SEVENTH FRAMEWORK PROGRAMME

Specific Targeted Research Project

Call Identifier:	FP7–ICT–2011–7
Project Number:	287305
Project Acronym:	OpenIoT
Project Title:	Open source blueprint for large scale self-organizing cloud environments for IoT applications

D6.3.1 Proof-of-Concept Validating Applications a

Document Id:	OpenIoT-D631-131226-Draft
File Name:	OpenIoT-D631-131226-Draft.pdf
Document reference:	Deliverable 6.3.1
Version:	Draft
Editor(s):	Arkady Zaslavsky
Organisation:	CSIRO
Date:	2013 / 12 / 27
Document type:	Deliverable (Report, Prototype)
Security:	PU (Public)

DOCUMENT HISTORY

Rev.	Author(s)	Organisation(s)	Date	Comments
V01	Arkady Zaslavsky	CSIRO	2013/11/04	ToC & contributors
V02	Mehdi Riahi	EPFL	2013/11/29	Utility Metrics Specifications
V03	Nikos Zarokostas, Panos. Dimitropoulos	SENSAP	2013/11/29	Input in Section2, Description of SENSAP's Manufacturing Use case
V04	Prem Jayaraman and Ali Salehi	CSIRO	2013/12/10	Digital agriculture Phenonet (Section 5)
V05	Arkady Zaslavsky	CSIRO	2013/12/10	Editing sections
V05	Reinhard Herzog	IOSB	2013/12/12	FH IOSB scenario
V06	Nikos Zarokostas, Panos Dimitropoulos	SENSAP	2013/12/14	Updates to SENSAP's Manufacturing Use case
	Arkady Zaslavsky	CSIRO	2013/12/15	Editing/adding sections/content
	Johan E. Bengtsson, Keith Spiteri	AL	2013/12/19	Completed section 3.2 Silver Angel
V07	Nikos Zarokostas	SENSAP	2013/12/23	Technical Review
V08	Dinko Oletic	UniZ-FER	2013/12/23	Quality review
V09	Arkady Zaslavsky	CSIRO	2013/12/24	Final editing
V10	Martin Serrano	DERI	2013/12/24	Circulated for Approval
V11	Martin Serrano	DERI	2013/12/27	Approved
Draft	Martin Serrano	DERI	2013/12/27	EC Submitted

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

TERMS AND ACRONYMS

Term	Meaning
6LoWPAN	IPv6 over Low power Wireless Personal Area Networks
AAL	Ambient Assisted Living
ARM	Architecture Reference Model
BPM	Business Process Language
BPMN	Business Process Modelling Notation
BPWME	Business Process Workflow Management Editor
CoAP	Constrained Application Protocol
CPI	CSIRO Plant Industry
CRUD	CReate, Updated, Delete
DOLCE	Descriptive Ontology for Linguistic and Cognitive Engineering
DoW	Description-of-Work
DSO	Decision Support Ontology
EPC	Electronic Product Code
EPC-ALE	Electronic Product Code Application Level Events
EPC-IS	Electronic Product Code Information Sharing
ERP	Enterprise Resource Planning
GPL	General Public Licence
GPS	Global Positioning System
GSN	Global Sensor Networks
GTIN	Global Trade Item Number
HTML	HyperText Markup Language
HTTP	Hypertext Transfer Protocol
JSF	Java Server Faces
ICO	Internet-Connected Objects
ICT	Information and Communication Technologies
IEEE	Institute of Electrical and Electronics Engineers
IETF	Internet Engineering Task Force
IERC	Research Cluster for the Internet of Things

IoT	Internet of Things
LGPL	Lesser General Public License
MRP	Manufacturing Resource Planning
OGC	Open Geospatial Consortium
OMG	Object Management Group
ONS	Object Naming Service
OSS	Open Source Software
PDA	Personal Digital Assistant
PET	Privacy Enhancing Technologies
QoS	Quality of Service
QR-Code	Quick Response Code
RDF	Resource Description Format
REST	Representational State Transfer
RFID	Radio Frequency Identification
SGTIN	Serialized Global Identification Number
SLA	Service Level Agreement
SME	Small Medium Enterprise
SOA	Service Oriented Architecture
SOS	Sensor Observation Service
SPS	Sensor Planning Service
SSN	Semantic Sensor Networks
UML	Unified Modelling Language
WSN	Wireless Sensor Networks
XML	eXtensible Markup Language

GLOSSARY AND TERMINOLOGY

Term	Meaning
(OpenIoT) Architecture	The set of software and middleware components of the OpenIoT platform, along with the main structuring principles and inter-relationships driving their integration in an IoT/cloud platform.
(OpenIoT) Middleware	System level software (compliant to the OpenIoT architecture), which facilitates the integration of on-demand cloud-based IoT services.
(OpenIoT) Platform	A set of middleware libraries and tools, which enable the development and deployment of (OpenIoT compliant) cloud-based IoT services.
(OpenIoT) Use Case	A domain-specific application serving needs of end users, which is implemented based on the OpenIoT platform.
(OpenIoT) Service	An IoT service deployed over the OpenIoT platform.
(OpenIoT) Scenario	A specific set of interactions between OpenIoT components serving the needs of an application
(OpenIoT) Cloud	A set of computing resources enabling the delivery of IoT services over the network and based on the use of OpenIoT platform.
Global Scheduler	A software component that regulates how IoT services access the different resources managed by the OpenIoT platform.
Local Scheduler	A software component that regulates how IoT services access the local resources managed by an instance of the sensor middleware (and more specifically the GSN middleware).
Utility Metrics	A set of quantities that are used for the metering of IoT services.
(OpenIoT) Service Delivery	The process of deploying and offering an OpenIoT service, after selecting the resources that are involved in the service

(OpenIoT) Request Presentation	The software component that visualises the outcomes of an OpenIoT service based on the use of appropriate mashups and mashup libraries.
Sensor Selection	The process of selecting sensors that can contribute information to a particular service.
Virtual Sensor	All the physical or virtual items (i.e. services, persons, sensors, GSN nodes) which provide their information through a GSN endpoint.
Sensor Discovery	The process of detecting physical and virtual sensors, as well as of the services offered by them.
Resource Discovery	The process of detecting an IoT resource (such as a sensor, a service or a database).
Utility Manager	A software component (part of the OpenIoT platform), which performs metering based on the tracking and combination of utility metrics.
Sensor Directory	A software service which stores, organizes and provides access to information about (physical and virtual) sensors.
(OpenIoT) Sensor Middleware	The part of the OpenIoT middleware platforms that facilitate access to, collection and filtering of OpenIoT data streams.
Global Sensor Networks (GSN)	An open source sensor middleware platform enabling the development and deployment of sensor services with almost zero-programming effort.
Data Streams	A stream of digital information stemming from a physical or virtual sensor.
Data Stream Engine	A software component enabling the processing of data streams, as well as the management of the process of publishing and subscribing to data stream.
Linked Sensor Data	Set of (Semantic Web) technologies for exposing, sharing, and connecting sensor data, information, and knowledge.

1 INTRODUCTION

1.1 Scope

This deliverable is a document describing Proof-of-Concept Validating Applications (a), Open source implementations of the OpenIoT proof-of-concept applications in e-Science (Digital Agriculture), Manufacturing/logistics (Intelligent Manufacturing) and Smart Cities (Campus Guide and Silver Angel). The applications will be released based on two iterative releases, in-line with the intentions and exploitation modalities listed in the description of work. It is the first of two releases planned for November, 2013 and the second release is planned for September 2014. Each release will be using the latest available version of the OpenIoT platform (released as part of successive versions of deliverable D4.3).

1.2 Audience

The audience of this document is primarily the European Commission, the OpenIoT consortium, user and developer communities, but also solution providers that would like to use OpenIoT for developing and deploying IoT services. This document also acts as a reference for OpenIoT end-users, software developers and anyone interested in the software developments of the OpenIoT project use cases for development of applications and services based on OpenIoT open source software platform as well as for dissemination purposes.

More specifically, the target audience for this deliverable includes:

- **OpenIoT project members**, notably members of the project that intend to engage in the deployment and/or use of the OpenIoT open source middleware framework. For these members the deliverable could serve as a valuable guide for the installation, deployment, integration and use of the various modules that comprise the OpenIoT software to develop use cases and implement scenarios that will be mapped onto OpenIoT software platform.
- **The IoT open source community,** which should view the present deliverable as the demonstration of OpenIoT platform middleware for integrating IoT applications, notably applications that adopt a cloud/utility-based model. Note also that members of the open source community might also be willing to contribute to the OpenIoT project. For these members, the deliverable can serve as a basis for understanding the technical implementation of the components that comprise the first release of the OpenIoT middleware as well as the diverse scenarios and use cases that demonstrate the OpenIoT software platform.
- **IoT researchers at large,** who could find in this deliverable a practical guide and hints on how to develop domain-specific applications with the OpenIoT software platform.

- **IERC projects and their members,** who could find in this deliverable the practical examples and diverse scenarios how OpenIoT software platform and IoT at large could be used in a variety of application domains and applications. As already outlined, the OpenIoT architecture is largely based on the IERC reference architecture.
- **Solution Provider** and specifically companies wishing to provide some IoT solution based on OpenIoT. These companies will benefit from reading concrete examples of OpenIoT use cases implementations as part of this document.

All the above groups could benefit from reading the report, but also from delving into details of the released prototype implementation as well as detailed specifications and details of implemented use cases.

1.3 Summary

This document reports on the first release of Proof-of-Concept Validating Applications (a), on partner experience and lessons while implementing diverse use cases on the basis of OpenIoT software platform. The document summarises the OpenIoT software platform, core middleware and components that are used to develop OpenIoT use cases and map the use cases scenarios onto OpenIoT software platform. The implemented use cases include: (a) Smart Cities – Campus Guide; (b) Smart Cities – Silver Angel; (c) Smart Industries – Intelligent manufacturing – Materials Flow and Manufacturing Performance Traceability; (d) Smart Industries – Digital Agriculture – Phenonet. The fifth use case – Smart Cities – Urban Crowdsourcing for Air Quality Monitoring is reported in a separate deliverable. All the reported use cases describe how the OpenIoT use cases are/can/will be mapped to OpenIoT platform services according to 3-stage priorities. They include description of test data sets, whether they're already available for public use, or what are the plans to release them for public use via the OpenIoT Github. The use case sections also describe experience, issues, difficulties, lessons while developing use cases on the basis of the OpenIoT software platform.

1.4 Structure

The deliverable is structured as follows: section 2 provides a high level overview of the OpenIoT software architecture, core middleware components and features; section 3 describes two Smart Cities use cases – Campus Guide and Silver Angel; section 4 describes the Intelligent Manufacturing use cases, section 5 describes the Digital Agriculture – Phenonet use case; and section 6 concludes the deliverable.

2 OPENIOT ARCHITECTURE AND PLATFORM

This section provides a brief overview of OpenIoT software architecture and platform, including core middleware components, features and modules which are essential for developing the OpenIoT use cases reported in the subsequent sections. More details about the OpenIoT platform are available in other deliverables of the project (D4.3, D4.4).

2.1 Overview of OpenIoT Architecture

The OpenIoT Architecture is comprised by seven main elements as depicted in Figure 1. The Sensor Middleware, the Cloud Data Storage, the Scheduler in conjunction with Discovery Services functionality, the Service Delivery and Utility Manager, the Request Definition, the Request Presentation and the Configuration and Monitoring. The main core components have been introduced first in previous project documents (i.e. D2.2[1], D2.3[2]) and described with more details in terms of service architecture functional blocks in D4.1[3]. In this section an overview of those components with accurate refinements in functionality is included.

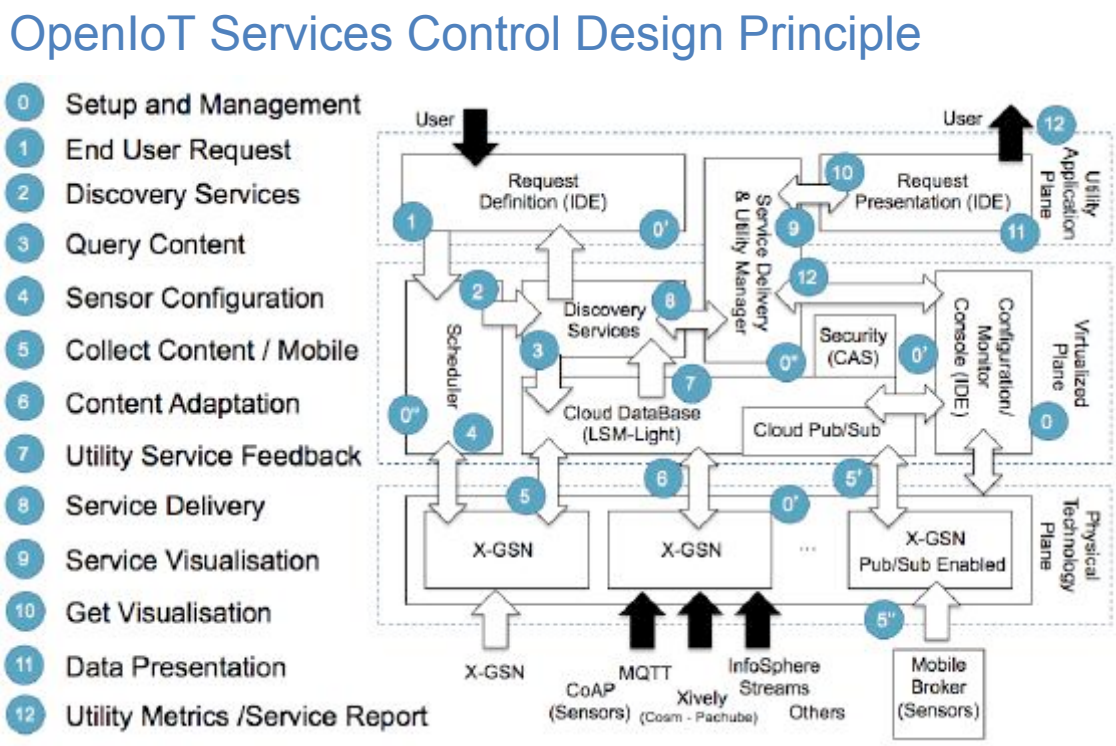


Figure 1: OpenIoT Main Core Components Functional Blocks.

[1] D4.1 Service Delivery Environment Formulation Strategies

[2] D2.2 OpenIoT Platform Requirements and Technical Specifications

[3] D2.3 OpenIoT Detailed Architecture and Proof-of-Concept Specifications

- The **Sensor Middleware** (Extended Global Sensor Network**,** X-GSN) collects filters and combines data streams from virtual sensors or physical devices. It acts as a hub between the OpenIoT platform and the physical world. The Sensor Middleware is deployed on the basis of one or more distributed instances (nodes), which may belong to different administrative entities. The prototype implementation of the OpenIoT platform uses the GSN[4] sensor middleware that has been extended and called X-GSN (Extended GSN).
- The **Cloud Data Storage** (Linked Stream Middleware Light, LSM-Light) enables the storage of data streams stemming from the sensor middleware thereby acting as a cloud database. The cloud infrastructure stores also the metadata required for the operation of the OpenIoT platforms (functional data). In addition to data streams and metadata, the cloud could also host computational (software) components of the OpenIoT platform (i.e. Schedules and SD&UM) in order to benefit from the elasticity, scalability and performance characteristics of the cloud. The prototype implementation of the OpenIoT platform uses the LSM Middleware, which has been re-designed with push-pull data functionality and cloud interfaces for enabling additional cloud-based streaming processing.
- The **Scheduler** processes all the requests for on-demand deployment of services and ensures their proper access to the resources (e.g. data streams) that they require. This component undertakes the following tasks: it discovers the sensors and the associated data streams that can contribute to service setup; it manages a service and selects/enables the resources involved in service provision.
- The **Service Delivery & Utility Manager** performs a dual role. On the one hand, it combines the data streams as indicated by service workflows within the OpenIoT system in order to deliver the requested service (with the help of the SPARQL query provided by the Scheduler). To this end, this component makes use of the service description and resources identified and reserved by the Scheduler component. On the other hand, this component acts as a service metering facility which keeps track of utility metrics for each individual service. This metering functionality will be accordingly used to drive functionalities such as accounting, billing and utility-driven resource optimization. Such functionalities are essential in the scope of a utility (pay-as-you-go) computing paradigm, such as the one promoted by OpenIoT.
- The **Request Definition** component enables on-the-fly specification of service requests to the OpenIoT platform. It comprises a set of services for specifying and formulating such requests, while also submitting them to the Global Scheduler. This component may be accompanied by a GUI (Graphical User Interface).
- The **Request Presentation** component, which is in charge of the visualization of the outputs of a service that is provided within the OpenIoT platform. This component selects mash-ups from an appropriate library in order to facilitate service presentation. It is expected that service integrators implementing/integrating solutions with the OpenIoT platform are likely to

[4] http://sourceforge.net/apps/trac/gsn/

enhance or even override the functionality of this component on the basis of a GUI pertaining to their solution.

- The **Configuration and Monitoring** component, which enables management and configuration of functionalities over the sensors and the (OpenIoT) services that are deployed within the OpenIoT platform. It is also supported by a GUI.

2.1.1 OpenIoT Architecture Proof of Concept

The OpenIoT project provides a proof-of-concept (PoC) implementation which presents a minimal set of components that demonstrates the basic workflows of OpenIoT architecture services. This implementation is used as the “skeleton” of the final platform and is available through the OpenIoT open source portal. The objectives of this implementation are the following:

- To provide the first integrated version of the OpenIoT software to the open source community.
- To involve the first users from outside the consortium, to get a feedback from the users/developers regarding the OpenIoT architecture.
- To bootstrap the OpenIoT open source community.

