes are unpredictable, then we need to establish a fast and stable infrastructure without any assumptions on the traffic." Reference: *G. Drea Rodosek, A. Pras, H. Schulzrinne, and B. Stiller, "Learning from the Past: Implications for the Future Internet and its Management?", Dagstuhl Seminar 11042, 2011*	The system is able to accommodate fast developmental changes in applications and network. For example, management is able to deal with different systems.

UNI.703	Design constraint	Architecture	A system built using the ARM shall be based on a cross-layered architecture	"Full decoupling of planes (management, user, control) is good in an "old-style telco world", however, it will not work in the Future Internet." Reference: *G. Drea Rodosek, A. Pras, H. Schulzrinne, and B. Stiller, "Learning from the Past: Implications for the Future Internet and its Management?", Dagstuhl Seminar 11042, 2011*	Functional Components from different Functionality Groups interact with each other rather than working in a decoupled manner.
UNI.709	Functional Requirement	Architecture, Usability	A system built using the ARM shall provide a single, simple management interface for all communication protocols	"The operational complexity of protocols should be confined to their implementation, and they should express the information required for managing them through a simple management interface. This includes the responsibility on the protocol implementer for a detailed understanding of the protocol operation while reducing the burden on management applications." Reference: *S. Kim, M. Choi, H. Ju, M. Ejiri, J. Hong, "Towards management requirements of Future Internet", In: "Challenges for next generation network operations and service management.", Springer, 2008, pp 156–166*	The system provides a single, simple management interface for all communication protocols.

UNI.710	Functional Requirement	Architecture	A system built using the ARM shall include a management repository to store information on the state of the system	"Management of the Future Internet architecture will require data on the current state of the network, available in real time. The challenge is that the proposed instrumentation systems can potentially gather vast quantities of high-dimensional data. This implies the requirement of a repository unit that will organize the measurement data." Reference: *S. Kim, M. Choi, H. Ju, M. Ejiri, J. Hong, "Towards management requirements of Future Internet", In: "Challenges for next generation network operations and service management.", Springer, 2008, pp 156–166*	The system includes a management repository to store information on the state of the system
UNI.713	Non-functional Requirement	Performance	Management functionalities shall react to dynamic operation and system changes in real time	Management system shall react to dynamic operation [and system] changes in real time. Reference: *Q. Wang, R. Jäntti, Y. Ali, "On Network Management for the Internet of Things", ", 8th Swedish National Computer Networking Workshop (SNCNW) http://users.tkk.fi/wangq1/SNCNW_OnNetworkManagement.pdf, 2012*	The Management functionalities react to dynamic operation and system changes in real time

UNI.717	Functional Requirement	Autonomicity	A system built using the ARM shall be able to perform self-optimisation	"The system can measure its current performance and it is able to compare it against to the known optimum level of performance. The system will adjust its operation to reach closer the optimal performance. The system is also able to change its operation to cope with new user set policies." Reference : *T. Töyry, "Self-management in Internet of Things", https://wiki.aalto.fi/download/attachments/59704179/toyry-self-management.pdf?version=1&modificationDate=1324369262000*	The system is able to perform self-optimisation, i.e. it "can measure its current performance and it is able to compare it against the known optimum level of performance. The system [then] adjust its operation to reach closer the optimal performance. The system is also able to change its operation to cope with new user set policies." *https://wiki.aalto.fi/download/attachments/59704179/toyry-self-management.pdf?version=1&modificationDate=1324369262000*

A.3 24 Requirements Marked as Not Important in the Use Cases and Not Implemented

UNI ID	Requirement Type	Category	Description	Rationale	Fit Criterion
UNI.020	Functional Requirements	Radio-awareness	A system built using the ARM shall support real-time monitoring of radio usage of devices and gateways	"The application knows the current radio transmission activity of the M2M device"	Radio usage of the whole system is monitored
UNI.065	Non-functional Requirements	QoS, Reliability	A system built using the ARM shall provide reliable services	"In order to accommodate certain scenario, support of a certain degree of reliability might be necessary"	The system provides services with an agreed upon level of reliability
UNI.094	Non-functional Requirements	N/A	The Architectural Reference Model shall support any IoT business scenario.	"The reference architecture shall provide the building blocks in a creative way coming from a business perspective."	Any given IoT business scenario should be made possible by the ARM
UNI.234	Non-functional Requirement	Data Handling & communication	Events shall be processed on a set of distributed nodes	A distributed architecture provides more flexibility in the way events are processed, saves energy and allows minimal functionality if there is no network connectivity	Implementation of a distributed event-processing component
UNI.235	Functional Requirement	QoS, Data Handling and communication	Processing of events shall take quality of information (QoI) into account	In IoT the quality of information stemming from events is often questionable.	Implementation of a QoI aware event processing algorithm