Figure 2 bellow illustrates the high level view of the functional blocks of the main components of the OpenIoT Architecture, which were listed above.

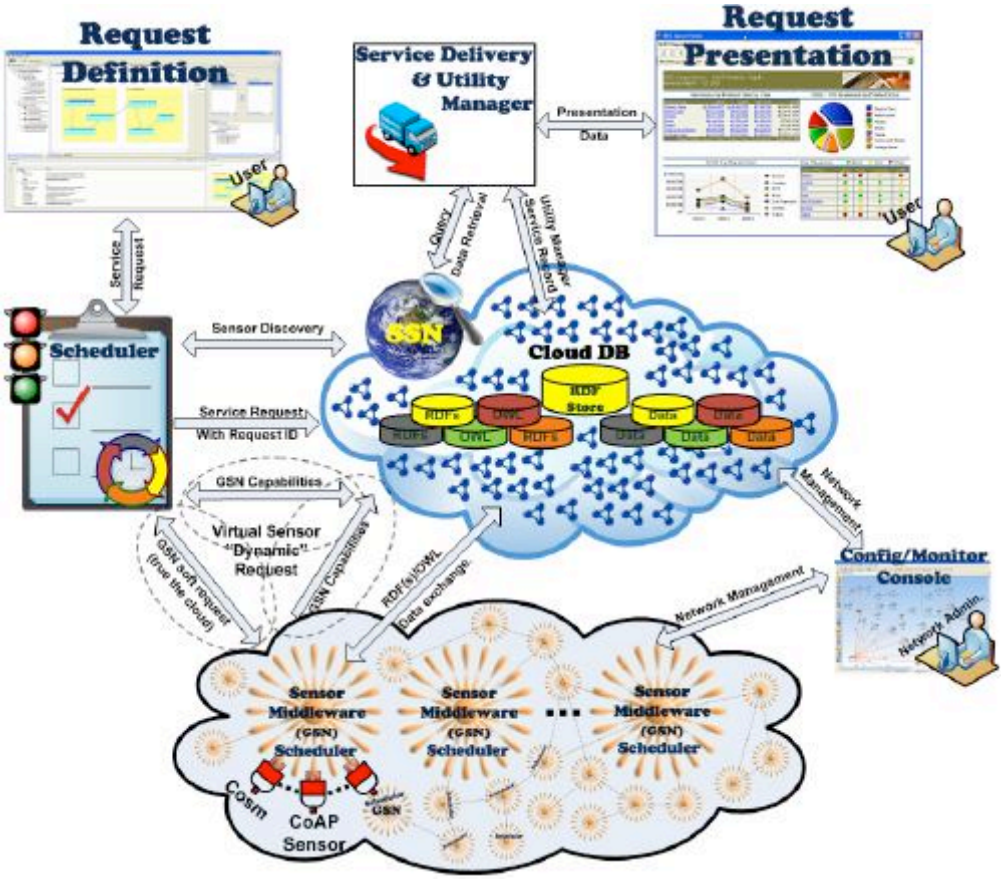


Figure 2: OpenIoT Architecture Overview (functional view)

In addition, the first PoC implementation enables the identification of issues, problems and potential needs for platform redesign that may arise during platform usage. The PoC implementation requires a minimal set of modules and services, from the ones described in high level above and in detail in deliverable D2.3 (OpenIoT Detailed Architecture and Proof-of-Concept Specifications), to be implemented. A high level view of the core modules and the main entities involved are depicted in Figure 3. The PoC architecture modules respectively provided are:

- **Directory Service**: The directory service is provided in the form of an RDF Store (annotated database using RDF format) also called triple store that is accessed through the LSM-Light module.
- **GSN-RDF/SSN integration (X-GSN)**: The original GSN implementation has been upgraded to an OpenIoT version, named Extended GSN (X-GSN), which can announce sensors and semantically annotate received data streams.
- **Scheduler**: The Scheduler provides a basic implementation of its complete API (i.e. sensor discovery and application/service management).
- **Service Delivery & Utility Manager (SD&UM)**: The SD&UM provides the basic implementation of its complete API which mainly interacts with the "serviceDescription" entity, "virtualSensorsDataStorage" entity and the Request Presentation UI.
- **Request Definition UI**: A basic functionality of discovering sensor models and building service requests is supported. The module which the Request Definition UI interacts with is the Global Scheduler.
- **Request Presentation UI**: A basic functionality of requesting of available services of a specific user, polling for data regarding a specific service and visualizing them is supported. The module which the Request presentation UI interacts with is the SD&UM.

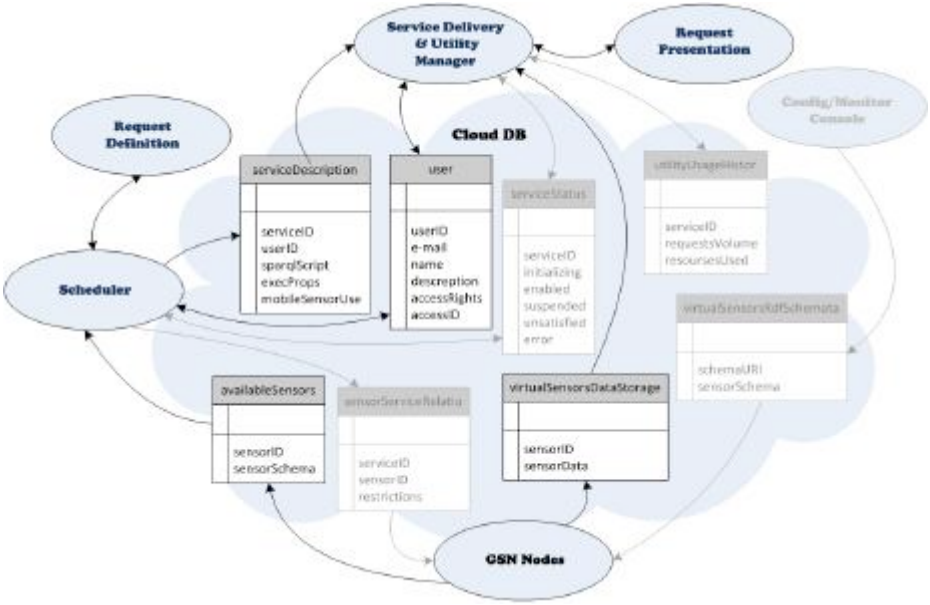


Figure 3: OpenIoT PoC Modules and Entities

The open source implementation of the PoC is available as open source software: https://github.com/OpenIotOrg/openiot/

2.1.2 Data Flow

Figure 4 below depicts the OpenIoT PoC implementation by providing a complete example of the platform's data and service flow from the deployment, configuration and presentation. The flow can be outlined as follows:

- X-GSN nodes are "announcing" the available virtual sensors to the Directory Service and start to publish their data in SSN compliant RDF format based on each X-GSN local configuration (Step 0).
- A User requests from the Scheduler (Step 1) all the available sensor types that satisfy specific attributes (coordinates and radius) by using the Request Definition UI from the Directory Service. The request is sent to the Scheduler service.
- The Scheduler executes (Step 2) a combination of queries (SPARQL) to fulfil the previously user specified query provided by the previous step.
- The Directory Service retrieves the data and replies back to the Scheduler (Step 3) with the available sensor types.
- The reply is forwarded to the Request Definition UI from the Scheduler (Step 4) and the retrieved information is provided to the User.
- The User, with the help of Request Definition UI, defines the request by implementing rules, provided by the tool, over the reported sensor types. This information, along with execution and service presentation preferences is then pushed to the Scheduler (Step 5).
- The Scheduler analyses the received information and sends the request (Step 6) to the Directory Service.
- After having configured the request, the User is able to use the Request Presentation UI for visualising a registered Service's data.
- With the help of SD&UM the Request Presentation retrieves (Steps 7, 8, 9 and 10) all the registered applications/services related to a specific User.
- Having selected a specific service, the User requests to retrieve the results related to it. This is done by submitting a "pollForReport" from the Request Presentation to the SD&UM having the applicationID as input (Step 11).
- The SD&UM requests (SD&UM's "getService") from the Directory Service to retrieve (Step 12) all related information for the specific Service.
- The Directory Service provides this information to the SD&UM (Step 13).
- The SD&UM analyses the retrieved information and forwards the included SPARQL script (Which has been created by the Request Definition UI (step 5) and stored by the Scheduler (Step 6)) to the Directory Service SPARQL interface (Step 14).

- The result is sent from the Directory Service to the SD&UM (Step 15), in a SparqlResultsDoc[5] format. Then the SD&UM forwards it to the Request Presentation (Step 16) that also includes information on how these data should be presented.

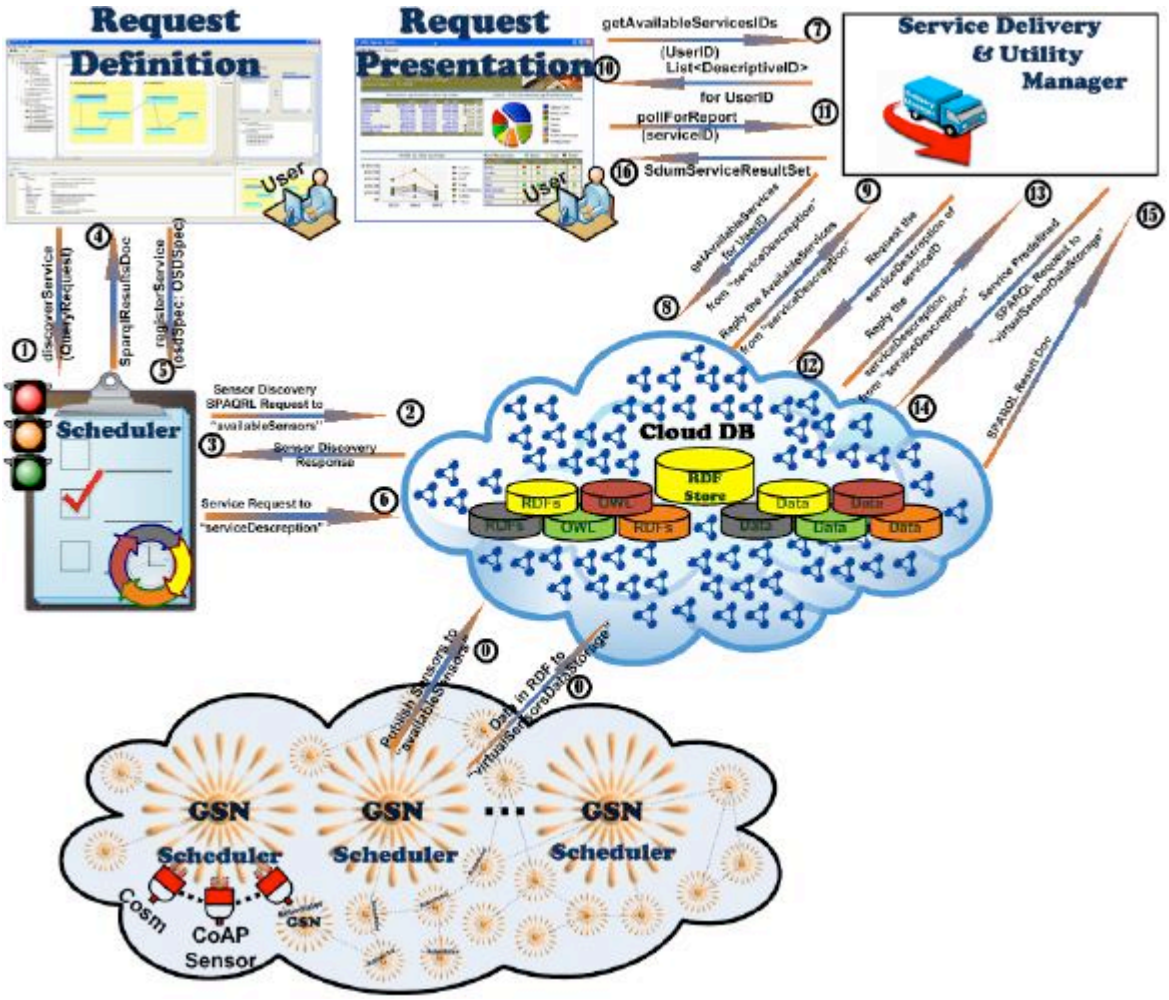


Figure 4: Proof-of-Concept Architecture

2.2 User/Platform Interaction

The OpenIoT architecture was designed with user-friendliness in mind. This mainly involves hiding the complexity to automate and create an environment as pleasant as possible for all the OpenIoT user interfaces and applications. So the core environment of user interaction with the rest of OpenIoT platform is the OpenIoT IDE (Figure 5 below). This environment resides in the higher tier of the OpenIoT architecture, as seen in Figure 2, and is responsible for the sensors, modules

[5] http://www.w3.org/TR/rdf-sparql-XMLres/#defn-srd

management and enabling the configuration of the OpenIoT service delivery capabilities. It provides a single point of entry for all the OpenIoT tools and applications by utilizing Web 2.0 technologies. Currently, the OpenIoT IDE provides the Core Environment, the Request Definition, the Request Presentation, the RDF Schema Editor, and the monitoring tools.

Figure 5: IDE Layout and Main Screenshots

2.2.1 Automatic Formation of IoT Service Delivery Environments

The Service Delivery & Utility Manager has a dual functionality. On the one hand (as a service manager), it is the module enabling data retrieval from the selected sensors comprising the OpenIoT service. On the other hand, the utility manager maintains and retrieves information structures regarding service usage and supports metering, charging and resource management processes.

Global Scheduler (**Figure 2** above) formulates the request based on the user inputs (request definition). It parses each service request and accordingly it interacts with the rest of the OpenIoT platform through the Cloud Database (DB). In particular, the Scheduler performs the following functions:

- **Sensor and ICO selection**. As part of this function the Scheduler queries the OpenIoT discovery service through the "availableSensors" entity in order to find the set of sensors (physical or virtual) and ICOs that fulfil the criteria specified in the scope of service request. Note that the service discovery functionality is based on the semantic properties of the sensors. The Service Discovery components will return to the Scheduler a list of sensors (along with their unique identifiers in the OpenIoT system) that could be used for delivering the specified service (i.e., a list of sensors fulfilling the criteria set as part of the service request).
- **Virtual Sensor "indirect" activation**. Following the selection/identification of the sensors and ICOs that meet the specified criteria and at the request formulation time, the Scheduler will inform the virtual sensors (GSN nodes) about which of them are used by the service being scheduled. This information is kept in the "sensorServiceRelation" entity and is accessed by the virtual sensors.
- **Request Storing and Activation**. The Scheduler is also responsible to provide information to the Service Delivery & Utility Manager (SD&UM) regarding the services to be delivered. This is done through the cloud storage and more specifically the "serviceDeliveryDescription" entity where all the information related to a specific service is stored.
- **Service Status Update**. Through a service lifecycle the Scheduler updates its status at the "serviceStatus" entity. Moreover, it is able to retrieve a service status to inform the User.
- **Access Control**. Finally, the Scheduler implements access control mechanisms with the help of the "user" entity.

2.2.2 Storage and Data Management

Linked Stream Middleware Light (LSM-Light) is a platform that brings together the live real world sensed data and the Semantic Web. The prototype implementation of the OpenIoT platform uses the LSM Middleware, which has been re-designed with push-pull data functionality and cloud interfaces for enabling additional cloud-based streaming processing. An LSM deployment is available at http://lsm.deri.ie/. It provides functionalities such as 1) Wrappers for real time data collection and publishing; 2) A web interface for data annotation and visualization; and 3) A SPARQL endpoint for querying unified Linked Stream Data and Linked Data. The first and third functionality are the ones used in the proof-of-concept implementation in OpenIoT.

In order for LSM-Light to support stream data processing programmatically, a Java API is provided[6]. By using this API, a developer can add, delete and update GSN-generated sensor data into the implemented LSM-Light Server (triple store.

Some of the main OpenIoT architecture's objectives are the dynamic and on-demand nature of the collected and combined data streams. In this case, the first challenge consists in hiding the heterogeneity of the underlying sensor network on one side; and enabling any kind of device to be included in the dynamically-expansible cloud-based infrastructure, on the other side. Then, the middleware is required to provide easily-deployable wrappers that are hardware and software independent. Also, it must be able to directly access the capabilities exposed by each device in a RESTful way. This Data Acquisition Layer provides three wrapper types:

- Physical Wrappers that are designed for collecting sensor data from physical devices;
- Linked Data (LD) Wrappers that expose relational database sensor data into RDF;
- Mediate Wrappers, which allow for collection of data from other sensor middleware such as Global Sensor Networks (GSN), Cosm (aka Pachube), and the sensor gateway/Web services from National Oceanic and Atmospheric Administration (NOAA)[7].

These operations transparently allow users in the upper layers of the architecture, to lookup, select and discovery resources. The middleware, by directly communicating with sensing devices, can maintain an up-to-date directory of Internet-connected Objects that can be queried by users. In order to easily merge the different streams, data are semantically annotated by using RDF and following the Linked Data principles. This Linked Data Layer allows access to the Linked Sensor Data created by the wrappers but linked to the Linking Data cloud.

Data streams can also be queried in those scenarios where live data are most often required, e.g., emergency scenarios. This Data Access Layer provides two query processors, a Linked Data query processor and the Continuous Query Evaluation over Linked Streams (CQELS) engine (Le-Phuoc et al., 2011), and exposes the data for end users or machine users.

Additional tools that can be used to look up and discover the ICO's data, are in the fourth layer, the Application Layer, which offers a SPARQL endpoint, a mashup composer, a linked sensor explorer, and streaming channels.

In the OpenIoT architecture, GSN serves as sensor middleware to publish sensor data from all kinds of physical devices via a common Web service interface. LSM sits on top of it, fetching the data from GSN via HTTP, transforming it into Linked Data, enriching it with semantic information, and storing the data into an RDF storage system. Adding sensors (here: adding sources of sensor data provided by GSN) and acquiring the data is done by LSM via wrappers. In case of data streams from GSN, wrappers apply data transformation rules to map the data in a given format into RDF.

[6] D4.3.1 Core OpenIoT Middleware Platform

[7] http://www.noaa.gov

For example, sensor data in XML can be transformed to RDF using XSLT transformations[8] and the meanings of the sensor readings contained in the XML tags are annotated with concepts in the ontology via an XSLT transformation rule.

2.2.3 Utility Metrics Specifications

We can classify the various utility metrics into two broad categories, namely utility metrics for physical sensors, such as energy, bandwidth, data volumes, and metrics for virtual sensors, which in several cases coincide with those of the physical sensors (see D4.3.1). In addition to specifying metrics for utility driven functions, we review also popular algorithms and schemes for metering and accounting, notably schemes inspired from internet networking. These schemes are transferred and described in the IoT domain. They include flat-rate schemes, time-based schemes, volume-based schemes, smart-market schemes and more. Most of them are applicable to several IoT applications and hence could be implemented as add-ons to the OpenIoT platform. The implementation of all these schemes is however out of the scope of the OpenIoT work plan, yet they can serve as a sound basis for contributions by the open source community.

2.2.3.1 Utility Metrics for Physical Sensors and ICOs

The physical sensors and ICOs are the fundamental (lowest level) data producers in OpenIoT. We can distinguish the following most important utility metrics for physical sensors and ICOs.

1. **Quality:** The quality of sensors and ICOs is the most fundamental metric that determines the accuracy and sensitivity of the measurements provided by a sensor and it may also influence energy consumption. In a given sensor network we may have a mix of high quality expensive sensors and low quality inexpensive sensors.
2. **Energy consumption**: Energy consumption is one of the most crucial utility metrics for sensor systems and more specifically wireless sensor networks (WSN). In WSN energy consumption is directly associated with the lifetime of the sensor network and therefore this metric can be used for functions like accounting, resource optimization and billing. The use of energy consumption in order to measure utility requires the introduction of an appropriate energy model. Most of the energy models are usually simply taking for example into account the (total) number/volume of packets/data sent. Even though the volume of data is analogous to the energy consumption, other factors (such as the energy spent when listening for packets and the energy consumption of sensors' microcontrollers) should be also taken into account. To the extent that OpenIoT will rely on the measurement of WSN energy as a utility metric, the project's platform shall integrate one or more energy models.

[8] http://www.w3.org/TR/xslt.

3. **Bandwidth**: The bandwidth of a physical sensor refers to a bit-rate measure, representing the available or consumed data communication resources expressed in bits per second or multiples of it (bit/s, Kbit/s, Mbit/s, Gbit/s, etc.). In signal processing the word 'bandwidth' is used to refer to analogue signal bandwidth measured in hertz. The connection is that according to Hartley's, the digital data rate limit (or channel capacity) of a physical communication link is proportional to its bandwidth in hertz. Thus, the utility of a sensor can be measured as an available or consumed bandwidth in the scope of an application.
4. **Data volume**: The volume of data (amount of data) produced by a physical sensor can be used as a utility metric. The more data streamed by the sensor, the more the utilization of the sensor. Hence, the utility of the sensor can be analogous to the volume of sensor data streamed or consumed in the scope of an application. Thus, the utility of a sensor can be measured as a delivered data volume in the scope of an application.
5. **Trustworthiness**: The trustworthiness of a sensor can be measured as a trust one can place on a sensor that it will deliver true measurements on time within the scope of its technical parameters. Thus, the trustworthiness is related to the quality of a sensor.

For each physical sensor registered in the OpenIoT sensor directory, the OpenIoT utility manager should keep track of the following five utility parameters: quality (as a semi-static value), energy consumption, bandwidth (in the scope of an application or service or time window), data volume (in the scope of an application or service or time window), and trustworthiness (as a semi-static value correlated with the quality of the sensor).

2.2.3.2 Utility Metrics for Virtual Sensors and ICOs

In practice, several physical sensors that will be integrated with the OpenIoT sensor cloud infrastructure will be in the form of virtual sensors. This is also the case since most of the sensors are likely provided by third-party providers of sensing infrastructures (instead of the OpenIoT cloud service provider) and integrated through the GSN middleware. Therefore, OpenIoT will in several cases have to deal with virtual sensors that announce themselves to the semantic directory service, and accordingly stream their data to the cloud. Furthermore, it is likely that OpenIoT will have to deal with virtual sensors, without knowing or controlling the composition of the physical sensors that contribute to the production and streaming of the virtual sensor data. For this reason, OpenIoT will have to keep track of the utility of virtual sensors, which bear several differences associated from physical sensors.

The following parameters can be used to measure the utility of virtual sensors:

1. **Data Volume**: The volume of data (i.e. number of bytes) streamed by the virtual sensors can be used as a utility metric. The more data streamed by the virtual sensor, the more the utilization of the virtual sensor. Hence, the utility of the

sensor can be analogous to the volume of sensor data streamed or consumed in the scope of an application.

2. **Bandwidth**: Directly related to the data volume of a virtual sensor is the bandwidth (i.e. bytes/sec) consumed/associated with the data volume streamed by the virtual sensor i.e. the rate of data streaming. Similarly to the previous case the utility of a sensor can be analogous to the bandwidth consumed (by the virtual sensor) in the scope of an application.
3. **Time of the Usage Session:** Virtual sensors can be used in the scope of application sessions. The time during which a sensor has been used can serve as a metric for utility calculation. Note however that there are different ways to define the timing boundaries associated with the usage of a sensor (e.g., according to the overall application session where the virtual sensor is used, or the actual time a specific virtual sensor has been occupied). In order to keep track of the time boundaries the start and finish time associated with the use of a resource or a service should be recorded.
4. **Virtual Sensor Location**: The location of a virtual sensor is another prominent parameter that defines its value (or utility). Different locations can signify different business values for the same sensor.
5. **Virtual Sensor Task**: In the scope of business processes, virtual sensors serve some task (process step). The relative business value of this step can drive the definition of the utility or the business value of a sensor. Note that the task associated with a virtual sensor can in several cases be related with the location of the sensor.
6. **Number and type of Physical Sensors used**: A virtual sensor utility metric can be defined and calculated on the basis of the number and types of the physical sensors that comprise the virtual sensor. In particular, a weighted formula can be used to define and calculate the utility of the virtual sensors on the basis of one or more utility metrics associated with the physical sensors that comprise the virtual sensor. In this way, the utility metrics associated with physical sensors could be used in order to calculate the utility of the virtual sensor.
7. **User Defined Cost**: Similar to the case of a physical sensor, a user-defined cost/utility value could be assigned to the sensor. The assignment of the user-defined cost could take into account the above criteria and parameters, but also other criteria defined by the owner, deployer or integrator of the sensors infrastructure.

The OpenIoT platform should keep track of the above parameters as a means to enable utility calculation for accounting, billing and resource management purposes. Resource management concerns primarily the service provider's perspective, while billing and accounting concerns also the end-user's perspective. For the latter, the utility should be ultimately assigned to the OpenIoT service, typically on the basis of a combination of the utility metrics for virtual sensors.

2.2.3.3 Accounting and Billing

In terms of billing mechanisms, a number of different schemes can be adopted based on approaches that have been proposed in literature, notably in the area of charging for Internet resources and services. Note that such schemes consider one of more virtual sensors comprising a service, as well as one or more of the virtual sensor utility metrics outlined above. Some of these approaches are provided below:

- **Flat-rate schemes**: Flat-rate schemes are the simplest billing schemes and are calculated on the basis of fixed tariffs for a specified amount of time. Flat-rate schemes should be based on the assignment of a utility-rate to OpenIoT services, which are typically provided based on the combination of multiple sensors.
- **Time-based schemes**: On the basis of these schemes pricing is based on how long a service is used. In general time-based pricing bills for resources and services on the basis of the time a service or resource is utilized. The usage time associated with a resource or a service should be therefore taken into account. The price can be defined as a function of this time, and more specifically of the start and finish time. This is directly related to the time usage metric outlined in the previous section.
- **Volume-based schemes**: Volume based billing/pricing schemes apply functions over the volume of data incurred in the usage of the service. OpenIoT services will typically comprise multiple virtual (and physical sensors) and hence a volume based scheme will exploit the volume-related utility metrics of multiple sensors.
- **SLA (Service Level Agreement) or QoS (Quality of Service) based schemes**: Pricing is usually based on the quality of the service or the service level agreement associated with the utility services. These can be defined based on one or more of the defined utility metrics, including specific types of sensors, specific locations of the sensors, guaranteed volume of the sensors and more.
- **Priority-based schemes**: Priority schemes have their origins in the Internet, where services can be labelled and priced according to their priority. In this respect, priority schemes are relevant to the SLA/QoS based schemes. In OpenIoT, services can be labelled according to the number and type of sensors that they use, thereby getting some priority over others. Another alternative could be based on the SLA between end-users and OpenIoT service provider.
- **Schemes based on number, type and location of ICOs**: In the scope of internet-based pricing, there have been proposed schemes that calculate bills on the basis of the distance (or number of hops) between the service and the user. As a variation of this scheme, pricing could be based on the number, type and location of ICOs.
- **Smart market based schemes**: Such schemes foresee pricing on the basis of an auction for specific resources (i.e. sensors/ICOs) or services. They can be implemented on the basis of a dynamic market based regulation of the user-defined cost parameters associated with the virtual sensors outlined above. The idea is the more the demand (i.e. number of users/services asking for a sensor) the higher its price (i.e. user-defined cost/utility).

In addition to these schemes, several others can be proposed/derived on the basis of variations and combinations of the above, such as location-based schemes (i.e. charging according to the locations of the sensors) and content based pricing (i.e. pricing based on the type and volume of the content delivered to the user.

The schemes outlined above can be used in order to combine the utility of the various sensors into utility metrics for wider applications or services that comprise multiple sensors. For example, a volume based scheme can be applied towards calculating the utility of an IoT service as a weighted sum of the utility metrics of the various sensors comprising the service.

2.2.4 Cloud Services HMI (Human Machine Interfaces)

The OpenIoT platform comes with a tool (i.e. the request presentation tools) which enables visual development of IoT services. The tool is implemented as a web application, which allows end-users and integrators to visually model their OpenIoT-based services using a node-based WYSIWYG (What You See Is What You Get) UI (User Interface). In this way this tool enables zero-programming modelling and development of IoT services.

The request presentation tools models IoT applications and services as graphs. End-users can therefore access and manage different applications (i.e. graphs) from within a single account to the web based tool. The graphs consist typically of various nodes, which represent data sources (i.e. sensor/ICO sources), filtering nodes (which process and filter sensor data), comparator nodes (which define temporal filters and processing over the sensor data), as well as sink nodes (which provide mashup for the visualization of IoT services). Each graph is accordingly expressed as a SPARQL query/service, which can be later made persistent by the OpenIoT scheduler and subsequently executed over the OpenIoT infrastructure. The tool offers the possibility of validating the graph and the resulting SPARQL that corresponds to the IoT service.

Overall, the request presentation tool provides a powerful HMI for the development of IoT services. The tool is at the disposal of OpenIoT use case developers and integrators and can therefore provide a facility for accelerating the development of IoT services in the scope of the various OpenIoT use cases and scenarios. While the development of scenarios is certainly possible without using the tool, the request presentation HMI can be useful in a number of cases such as: (A) Cases where fast prototyping is needed and (B) Cases where a simple service is (initially) developed and accordingly enhanced based on programming. In this case, the request presentation UI can be used to bootstrap the development of the services, while programming could be employed in following stages.

3 SMART CITY USE CASES

This section describes two Smart Cities use cases: University Smart Campus[9] and Silver Angel.

3.1 University Smart Campus

3.1.1 Description

The University Smart Campus (synonymous with CampusGuide) is an application framework to support students, teachers and guest of a university. It offers features like information's about buildings and rooms, reservations of meetings rooms and workplaces, and collaboration between people. The novelty in this will be the contextualization of all aspects within one semantic framework. Each piece of information will be expressed as an asset in a common ontology and will be associated with related information elements.

The "proof of concept" use case story of the smart campus is about having a smart meeting, including situation and environment aware locations with collaboration situations between people and things (see **Figure 6** below).

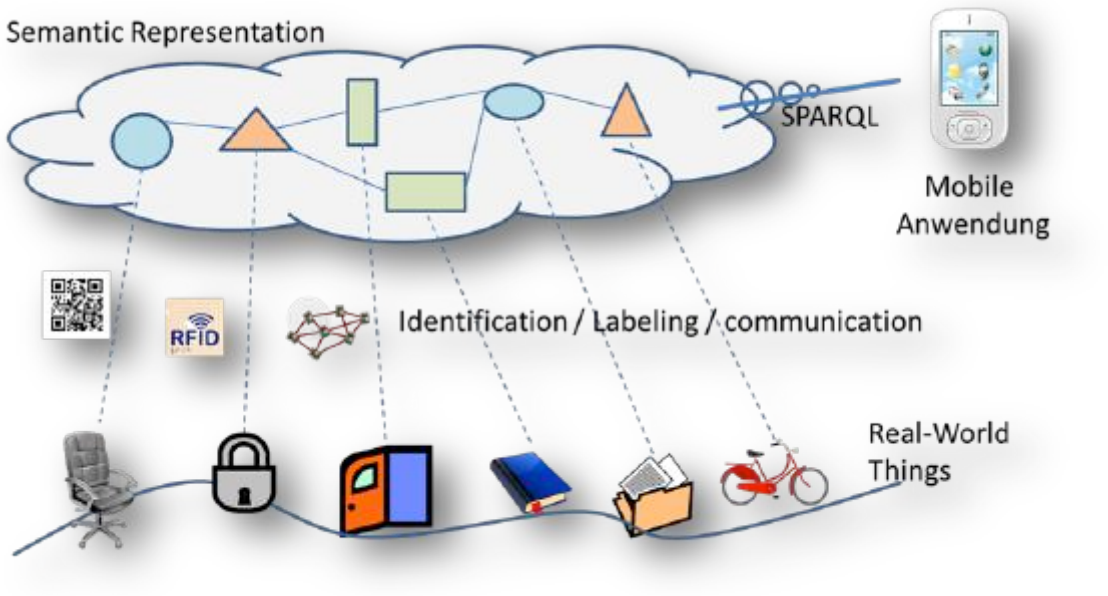


Figure 6: Semantic Context of Things

The meeting rooms will be equipped with sensors providing live data and characterizing different working situations, like collaborative or private working environments. Working materials will be tagged with unique identifiers, providing a

[9] Is used interchangeably with the name "CampusGuide"

URI into a RDF store. Within this RDF store all elements of the use case will be described as concept instances in a common ontology.

The smart campus application will be a mashup of standard applications to be found on a mobile device like a smart phone and new features like live data streams and interactions with and through things. The common ground for the mashup will be overarching semantic model in the OpenIoT Ontology.

The resulting application will be developed in cooperation with University of Karlsruhe and will be called KITSmartCampus.

3.1.2 Scenario and implementation strategy

The implementation will be done in cooperation with students of the Karlsruhe Institute of Technology, the KIT. The development phases therefore needed to be synchronized between the OpenIoT middleware implementation and the semester periods. The priorities of the Smart Campus use case in terms of used the OpenIoT services are distinguished into three categories (see **Figure 7** below).

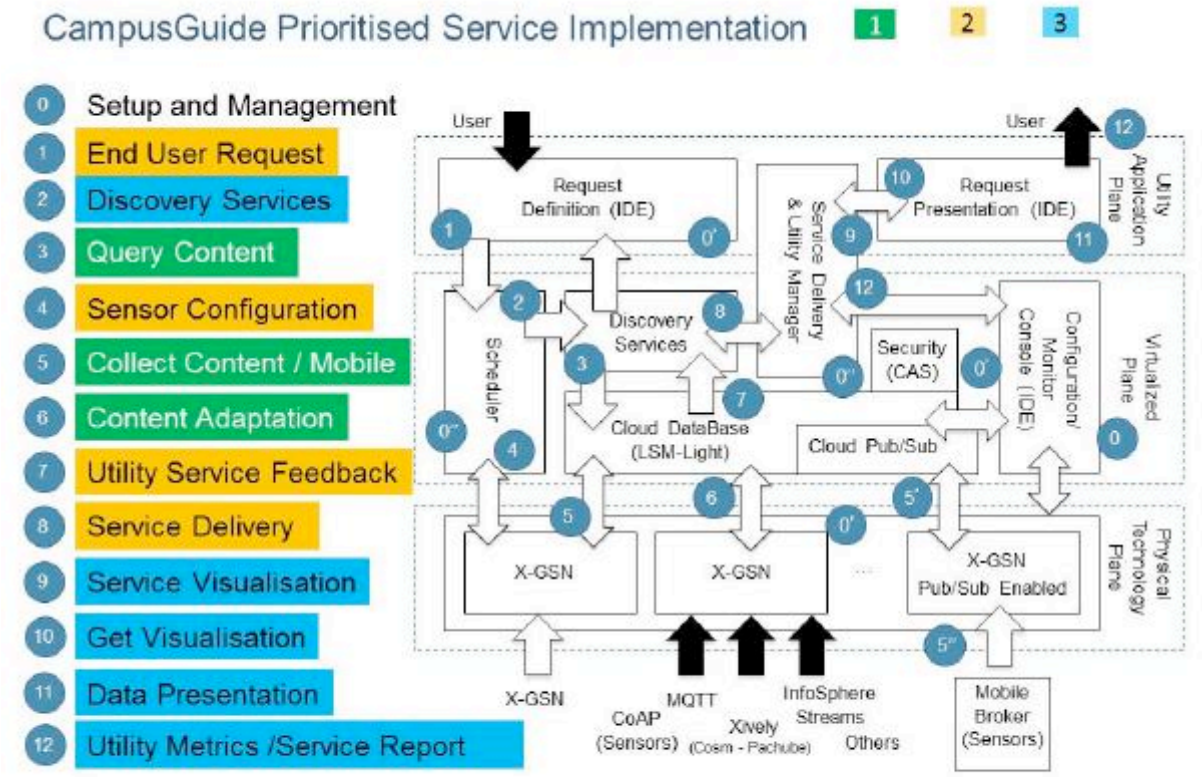


Figure 7: Campus Guide Implementation Strategy

The most important services with priority 1 are needed in the first implementation phase. These are the (3) Query Content in order to retrieve the Ontology data which are the basis for the Smart Campus application. Also the (5) Collect Content services will be needed at the beginning as it is required for collecting sensor data and adopting them to the Ontology with the (6) Content Adaptation service.