UNI.244	Functional Requirement	Service composition & programmability	The orchestration engine shall interpret service descriptions	Service orchestration needs to be done based on IOPE information provided in service descriptions. Reference: Bell, Michael. 2008. Service-Oriented Modeling. Chichester: John Wiley & Sons.	A service composition component is implemented.
UNI.245	Functional Requirement	Service composition & programmability	The service organization shall support creation of new applications	Composite services allow added value services based on simple services	A service composition component is implemented.
UNI.251	Functional Requirement	Service composition & programmability, Usage	The service organization shall provide a feedback to the user who sent a composition request	The service user needs to be informed whether or not the composition request has succeeded or failed due to uncertainty of service availability. Reference: Nielsen, J. (1993). Usability Engineering. Retrieved from http://dl.acm.org/citation.cfm?id=529793	A service orchestration component is implemented.
UNI.252	Non-functional Requirement	Usage	The service organization shall provide feedback within a reasonable amount of time.	A time out must be set for request/response loops. For requests entered by human users a limit of 10 seconds could be reasonable. After that an error is assumed. Reference: Nielsen, J. (1993). Usability Engineering. Retrieved from http://dl.acm.org/citation.cfm?id=529793	A service orchestration component is implemented.
UNI.253	Functional Requirement	Service composition & programmability, Usage	The orchestration engines shall support setting preferences for	Users can have the possibility to prefer one service over another for any reason	A service orchestration component is implemented.

			selecting services involved in composition		
UNI.405	Functional Requirement	Service composition and programmability, Extensibility, Geolocation	A system built using the ARM shall allow programmers to add new coordinate reference systems and shall support the transformation of coordinates among them	Derived from SP requirement: The location model shall allow programmers to add new coordinate reference systems and shall support the transformation of coordinates among them [SmartProduct Deliverable: "D6.3.1 & D6.4.1 & D6.5.1 Initial Smart Products Communication Middleware, Initial Sensor and Actuator Integration Framework & Initial Context and Environment Model Framework". http://www.smartproducts-project.eu/media/stories/smartproducts/publications/SmartProducts_D6.345.1_Final.pdf]	Feature is incorporated in the system
UNI.409	Functional Requirement	Data handling & communication	A system built using the ARM shall allow storage of VE changes, including structural changes (e.g. changes in the aggregation of multiple VEs constituting one overarching VE).	This is a main functionality of the BRIDGE system which applies to RFID/assets tracked in the EPCGlobal framework Reference: [BRIDGE deliverable: "High level design for Discovery Services". http://www.bridge-project.eu/data/File/BRIDGE%20WP02%20High%20level%20design%20Discovery%20Services.pdf]	Feature is incorporated in the system

UNI.420	Functional Requirement	Discovery & lookup, Autonomicity, Geolocation	A system built using the ARM shall be able to discover dynamic associations based on geographic location and other context information.	Mobility is one of the key reasons for changing associations. By monitoring both the location of physical entities and the service area of resources, dynamic associations can be discovered. Based on the proximity of the physical entity, the service area of the resource and the functionality provided by the resource, it can be determined whether the resource can provide any information about the physical entity or enable any actuation on the physical entity. If this is the case, an association between the virtual entity, which represents the physical entity in the system, and the service, which makes the functionality of the resource accessible, can be established.	By using location services new dynamic associations can be found
UNI.421	Functional Requirement	Discovery & lookup, Autonomicity, Geolocation	A system built using the ARM shall be able to track dynamic associations between a virtual entity and services based on geographic location to determine whether they are still valid.	Mobility is one of the key aspects for changing associations. By monitoring the location of physical entities, e.g., using location services, it can be determined when associations become invalid due to the geographic distance of physical entities and the service areas of resources and possibly other and possibly other aspects.	By using location services, it can be determined when dynamic associations become invalid.
UNI.512	Non-functional Requirement	Energy-awareness, Autonomicity	An application shall share information about resource usage (for instance, when will the application need to transmit a	IoT systems are often resource constrained, especially in terms of energy consumption. Optimum energy efficiency can only be achieved by cross-functional-layer optimisation, which is dependent on application needs.	Application-layer needs are made available to other functional layer (well) before the fact.