The second priority contains all services required to implement the configuration and administration aspects of the use case. These are the (1) End user Request to implement the data access, the (4) Sensor Configuration to handle a large number of sensors in a programmatic manner, the (7) Utility Service Feedback to manage the utilization of the scenario assets, and finally the (8) Service Delivery to get the data.

In the third priority are all interface-related services for the specific utilization overviews of workplace allocations. These are the (2) Discover Service to select specific rooms and buildings, the (9) Service Visualization and the (10) Get Visualization to realise the Status Boards.

3.1.3 Current Architecture and Data Model

The current architecture of the Smart Campus Application is a simple Client-Server Pattern where the server part is utilizing the OpenIoT middleware.

In the Figure 8 below this is shown as a snapshot of the current architecture. In left side of the figure the currently called SmartMeetings Application, which is the Android App, is using the SmartMeetings Backend, which is Server-side application.

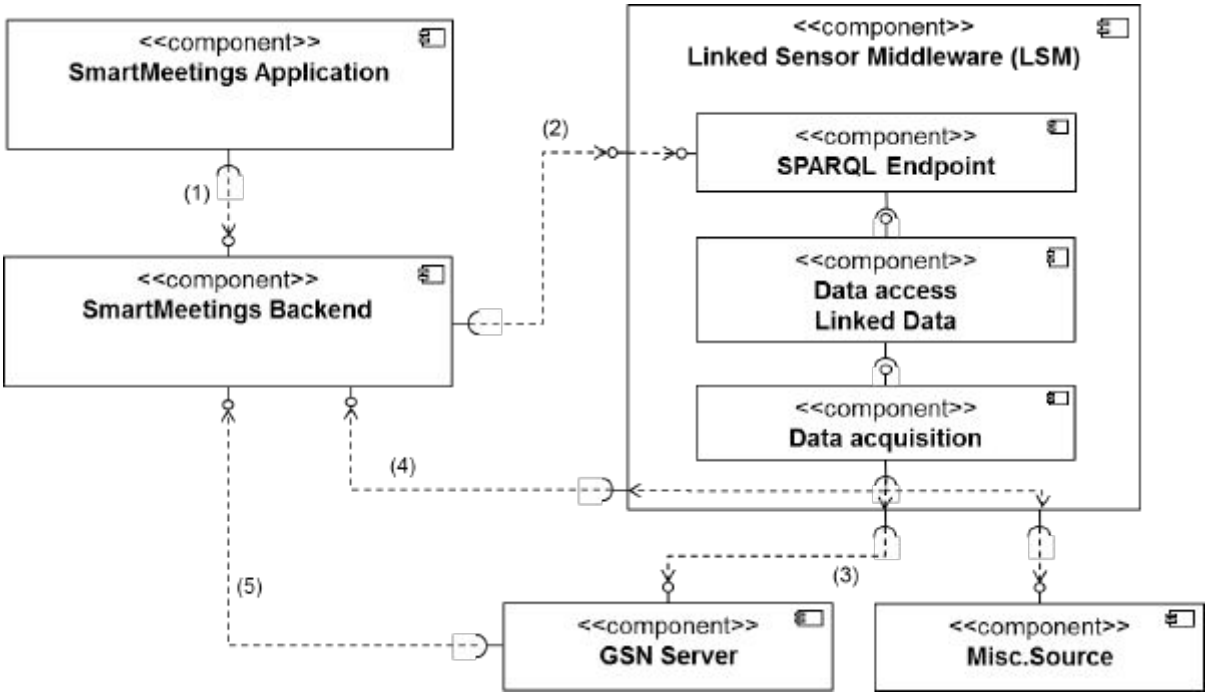


Figure 8: Smart Meeting Architecture Overview

The Smart Campus use case scenario will mainly deal with three different types of data:

- Live data derived from sensor information (based on the SSN ontology);
    - Real world things and electronic assets, describing environment and workspace elements (based on the DUL ontology);
- Collaboration situations within different events including persons and locations (based on the SIOC ontology);

All types of data will be semantically interconnected (see Figure 9) according to the sensor assignment, the arrangement of the involved entities, and the collaboration situation. In the following drawing shows a mapping of a simple collaboration situation with some sample entities to the ontology elements.

For example an office door may be described as a “ssn:FeatureOfInterest” with a sensor for the door lock status, can be observed with an “ssn:ObservationValue”. It also may associated with an “dul:Event”, like a lecture. This lecture may be visited at specific dates as “dul:Situation” by several people described by the their “sioc:UserAccount” which may take part in an adhoc discussion assigned to the “sioc:Site”. Common lecture material may be associated to the event by adding “dul:InformationElement” instances.

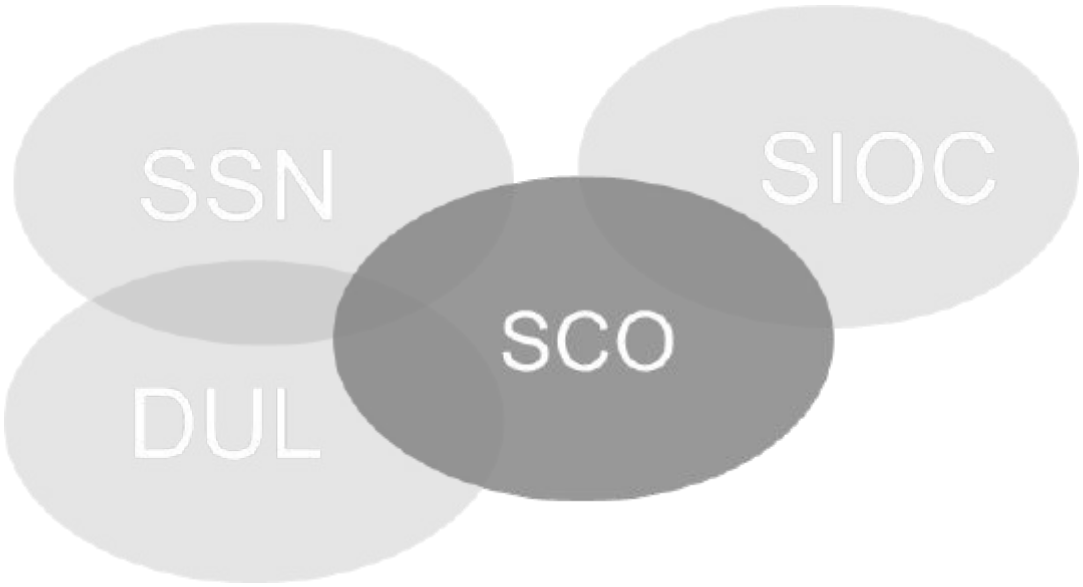


Figure 9: Smart Campus Ontology Alignment Concept

More specifically these alignments are shown in the **Figure 10** below.

The SSN ontology provides a base concept “FeatureOfInterest” where all possible observations are based on. This will be anchor concept for the discussion feature. The discussion will be models according to the SIOC ontology and the association to the FeatureOfInterest will be modelled as an object property “hasForm”. This allows the linking of everything which is a “feature of interest” within the SSN domain to the forum in the SIOC ontology which is the container of discussion postings.

The Buildings and Rooms in the Smart Campus Ontology are models as concept refinement of the “Place” concept from the DUL ontology. Specific Smart Campus properties will be modelled as sub-classes of the Property concept form the SSN.

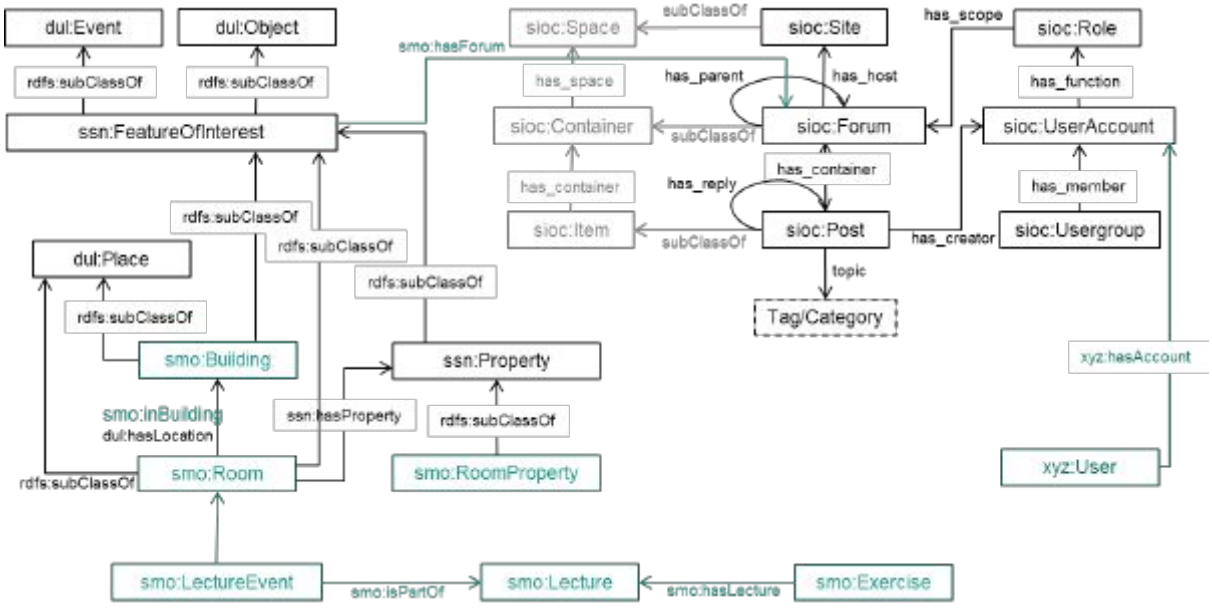


Figure 10: Smart Campus Ontology Alignment Details

An example of a query for a room with a linked forum the following SPARQL-query is shown in the Figure 11 below.

```
1.  PREFIX rdfs: <http://www.w3.org/2000/01/rdf-schema#>
2.  PREFIX smo: <http://lsm.deri.ie/OpenIoT/SmartMeetings/ont#>
3.  SELECT DISTINCT ?building ?label ?buildingNumber ?forum WHERE {
4.        ?building rdf:type smo:Building ;
5.        rdfs:label ?label ;
6.        smo:buildingNumber ?buildingNumber
7.        OPTIONAL { ?building smo:hasForum ?forum }
8.        FILTER contains( ?label, "Fasanen" )
9.  } ORDER BY ?buildingNumber
```

Figure 11: Example query

The result of such a query will be instance of a room with has a link to forum, as shown in the Figure 12 below.

building	label	buildingNumber	forum
'INFORMATIK, Kollegiengebäude am Fasanenga	"INFORMATIK, Kollegiengebäude am Fasanengs	"50.34"^^<http://www.w3.org/2001/XMLSch	
Fasanengarten-Hörsaalgebäude	"Fasanengarten-Hörsaalgebäude"^^<http://w	"50.35"^^<http://www.w3.org/2001/XMLSchTestForum	

Figure 12: Output with new feature "Forum"

3.1.4 Current status and demonstration

The current implementation implements two features:

- Building and Room information with a reservation service;
- Discussion Forums with associations to things;

The first feature (Figure 13) provides the user of the mobile app a search and navigation interface to all buildings and rooms within the KIT. This information has being extracted from an existing facility management system of the university and mapped to the Smart Campus Ontology. Attributes like location and year of construction are also imported into the ontology.

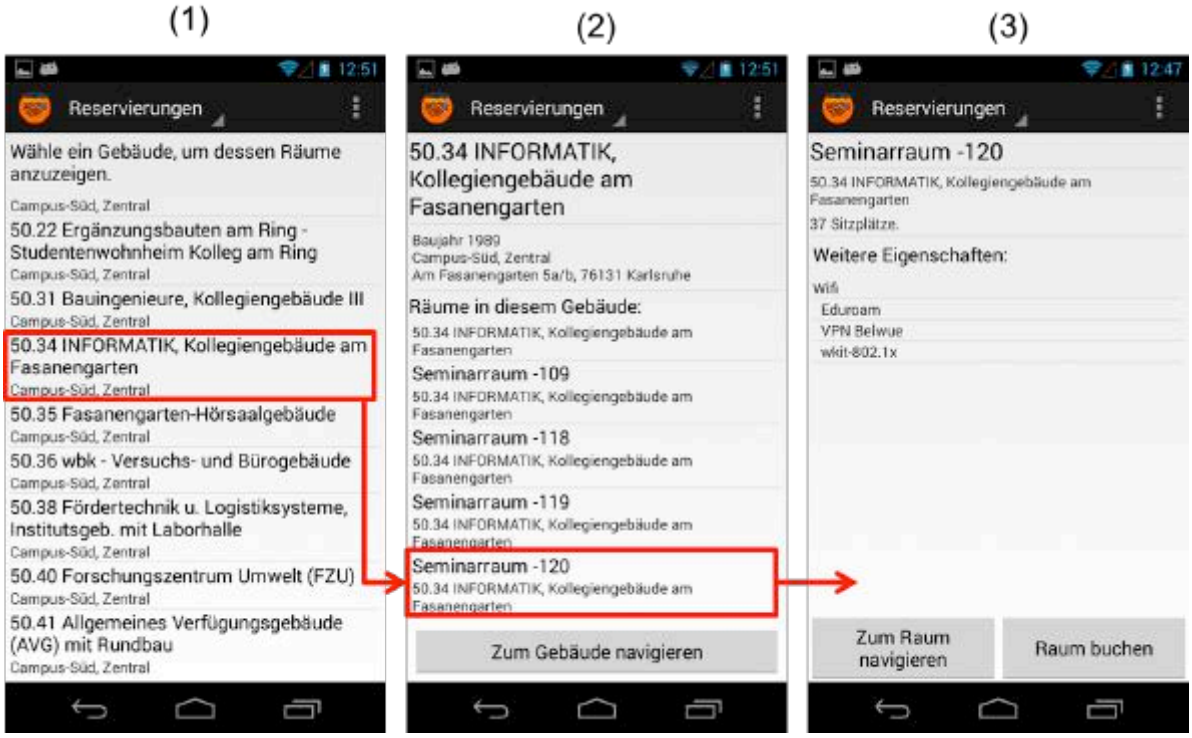


Figure 13: App Interface for Room Information

The user can select buildings (1) which bring a list of available rooms within the building (2). In the building screen (2) the user can request navigation support to the selected building. For that feature is realized as an Android “navigation intend”, which calls the preferred navigation service of the user.

In the room screen (3) the user may again request navigation to the room and also ask for a reservation for that room.

The reservation is currently only a placeholder implementation. In the next phase this will be realized as reservation events which will be implemented as a sensor observation.

The second feature (**Figure 14**) of the current Smart Campus implementation is the smart discussion.

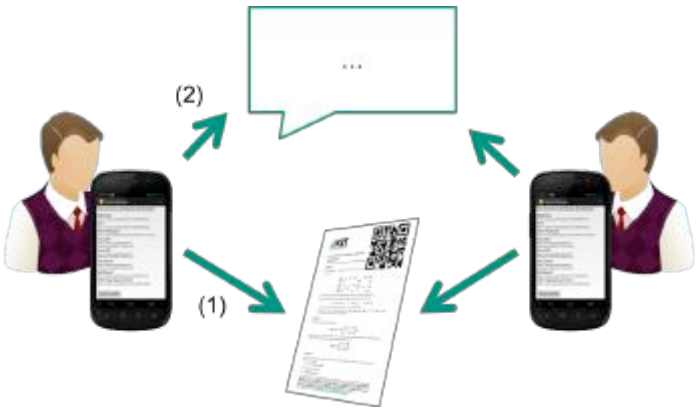


Figure 14: Discussion via Things

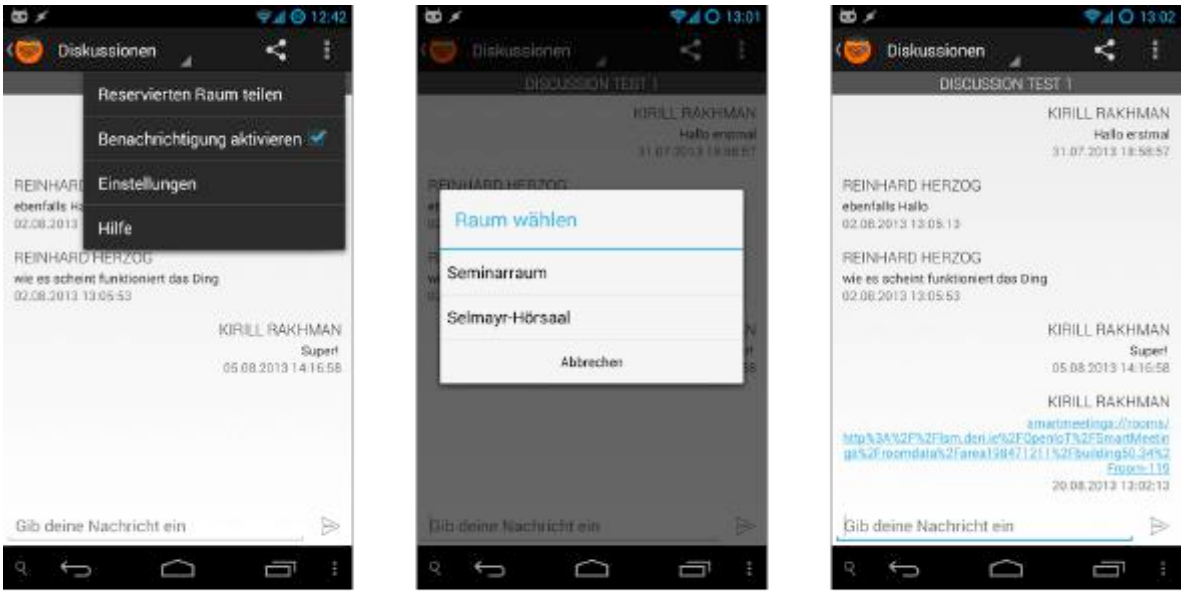
A discussion can be created from a specific room or directly from the main screen of the app. It is also possible to open a discussion by sharing the URI of a discussion by scanning an NFC-Tag or a QR-Code (Figure 15).

Figure 15: Discussion Sharing via QR-Code

Discussions are created as private elements which must be shared explicitly with other users (Figure 16).

Figure 16: Starting a Discussion for a Room

Room information can also be linked into a discussion as shown below in Figure 17. In that case a URL of a specific room is created and being posted in the discussion.



(1) (2) (3)

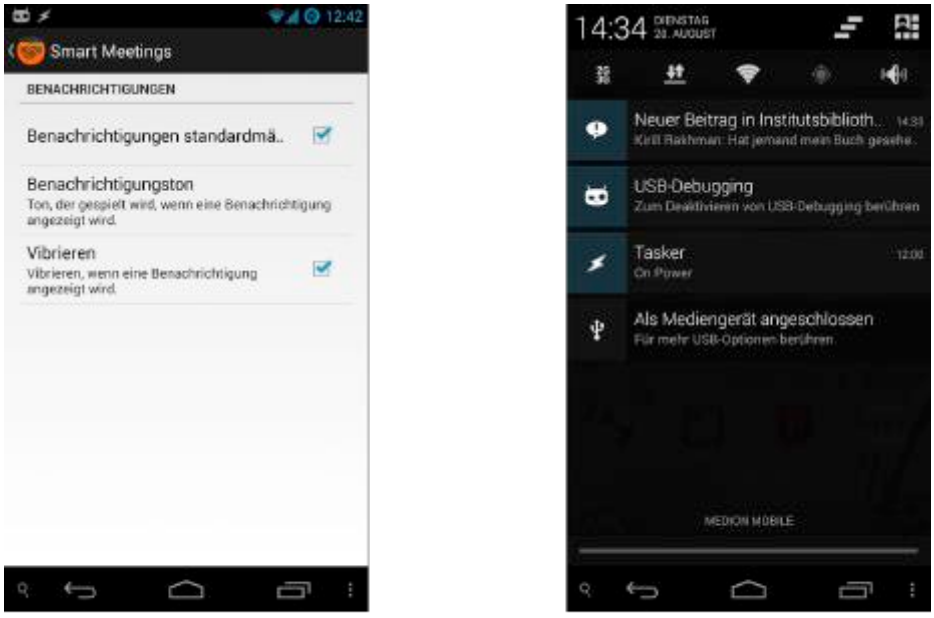
Figure 17: App Interface for Room Sharing

The discussion itself is realized in a typical chat style, as shown in the figure below. All postings are chronologically ordered. The own postings are one the left side and all other postings are on the right side for better orientation. On the bottom line is the entry field for new postings (Figure 18).

Figure 18: Discussion Sharing via QR-Code

In order to reduce network traffic and start-up time, only the latest items are loaded. Older entries are loaded on demand, if the user scrolls up.

New postings can be delivered on a push basis. For that purpose the Google Cloud Messaging (GCM) service is used. All settings are availed via the Overflow- Menu in the standard action bar (Figure 19).



(1) (2)

Figure 19: Push-Notifications

The standard action and sound for notifications can be defined in the settings. It is possible to overwrite this standard setting for specific discussion. The notifications for new postings are shown with a unique symbol and a preview in the overflow menu.

3.1.5 Future work

Currently in work is a feature to use the sensors within the mobile device to provide observations which are related to the room where the user just confirmed his reservation (Figure 20).

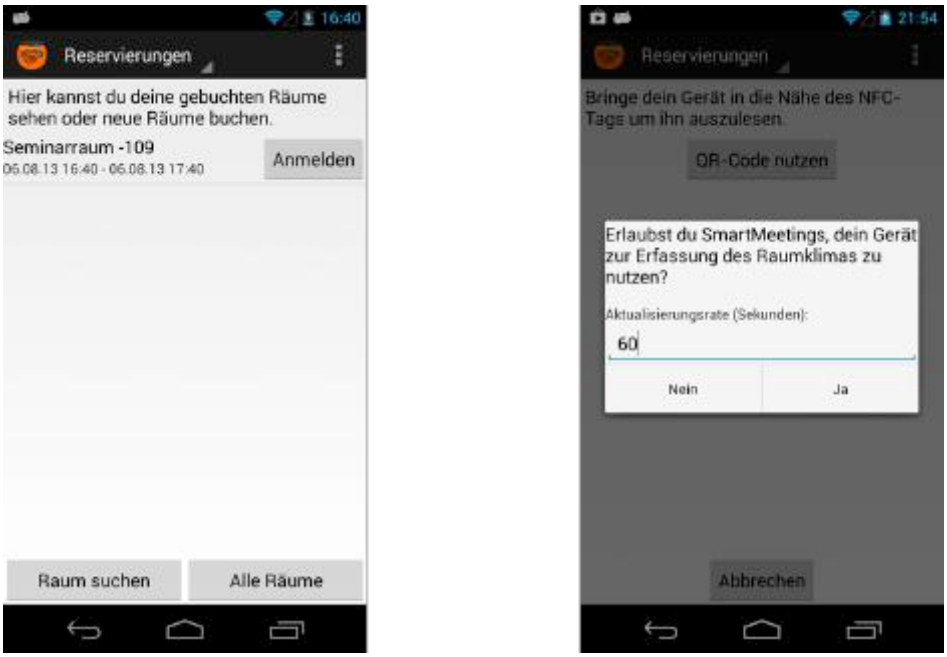


Figure 20: User Interaction for Crowd Sensing

The observations are shown as additional attributes of the room (Figure 21).

Figure 21: User Interaction for Crowd Sensing

The OpenIoT middleware provides a technology for continuous query evaluations based on a semantic model (Ontology). This method is based on a visual representation of the semantic query language (SPARQL) and a continuous reasoning evaluation engine. With this technology it is possible to evaluate a "world model" containing semantically annotated data about real world things, live sensor information associated to these things and social networking content for collaboration support.

The plan is to use these possibilities to enrich the Smart Campus application by providing semantically evaluated context information to provide a better user experience.

3.2 Silver Angel

3.2.1 Description

The purpose of Silver Angel (as described in D2.3) is to help ageing citizens live independently in their own homes, and to facilitate meeting more often with friends and relatives. There are three main Silver Angel services foreseen as depicted in Figure 22: Smart Meeting, Issue Reporting and Alarms.

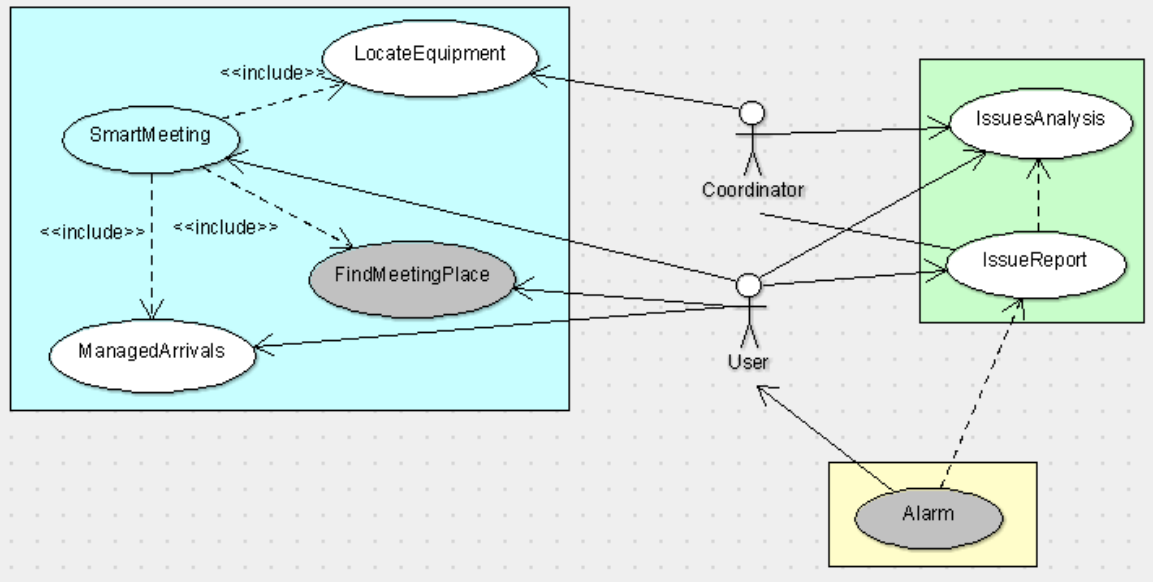


Figure 22: Silver Angel main functionalities

The overall concept is illustrated in a video:
http://www.youtube.com/watch?v=pbdGv9W4CYs

In the context of OpenIoT, Silver Angel uses open data from public administrations and crowd-sourced issue reports from citizens to find preferred locations for meeting with friends. Initially the meeting location criteria can be:

- People (few/many/ignored)

- Noise (low/high/ignored)
- Pollen levels (low or ignored)

However, the intention is to make it easy for city administrators to configure Silver Angel to use other types of criteria, for example air pollution levels. The idea is to create OpenIoT Virtual Sensors that deliver updated views of the city situation according to the user's meeting preferences. This will provide flexible services in finding areas that are less busy, noisy or polluted – and not sensitive for persons with allergies and other health problems.

The people in the city also help out, by anonymously reporting about pollen and street conditions, and letting their mobiles make available key data from the microphone and other sensors. OpenIoT brings thousands of sensors and humans together at city-wide scale, while also being capable of privately serving the personal needs of ageing citizens.

3.2.1.1 Smart Meeting

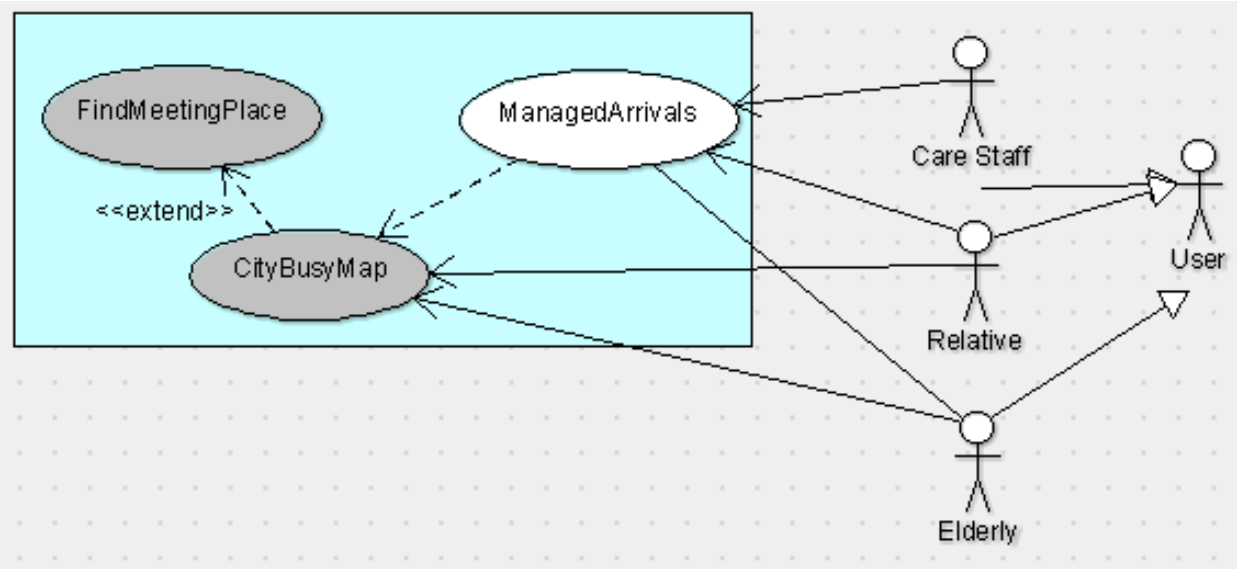


Figure 23: Smart Meeting use case

This service has the main goal of finding the best possible locations and times for two or more users to meet, and then to manage their arrivals.

Finding the meeting place (and time) can depend on various search criteria - such as, for example, availability, busyness, noisiness, pollen levels - the proof-of-concept implementation will provide an interactive *CityMap* that shows the attractiveness for meeting in various areas of the city, and makes it easy to view the more attractive places and schedule meeting with friends there.

The arrivals to agreed meetings is managed in the sense that all participants are made aware of at what time other participants are estimated to arrive, based on their location and the amount of distance left to reach the designated meeting location. The diagram below displays the service's sequence flow.

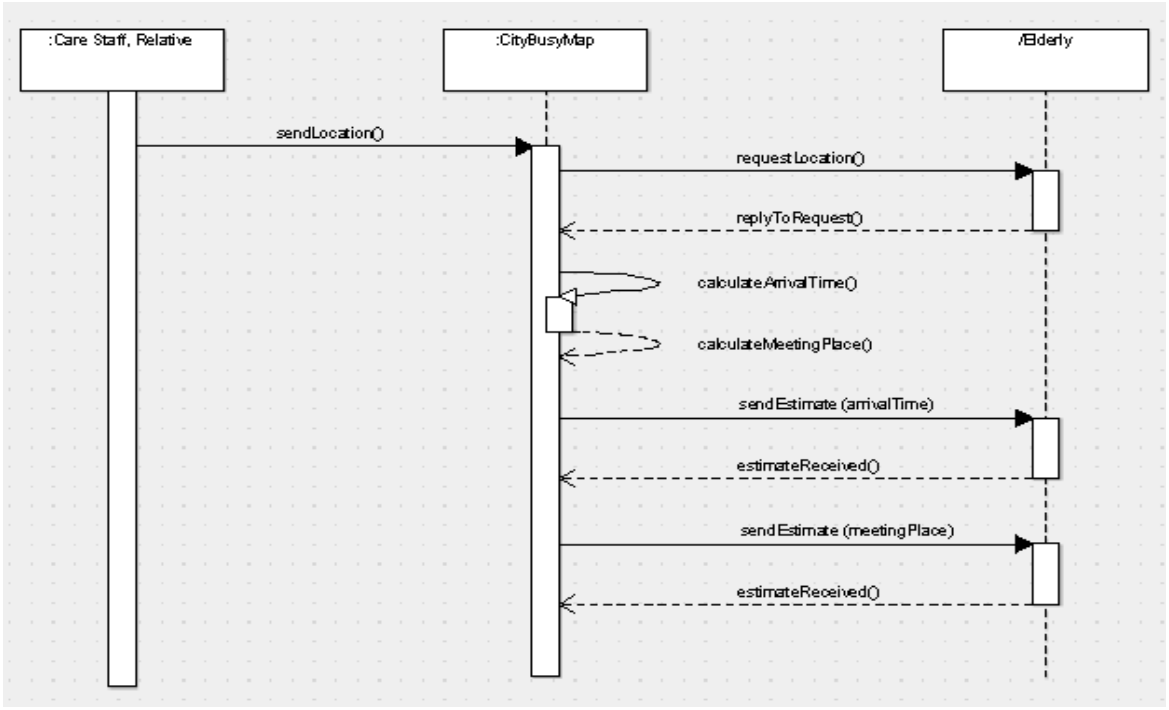


Figure 24: Smart Meeting service data flow diagram

3.2.1.2 Issue Reporting

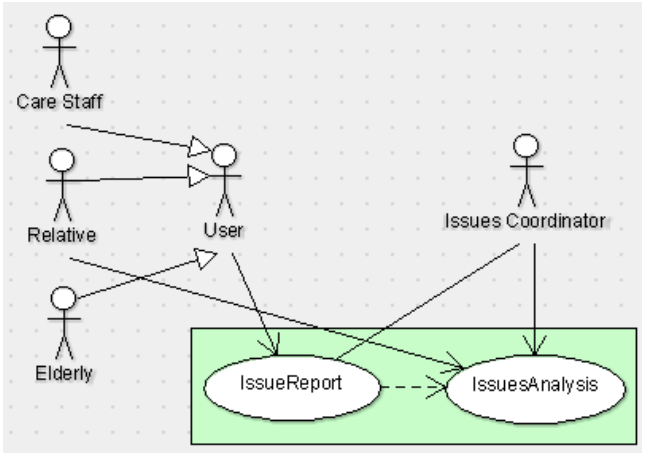


Figure 25: Issue Reporting use case

This service allows any user to report issues and be notified of any progress being made to fix an issue. This service will typically be used for human-reported issues in a city, but can also be used for automatic issue reporting (see *Alarms* below). The user can report any observed issue – for example, pollen levels, risks, problems, suggestions and points of interest that can help others decide where to meet. It can also be used for reporting street conditions, traffic accidents, full garbage containers etc., that immediately show up in a prioritised list for the Issues Coordinator within the local administration. Reported issues can be searched and analysed by the Issue

Coordinator, who will decide what issues to either address directly or to dispatch to someone else.