			message) with other functional layers.		
UNI.601	Non-functional Requirement	Security, Availability, Resilience	A system built using the ARM shall guarantee infrastructure availability	The services provided by the infrastructure should always be available, as their operation is critical to the operation of the Internet of Things. Users should thus be able to reach the infrastructure. The infrastructure services should be able to operate.	The system requires a monitoring and planning for all critical components
UNI.613	Functional Requirement	Security, Trust	A system built using the ARM shall be able to meter service reputation	As there is a high chance of nodes being compromised due to their physical availability to malicious users, a secondary mechanism for establishing trust is needed.	The system supports a service reputation metering (e.g. service trust level)
UNI.615	Non-functional Requirement	QoS	A system built using the ARM shall provide transport layer fairness	While congestion avoidance is important in any large network, in low bandwidth mesh networks this is essential.	
UNI.619	Non-functional Requirement	Security, Non-repudiation	A system built using the ARM shall ensure non repudiation at network level	Mobile devices should be able to join peripheral networks belonging to different provider. Devices entitled to join a given network must be able to do so.	The system supports non-repudiation so that allowed devices can join a specific network

UNI.702	Non-functional Requirement	Evolvability, autonomicity	A system built using the ARM development shall support iterative approaches (e.g. spiral model)	"The waterfall model does not work in practice in communications, for sure, software is not a “one-time instance”, changes will occur for some time. Thus, versions are needed, and for protocols we may arrive at the same iterative refinement approach." Reference: *G. Drea Rodosek, A. Pras, H. Schulzrinne, and B. Stiller, "Learning from the Past: Implications for the Future Internet and its Management?", Dagstuhl Seminar 11042, 2011*	The system development supports iterative approaches (e.g. spiral model).
UNI.708	Functional Requirement	Autonomicity	The system management shall auto-bootstrap	"The management plane should be operationally independent of the data plane and should be able to bootstrap without any pre-configuration." Reference: *S. Kim, M. Choi, H. Ju, M. Ejiri, J. Hong, "Towards management requirements of Future Internet", In: "Challenges for next generation network operations and service management.", Springer, 2008, pp 156–166*	The system management is able to auto-bootstrap. For example, the system management relies on purely internal knowledge to startup (script-based boot-strapping), or uses discovery mechanisms to gather required knowledge, or a mix of both approaches. The system management should be able to bootstrap without other components of the system to be present/up and running.
UNI.715	Functional Requirement	Autonomicity	A system built using the ARM shall perform data collection on its current state	In order to meet device and system constraints, the collection of system states is important. Reference: *Q. Wang, R. Jäntti, Y. Ali, "On Network Management for the Internet of Things", ", 8th Swedish National Computer Networking Workshop (SNCNW) http://users.tkk.fi/wangq1/SNCNW_OnNetworkManagement.pdf, 2012*	The system performs data collection on its current state

UNI.719	Non-functional Requirement	Autonomicity, Resilience, Availability	A system built using the ARM shall be able to perform self-protection	"The system defends itself against internal and external threats, which can be accidental, such as cascading failures, or malicious attacks against the system. To manage the threats the system must be aware of its environment and have means to react to detected threats." Reference : *T. Töyry, "Self-management in Internet of Things", https://wiki.aalto.fi/download/attachments/59704179/toyry-self-management.pdf?version=1&modificationDate=1324369262000*	The system is able to perform self-protection, i.e. it is capable of taking action to prevent faults and attacks.
UNI.721	Non-functional Requirement	Architecture	The system management shall be operationally independent of the specifics of the communication functionality.	"The management plane should be operationally independent of the data plane and should be able to bootstrap without any pre-configuration." Reference: *S. Kim, M. Choi, H. Ju, M. Ejiri, J. Hong, "Towards management requirements of Future Internet", In: "Challenges for next generation network operations and service management.", Springer, 2008, pp 156–166*	Management does not influence the system's messaging/communication aspects