The diagram below depicts the main sequence flow of Issue Reporting.

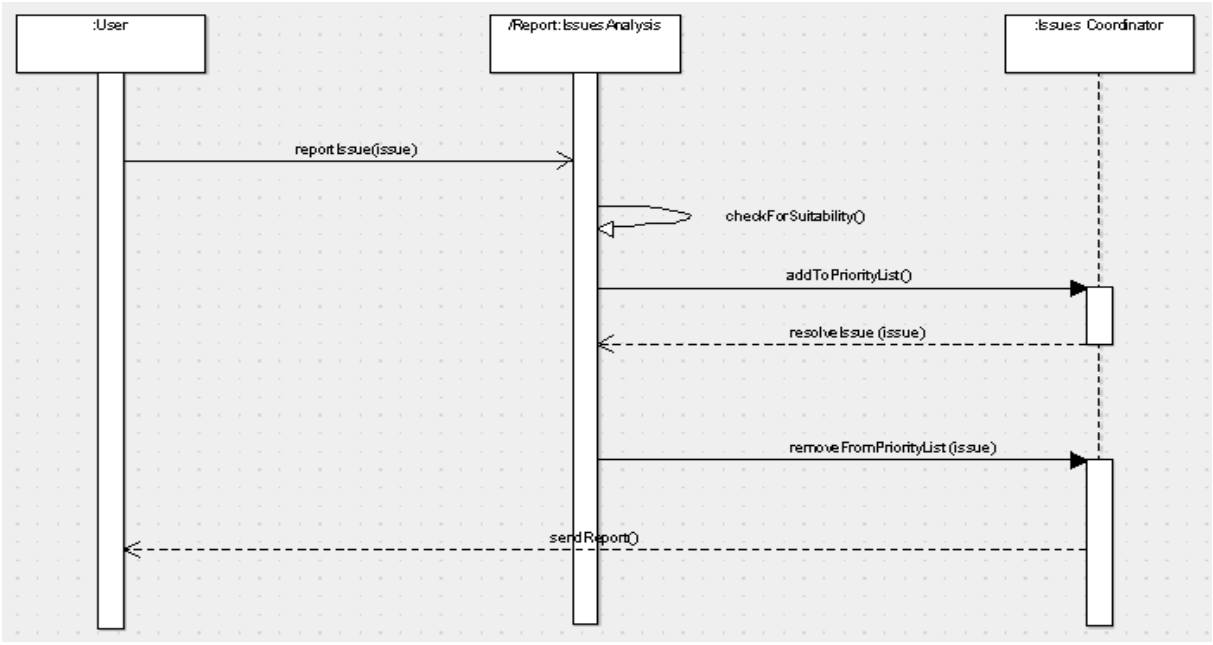


Figure 26: Issue Reporting data flow diagram

3.2.1.3 Alarms

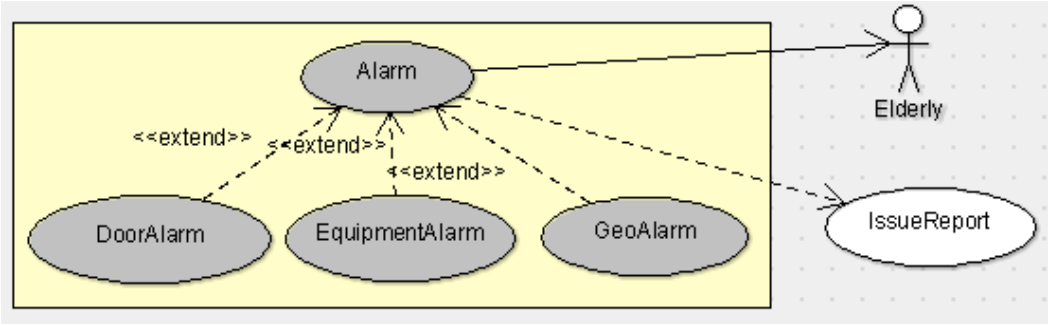


Figure 27: Alarm use case

In the Alarms scenario, which is in the context of the home of an ageing person, equipment, appliances and other objects of interest around the house or mobile phones can report issues that they have detected. For example, doors can be sensorised so that they can report when the door is open or closed (or locked/unlocked), appliances can report whether they are on or off, and rooms can report when the temperature is above or below a pre-determined threshold.

The diagram below depicts the scenario sequentially.

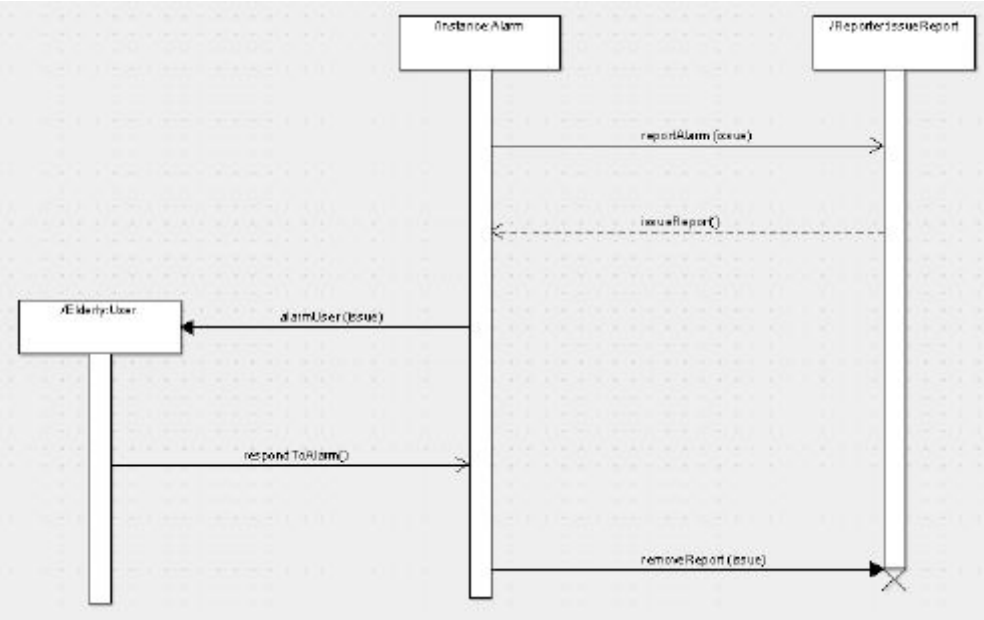


Figure 28: Alarm data flow diagram

3.2.1.4 The Use Case Story of Silver Angel

Emma is a 65-year old, fairly vital lady. She lives alone, but her daughter and son with families are living not very far away. Since Emma has developed some motor problems after a minor stroke, she receives house calls every day from the local care service. Emma has subscribed to the Silver Angel service which gives her a phone watching out for her, and makes it easy to call her loved ones and carry out daily tasks.

Today is a big day for Emma, as she has agreed with her friends Lucia and Maria to enjoy the city, but they did not yet agree where to go. Since Emma prefers to be in more quiet areas, she wants to check which parts of the city are less busy. Emma therefore activates the Silver Angel, where it immediately shows which areas of the city that are busier with people and noise for the moment. She remembers that Lucia suffers from pollen allergy, so she also ticks the box to show where pollen is more intense today. It turns out that Memorial Park looks the nicest. She ticks Lucia and Maria as participating in this activity, and by doing so they both get notified in their Silver Angel phones. Since the agreed time is three hours away, all three of them get on with other activities of their respective days. 15 minutes before the agreed time, Emma – who likes to arrive early to appointments – is in Memorial Park. Silver Angel has already started to indicate how far away Lucia and Maria are from the park. – Lucia will be a little late as usual, Emma mutters to herself, but that is OK, because she is such a sweet darling.

When Lucia goes home from the park, she passes an area where the trees just started blossoming. She immediately gets breathing problems and has to use her inhaler. When she feels OK again, she uses her phone to report the local pollen problem to the Silver Angel.

The next day, Emma will receive a visit by the physiotherapist Joey. She often stresses a bit about these visits by non-relatives, because she feels the house must be in perfect order then, and worries that she might miss him if she goes out to buy some missing groceries. Silver Angel again comes to the rescue. She verifies that Joey will arrive exactly on time, and that therefore she can walk down to the corner bakery to buy some nice muffins for the coffee. When Joey is five minutes away, Emma's Silver Angel signals that he will soon be there. Emma greets Joey with a smile.

During the evening, Emma sits in peace listening to the radio and solves Sudoku puzzles. When she sat down, Silver Angel notified her that the front door was not locked, as it should at this time. She goes to lock the door, and can then enjoy the evening without worrying about having forgotten to close or lock doors and windows. Emma's daughter, who frequently calls on her to check that everything is OK, is away to another city this evening, confident that she would receive an alert for any problem and that it is easy to check for indications of negative trends in her mother's life.

3.2.2 Scenario and implementation strategy

Silver Angel is mainly interested in reasonably finely granular information about person locations, the noise level in those locations, and more sporadic pollen problem reports (each user report weight decaying fairly quickly, let's say by 80% in three days), as well as air quality readings in a few strategic locations (and mobile on some buses).

Figure 29 illustrates a staged implementation of the Silver Angel app.

Initially (2013), existing open data sources will be used (Collect Content and Service Delivery) while seeking partnership with local administrations for deploying sensors to generate new valuable open data for city awareness. The Silver Angel app itself then will do multivariate interpolation in a 2D+time space (thus three dimensions), and apply suitable thresholds. The user interface will be implemented as a web application (End User Request and Get Visualisation) that uses pre-configured OpenIoT data services to deliver updated city attractiveness information according to the user's preferences.

In a second stage (spring 2014), Silver Angel Reporting will be added to feed OpenIoT Virtual Sensors for human-reported pollen levels and dynamic points-of-interest (Sensor Configuration and Collect Content). The OpenIoT integrated Development Environment will be used to demonstrate its capabilities to generate dashboards for Issue Coordinators at local administrations (Service Visualisation).

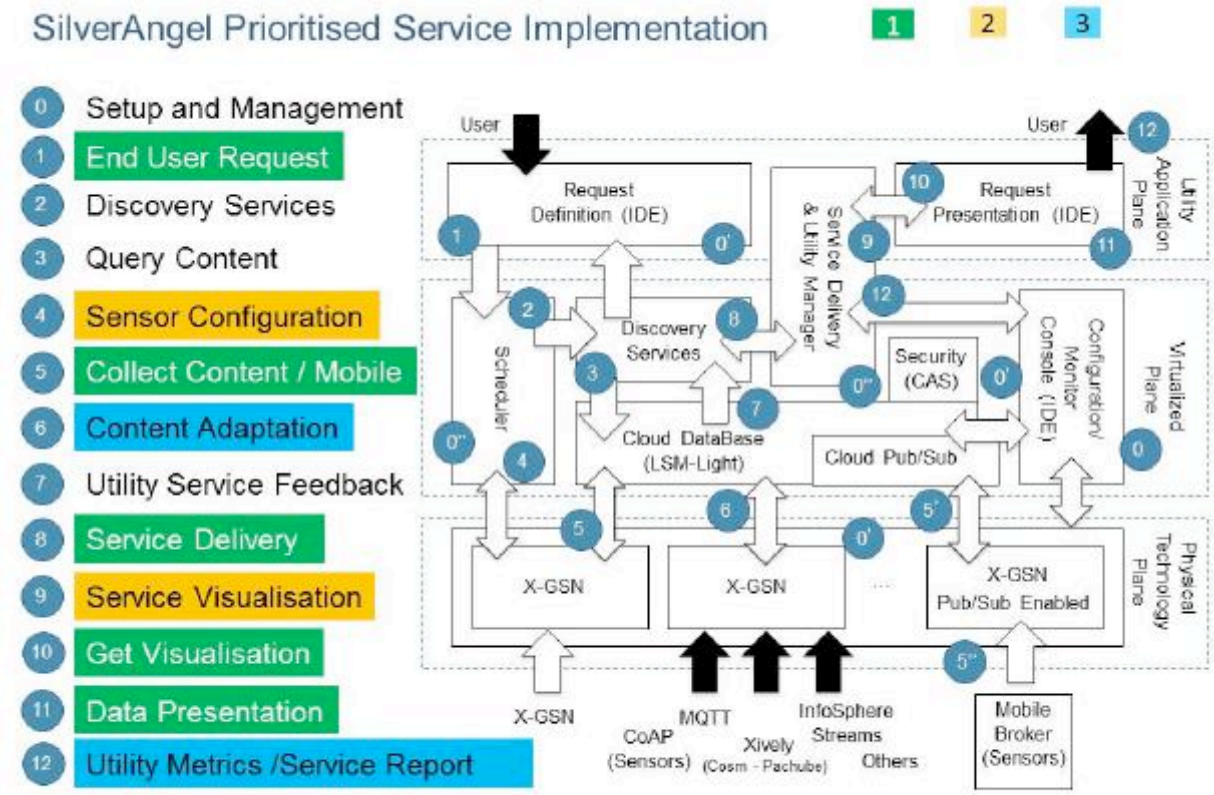


Figure 29: Silver Angel staged implementation

In the third stage (autumn 2014), the city administration support will be improved, to allow easy reconfiguration of the underlying data services for a Silver Angel deployment, using the OpenIoT Request Editor (End-User Request and Sensor Configuration). Mobile optimisations using the Publish-Subscribe mechanisms will be added (Collect Content) and data aggregation will be used for generating triggers and alarms (Content Adaptation).

3.2.3 Current status and demonstration – Stage 1

The current Silver Angel version is v3.3. It has the basic user interface for Meeting and Reporting, as well as for easy calling:

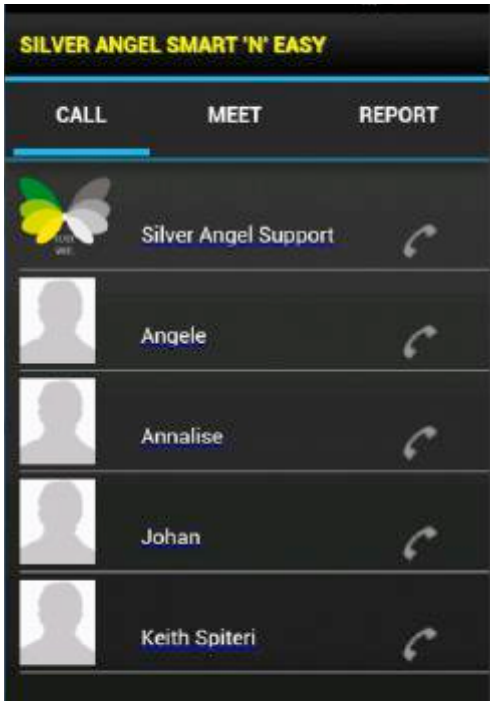


Figure 30: Silver Angel – Call screen

In the Meet screen, a map with suggested nearby meeting locations is shown:

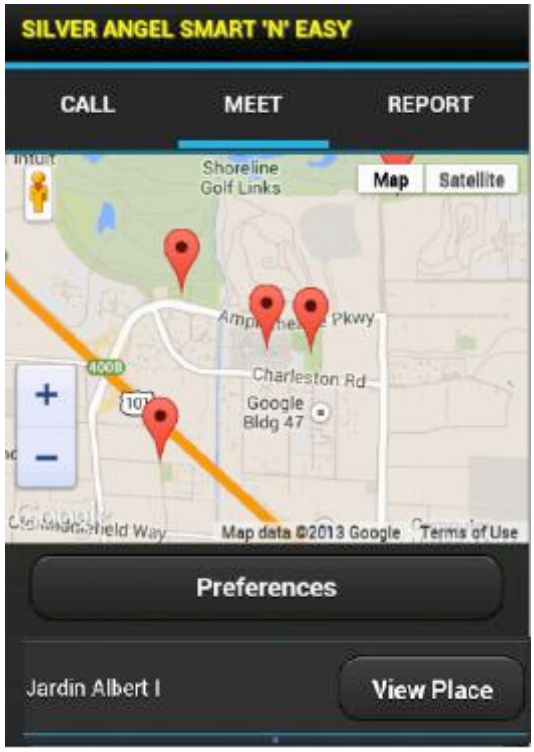


Figure 31: Silver Angel – Meet screen

The current location of the user is used to identify attractive nearby meeting locations. The location is obtained using the JavaScript *position.coords* element.

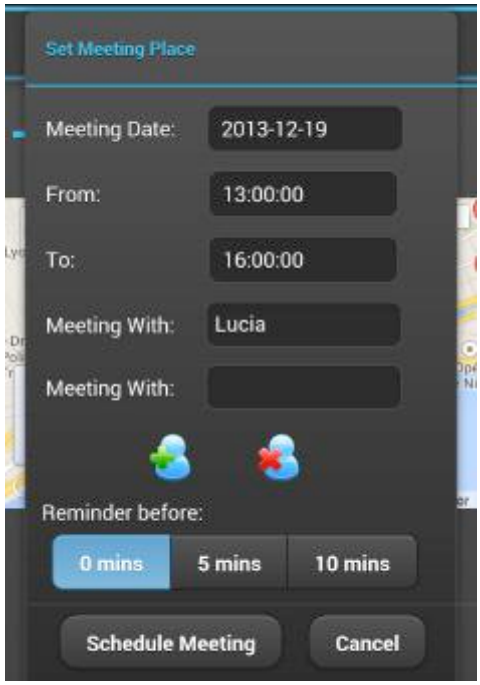


Figure 32: Silver Angel – Scheduling screen

In the Scheduling screen (above), the meeting time is set and the intended participants are selected. The meeting gets added to the Silver Angel of all participants, and an email is sent with a vCal file for easy adding to their calendars.

The user can easily select personalised preferences:

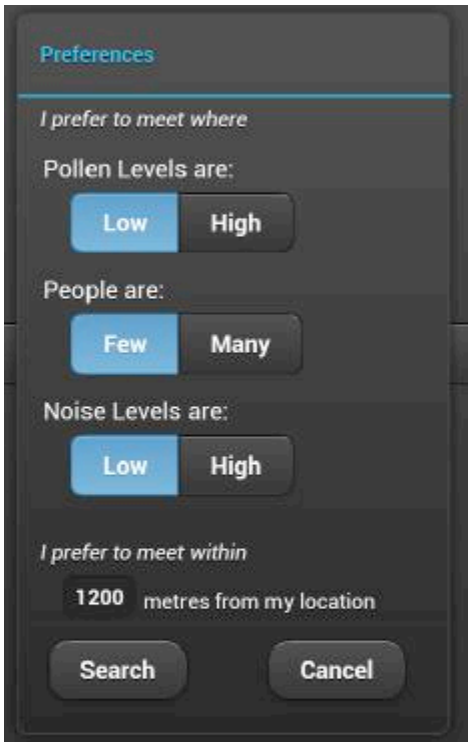


Figure 33: Silver Angel – Preferences

In the Report screen, different issue categories can be reported:

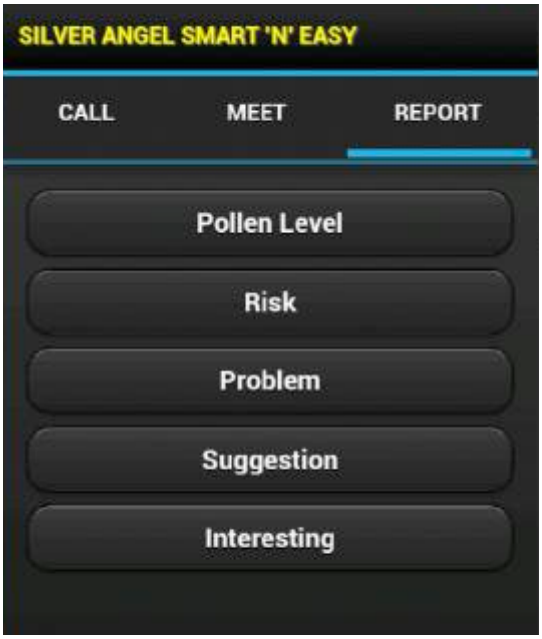


Figure 34: Silver Angel – Report screen

For Pollen Level reporting, a traffic light model is used:

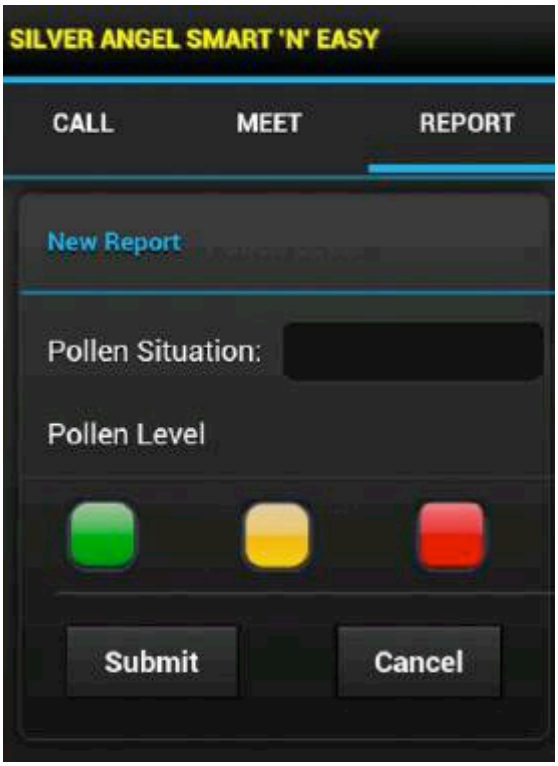


Figure 35: Silver Angel – Pollen Level Report screen

3.2.4 Future work

The Silver Angel App is a work in progress. The above descriptions are subject to change.

3.2.4.1 Stakeholder Engagement

Silver Angel relies heavily on the Open Source OpenIoT middleware for enabling local administrations to add new sensing capabilities that can be used for suggestions where to meet. The OpenIoT open source Integrated Development Environment can then be used to easily create dashboards for city administrators to get an overview of the city characteristics that are most important to its citizens.

It is therefore essential for Silver Angel to engage a local administration for creating a real demonstrator. The Stage 1 prototype and the air quality sensing by the FER partner will be used for this purpose, to explore the interest among local administrations related to AcrossLimits.

For a city to get started with Silver Angel, some ambient sensors will need to be deployed first to attract interest, and only then will the user base grow to provide additional data from the sensors in the mobiles.

For Silver Angel, the data needed would appear and more data sources would be added as follows:

1. **Air Pollution** - sensors deployed by local administration - we aim to use air quality sensing from the FER partner (as part of OpenIoT-Enlarged) and are then going to work with a local administration to deploy a number of them (Luleå, Sweden are currently carrying out a pre-study on how to provide Open Data, and air quality sensing is a prime candidate).
2. **Noise** - from the mobiles carried around by users of the Silver Angel app, and also from static devices deployed by the local administration – noise detection will then have to be added to the air quality sensors.
3. **People** - from the mobiles carried around by users of the Silver Angel app.
4. **Pollen** - from manual reporting by motivated people using the Silver Angel app.

3.2.4.2 Software Development

We use an iterative development approach, progressing towards the above description of the full Silver Angel app, and full use of the OpenIoT middleware. Below are given detailed implementation notes from trying to use the OpenIoT middleware in Silver Angel.

Firstly, since the OpenIoT platform itself only provides historic samples of open data, we need to push own data relevant for Silver Angel into the installed platform. For a limited period of time, we aim to continuously push open weather and noise data to the OpenIoT platform for Malta, Brussels, Luleå, Galway, Lausanne, Karlsruhe, Zagreb and Brisbane. The use of X-GSN for pushing user-entered data about pollen levels and other data sensed in the mobiles is being explored. We have encountered

some integration difficulties that are being resolved, and this will be reported in D6.3.2

To be able to push own data to the OpenIoT platform, it is necessary to create X-GSN sensor metadata and description files (**Figure 36** and **Figure 37**).

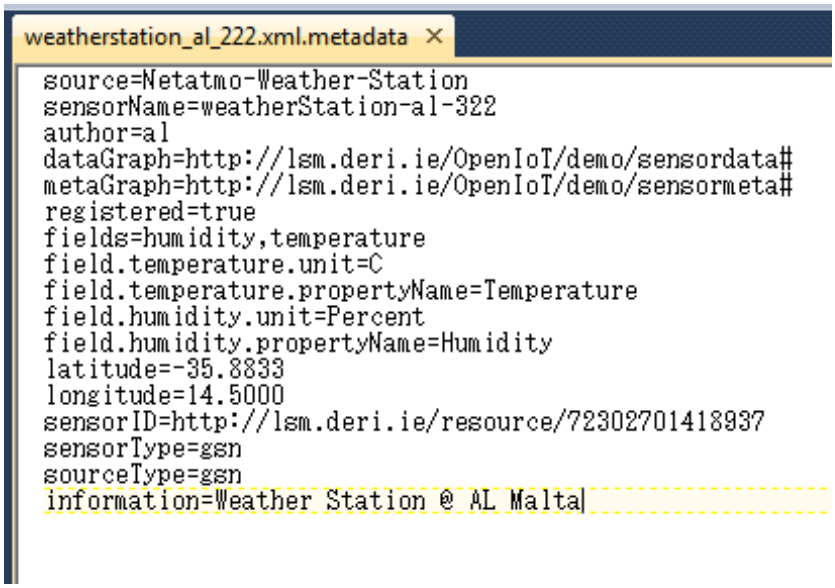


Figure 36: X-GSN sensor metadata file for weather stations

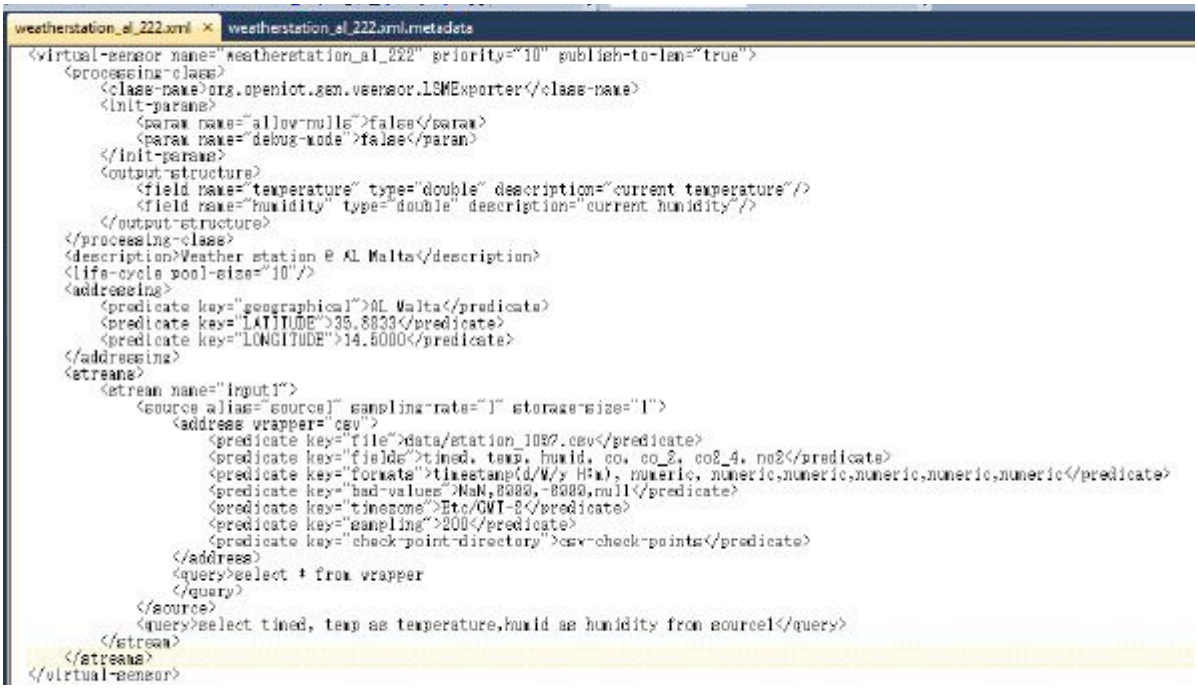


Figure 37: X-GSN virtual-sensor description file (XML) for weather stations

After successfully registering virtual sensors in the OpenIoT platform (details omitted, refer to the OpenIoT middleware documentation), SPARQL queries need to be defined for getting the data streams. The OpenIoT Request Definition tool was used to get a SPARQL to be used in the Silver Angel app. We tried first to get a data stream of average noise readings from Canberra as show in **Figure 38**.

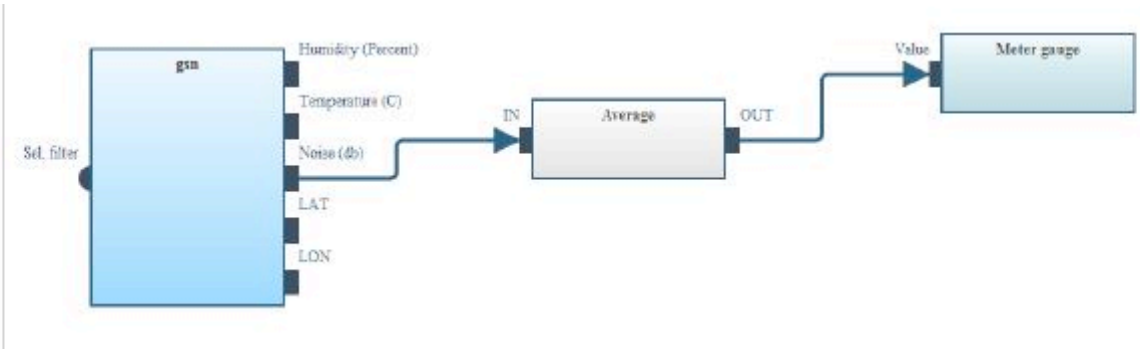


Figure 38: Canberra Average Noise Definition

The Request Definition tool generated a SPARQL query was generated so that it can be used in Silver Angel code.

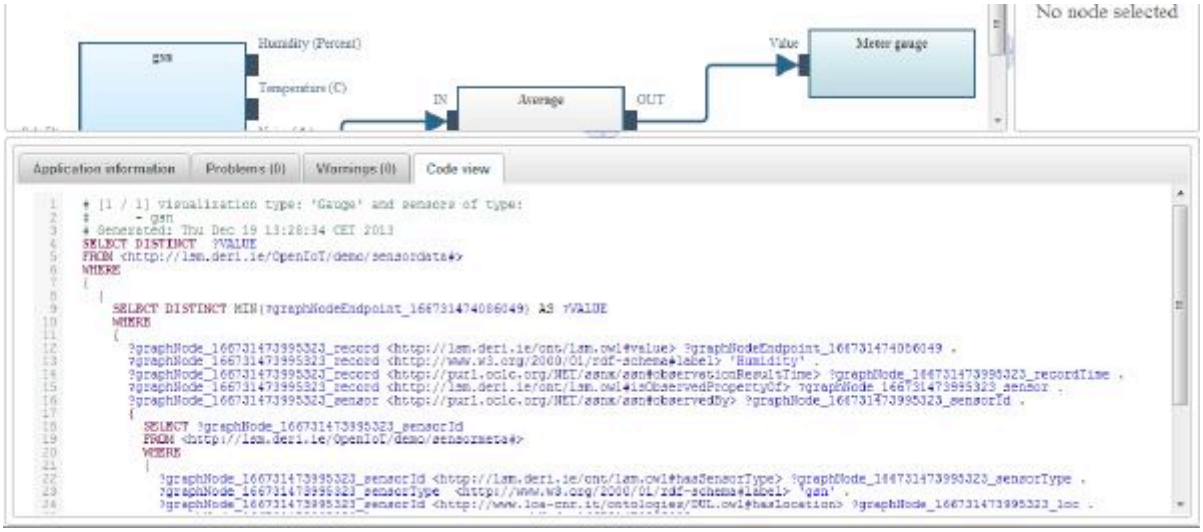


Figure 39: Canberra Average Noise SPARQL - Request Definition

We then used the Request Presentation tool for creating a Gauge to continually show the latest data received (**Figure 39**).

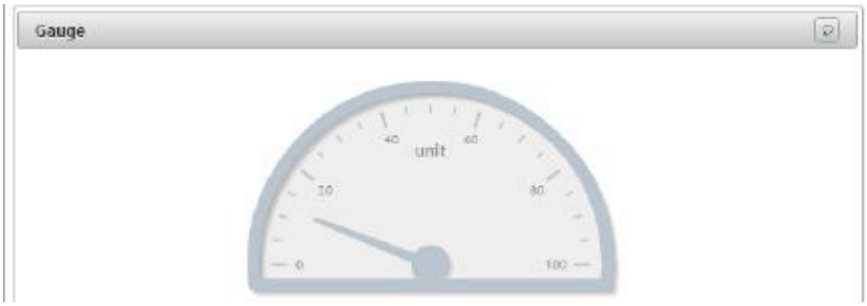


Figure 40: Canberra Average Noise Gauge - Request Presentation

Apart from Canberra, we also used Lausanne sensors in a similar way for Request Definition and Request Presentation.

OpenIoT Request Presentation cannot be used in the Silver Angel app, since it does not provide the visualisations foreseen (intensity maps). Instead, the Silver Angel app executes the SPARQL query itself and uses the received data in its Meet screen:

```
package com.example.testphonegap;

import org.apache.cordova.*;

public class MainActivity extends DroidGap {

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        super.setIntegerProperty("splashscreen", R.drawable.ldpi);
        super.setStringProperty("loadingDialog", "Loading Silver Angel ...");
        super.setIntegerProperty("loadUrlTimeoutValue", 50000);
        super.loadUrl("file:///android_asset/www/index.html",50000);


        String sparqlQueryString2 = "select * from <http://lsm.deri.ie/OpenIoT/demo/sensormeta#> where{<http://lsm.deri.ie/resource/

        Query query = QueryFactory.create(sparqlQueryString2);
        ARQ.getContext().setTrue(ARQ.useSAX);

        QueryExecution qexec = QueryExecutionFactory.sparqlService("http://lsm.deri.ie/sparql", query);
        //Retrieving the SPARQL Query results
        ResultSet results = qexec.execSelect();
        //Iterating over the SPARQL Query results
        while (results.hasNext()) {
            QuerySolution soln = results.nextSolution();

            System.out.println(soln.get("?o"));
        }

        qexec.close();
    }
```

Figure 41: SPARQL executed in the Silver Angel app

The example above shows the source code for executing a SPARQL query and logging the resulting data on the console. The intensity map widgets needed are being developed.

Silver Angel continues to evolve towards a prototype using the OpenIoT platform for Request Definition and Collect Data (for getting data streams within the app) and Request Presentation (for creating city administration dashboards).

4 INTELLIGENT MANUFACTURING – MATERIALS FLOW AND MANUFACTURING PERFORMANCE TRACEABILITY

4.1 Description

Nowadays plant managers install a large number of sensors in order to monitor their production processes. In particular, mid-sized plants are likely to comprise many hundreds of sensors of different types and for various purposes. These sensors generate a great volume of information, while they are associated with an always increasing rate of information. As a result significant information can be derived from these sensors regarding manufacturing performance (on the basis of appropriate KPIs (Key Performance Indicators)). In order to extract this information there is a need for efficient solutions for capturing, storing and processing sensor data is required. The monitoring of manufacturing performance could therefore greatly benefit from a system that would enable plant managers to dynamically select the information they want, through selecting information (in almost real-time) and using it to constructing KPIs dynamically. Such as solution would allow the dynamic construction of KPIs, beyond the fixed sets of KPIs that state-of-the-art manufacturing performance systems provide. Motivated by this need, the OpenIoT manufacturing use case provides the means for dynamically selecting sensor information, as well as for structuring this information on KPIs.

In terms of practical implementation and demonstration, the OpenIoT use case for intelligent manufacturing comprises a set of applications for monitoring and tracing the flow of materials in the scope of production processes for the paper and packaging industry. The applications enable manufacturers to dynamically define and visualize Key Performance Indicators (KPIs) associated with the manufacturing processes, by leveraging the semantic capabilities of the OpenIoT platform, and in particular its capabilities for dynamic discovery of sensors and of sensor data. The KPIs are defined and computed on the basis of a wide range of data streams, which are: (A) Collected by physical sensors residing in the shop floor, (B) Transformed into higher-level virtual sensors that comprise business events compliant to the data model of the EPC-IS standard, (C) Conveyed to the X-GSN sensor middleware, which ensures the semantic annotation of the data streams of the virtual sensors and their subsequent publication to the OpenIoT cloud.

While a large number of KPIs can be devised, the manufacturing use case will be demonstrated on the basis of a more specific scenario for the paper/packaging industry, which comprises three operations that might be executed in parallel, namely printing, die-cutting and gluing/folding. The above operations are executed sequentially and process each unit (of packages) produced. In this specific scenario, a set of (virtual) sensor streams are collected and published to the OpenIoT cloud.

The virtual sensors concern the calculation of:

- Operation rate for a specific process (measured in units/hour).
- Utilization metrics of the machines involved in the operation.
- Percentage of the operation that has been completed (in terms of the time required to complete the operation).
- Operation rate for the total process (measured in units/hour).

Once the information for these parameters is published in the OpenIoT cloud, manufacturers are able to dynamically discover and synthesise information from the above virtual sensors with a view to calculating (on-the-fly) KPIs (e.g., operation rates, utilization rates) for specific processes, while also being able to combine these metrics in order to calculate the parameters for composite (or even all) processes. Furthermore, interesting opportunities can arise in case the above virtual sensors are published to the OpenIoT cloud, not from just one, but rather from many (packaging/printing) manufacturers running the printing, die-cutting and gluing/folding processes. Overall, the semantic and dynamic discovery capabilities of the OpenIoT infrastructure are able to deliver on-the-fly manufacturing intelligence to end-users for monitoring, maintenance and planning purposes.

4.2 Scenario and implementation strategy

The implementation of the above scenario, involves:

- **Sensors**: A range of physical sensors, which are used to measure rates, quality information and traceability information associated with the printing, die-cutting and gluing/folding processes.
- **Traceability Kiosk and S-BOX Products**: SENSAP's ITK product is used to collect the data from the sensors and to process them in order to produce a set of virtual sensors (as data streams). The processing is performed by SENSAP's S-BOX product, which undertakes to transform the data into EPC-IS events, which comprise business context associated with the use of the sensors and the ITK in the scope of specific manufacturing processes (such as cutting, printing, folding etc.).
- **X-GSN middleware**: The implementation of the scenario foresees the interfacing of the EPC-IS data streams (produced within S-BOX) with the X-GSN sensor middleware. In-line with the OpenIoT architecture, the X-GSN middleware annotates the data streams in a way that makes them compliant to the OpenIoT ontology (which is an enhanced version of the W3C SSN ontology).
- **OpenIoT LSM Cloud Infrastructure**: The information (KPIs) associated with the manufacturing processes is streamed from the X-GSN middleware to the OpenIoT LSM cloud. This streaming process is a feature of the OpenIoT architecture.

- **KPIs Composition and Visualization Infrastructure**: Using the OpenIoT's request definition tool, one can dynamically define IoT services which (dynamically) calculate KPIs associated with manufacturing processes. The calculation of the KPIs is based on data available in the LSM cloud.

In following paragraphs we provide more detailed and specific information about each one of the above components.

4.2.1 Physical Sensors

The implementation of the manufacturing scenario hinges on the collection of information about the manufacturing process from the shop floor. This collection is empowered by the following physical sensors:

- An optical sensor for performance sensing, which sensor measures rate and quantity of produced items.
- A vision (image) sensor for quality control, which performs quality inspection of the produced items (e.g. checking colour patterns for detecting printing errors).
- RFID/barcode scanner for traceability, i.e. a sensor which collects traceability data (such as LOT numbers).

More detailed information about the above-listed sensors has been provided in the scope of deliverable D2.1. The sensors collect information about the materials used and the machines utilized in the scope of the execution of a production order. As illustrated in deliverable D2.1 additional information about materials and utilization can be collected based on optical diffusion sensors, weight sensors and temperature sensors, which however are not used in the demonstration scenario.

These physical sensors are part of SENSAP's ITK solution, since they are integral elements of the ITK component which is described in the following paragraph.

4.2.2 ITK and S-BOX Products

As already outlined the physical sensors outlined above are attached to the SENSAP™ INTEGRA ITK automation module, which has been designed and manufactured with an emphasis on the needs of the Printing and Carton-Converting industry. The ITK collects and processes information from the sensors towards quality and performance inspection, traceability data acquisition, as well as monitoring of material-consumption and resource-utilization. It encompasses all critical manufacturing operations, and auxiliary activities ranging from lamination and folio-sheeting to gluing/folding and shipping. Each ITK unit is directly connected on Printing, Die-Cutting, and Gluing/Folding machines, with a view to acquire and processes live data regarding machine performance, product quality, and raw-material consumption. An overview of an ITK unit is illustrated in **Figure 42**.

Figure 42: SENSAP's Integra Traceability Kiosk

In the scope of OpenIoT, an ITK unit is attached to each one of the machines performing the printing, die-cutting and gluing/folding operations of the scenario. In technical terms, the ITK collects sensors information and generates appropriate EPC-IS (Electronic Product Code Information Sharing) events i.e. events (information units) compliant to the EPC-IS standard [EPCIS]. EPC-IS events comprise business context, associated with the printing operations (e.g., such as the business process executed, the LOT number of the product, state changes to the product) and therefore have much richer semantics than the raw sensor data. The conversion of physical sensors data to EPC-IS compliant virtual sensors is controlled through a Graphical environment called PCS (Performance Control System). The PCS (**Figure 43**) allows manufacturers and/or integrators to define the EPC-IS virtual sensors on the basis of combining data from the physical sensors attached to the ITK. This offers significant flexibility in the definition and configuration of EPC-IS events. As part of the OpenIoT intelligent manufacturing use case integration, these events are structured as virtual sensors, which are providing information to the OpenIoT cloud.

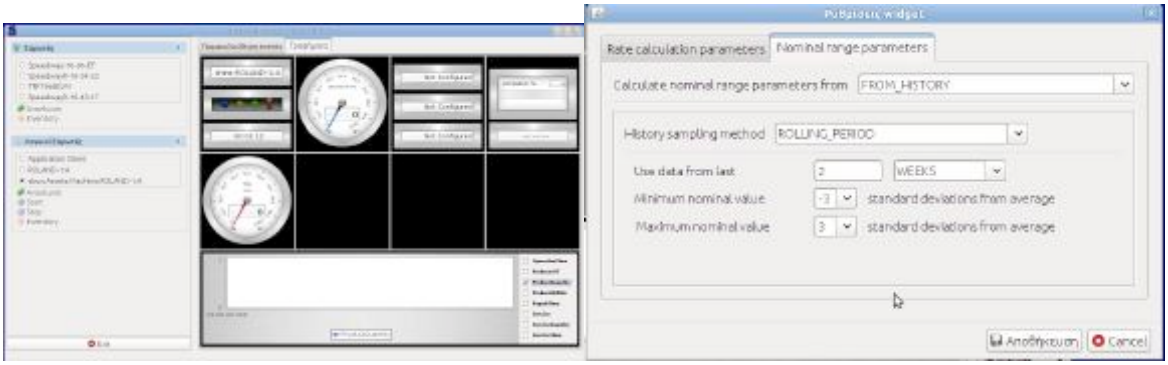


Figure 43: SENSAP's Performance Control System (Collects Information from Physical Sensors)

4.2.3 X-GSN and OpenIoT Linked Sensor Middleware (LSM)

The X-GSN middleware (i.e. the enhanced version of the GSN middleware that is contained in the OpenIoT platform) is used to stream (EPC-IS) virtual sensors to the OpenIoT cloud. To this end, the virtual sensors have been described/annotated as

GSN Virtual Sensors (i.e. based on the XML file of the Virtual Sensor) (see Table 1 and **Table 2** for two examples of virtual sensor descriptions). Accordingly, the X-GSN middleware converts the data to an OpenIoT compliant (i.e. compliant to the OpenIoT ontology) RDF format. This data is published to the OpenIoT cloud through the interface of the X-GSN to the LSM middleware. In this way, the use case takes advantage of the functionalities of the OpenIoT platform.

Table 1: Description of the Product Quantity Rate parameter of the ITK device as a Virtual Sensor of the X-GSN middleware

```
<virtual-sensor name="ITK B ProductQuantity Rate" priority="10">
    <processing-class>
        <class-name>gsn.vsensor.BridgeVirtualSensor</class-name>
        <init-params />
        <output-structure>
            <field name="rate" type="double"/>
        </output-structure>
    </processing-class>
    <description>This sensor calculates the rate of timeseries 'ProductQuantity' of
asset 'ITK-B'. The rate is expressed per 1 HOUR</description>
    <life-cycle pool-size="10" />
    <addressing>
        <predicate key="geographical">SBOX sensor</predicate>
        <predicate key="availableAfter">1386681444455</predicate>
        <predicate key="sensorClass">ITK-B Rate</predicate>
        <predicate key="latitude">37.95980984755405</predicate>
        <predicate key="longitude">23.694021356815032</predicate>
    </addressing>
    <storage history-size="5m" />
    <streams>
        <stream name="sbox105749178">
            <source alias="sboxsource105749178" sampling-rate="1" storage-size="1">
                <address wrapper="sbox">
                    <predicate key="host">192.168.0.249</predicate>
                    <predicate key="port">6006</predicate>
                    <predicate key="rate">30000</predicate>
                    <predicate key="sensorUID">105749178</predicate>
                </address>
                <query>SELECT * FROM wrapper</query>
            </source>
            <query>SELECT * FROM sboxsource105749178</query>
        </stream>
    </streams>
</virtual-sensor>
```


Table 2: Description of Machine State parameter of the ITK device as a Virtual Sensor of the X-GSN middleware

```
<virtual-sensor name="ITK B MachineState" priority="10">
    <processing-class>
        <class-name>gsn.vsensor.BridgeVirtualSensor</class-name>
        <init-params />
        <output-structure>
            <field name="red" type="TinyInt"/>
            <field name="yellow" type="TinyInt"/>
            <field name="green" type="TinyInt"/>
            <field name="blue" type="TinyInt"/>
            <field name="timeInState" type="BigInt"/>
        </output-structure>
    </processing-class>
    <description>This sensor reports the state of asset 'ITK-B' and the time that
the asset remains in current state</description>
    <life-cycle pool-size="10" />
    <addressing>
        <predicate key="geographical">SBOX sensor</predicate>
        <predicate key="availableAfter">1386681444392</predicate>
        <predicate key="sensorClass">ITK-B MachineState</predicate>
        <predicate key="latitude">37.95978902087909</predicate>
        <predicate key="longitude">23.693966310502454</predicate>
    </addressing>
    <storage history-size="5m" />
    <streams>
        <stream name="sbox293270425">
            <source alias="sboxsource293270425" sampling-rate="1" storage-size="1">
                <address wrapper="sbox">
                    <predicate key="host">192.168.0.249</predicate>
                    <predicate key="port">6006</predicate>
                    <predicate key="rate">30000</predicate>
                    <predicate key="sensorUID">293270425</predicate>
                </address>
                <query>SELECT * FROM wrapper</query>
            </source>
            <query>SELECT * FROM sboxsource293270425</query>
        </stream>
    </streams>
```


4.2.4 KPIs Composition and Visualization

Following the publication of the EPCIS compliant virtual sensors to the OpenIoT LSM infrastructure, the request definition tool can be used to construct KPIs associated with materials flows and manufacturing performance. Different (SPQRQL) queries are constructed based on different graphs towards calculating and visualizing manufacturing KPIs such as:

- Operation rate for a selected process (measured in units/hour) i.e. printing, folding, die-cutting. During the definition of the query/service, the target process can be specified.
- Utilization metrics (%) of one or more of the machines involved in the monitored manufacturing operations. The target machine or machines is a parameter of the query.
- The percentage of the operation that has been completed (in terms of the time required to complete the operation). The operation can be a parameter of the service that could be specified by the user.
- The calculation of (aggregate) operation rates for the combination of processes (measured in units/hour). The target processes is a parameter that can be specified by the user.

Note that the above queries/services are indicative in order to showcase the concept of dynamic selection, composition and calculation of KPIs. The relevant IoT services can deliver significant value in case where data from multiple printing plans are becoming available in the OpenIoT cloud.

4.3 Current status and demonstration

The current status of the use case has completed the steps up to the interfacing of the virtual sensors to the X-GSN middleware, towards creating and publishing OpenIoT compliant semantic data sets about the use case. In particular, the virtual sensors specified and configured through the PCS are currently deployed within the X-GSN middleware. Furthermore, SENSAP has experimented with the request definition tools towards creating sample models associated with the calculation of operation rates for various processes, as well as with the calculation of utilization metrics. In particular, graphs associated with these metrics have been produced, but not deployed over the released version of the OpenIoT platform. Note that **Figure 44** depicts a high-level mapping of the implemented components of the use case to OpenIoT's architecture components. In particular, components with green colour indicate implemented/supported functionalities as part of the available version of the use case. On the other hand, components with yellow colour will be implemented/integrated as part of future releases of the use case implementation.

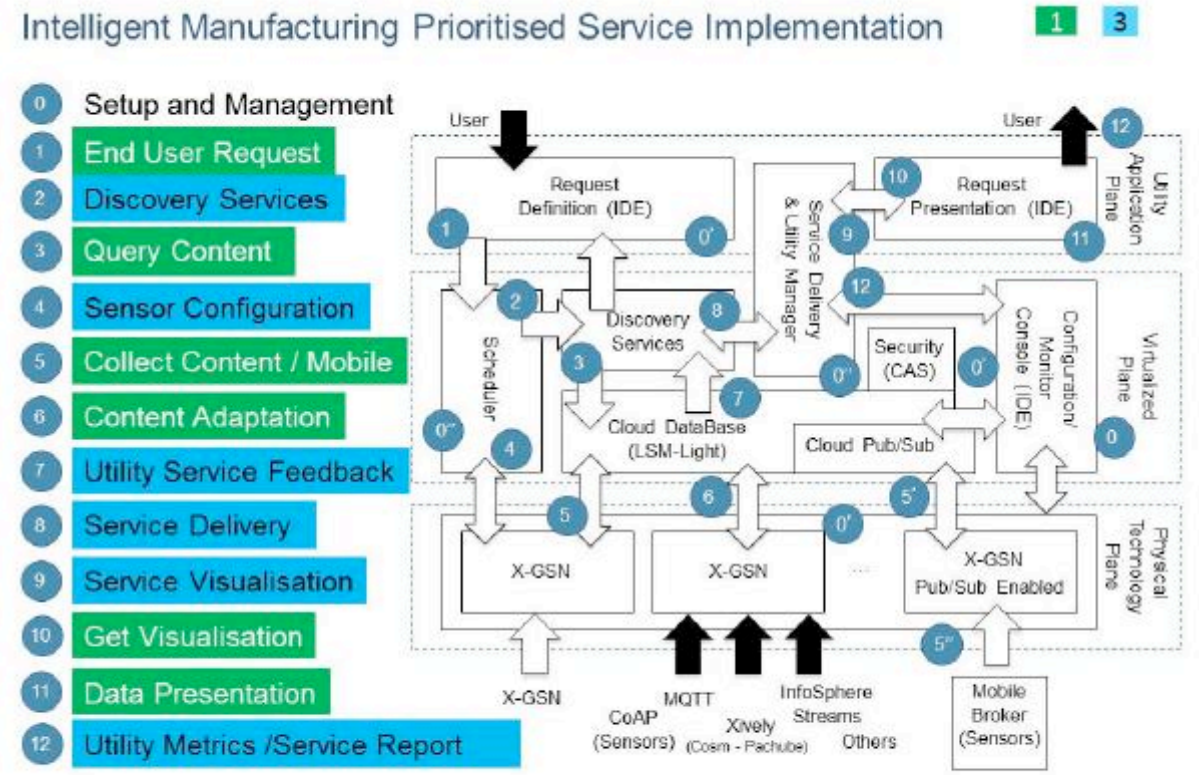


Figure 44: Mapping of the Current Implementation of the Manufacturing Use Case to OpenIoT architecture components (Green: Available Functionality Blue: Future/Planned Functionality)

A video demonstrating the main aspects of use case is available on YouTube at: http://www.youtube.com/watch?v=yrhyvx0znOY.

4.4 Future work

Towards the next version of the use case, a complete integration with the OpenIoT platform will be realized. This involves the following steps:

- Completion of the deployment of the OpenIoT platform, i.e. deployment of all its modules as an infrastructure for the use case.
- Interfacing of the available X-GSN infrastructure (comprising the EPC-IS events as virtual sensors) to the OpenIoT LSM cloud. This will enable the publication of manufacturing performance and materials flow traceability data in the cloud.
- Deployment of the models/workflows that have been created with the request presentation tool, over the OpenIoT infrastructure. This will also lead to visualization of operational rates and utilization rates to appropriate OpenIoT mash-ups.

Based on these steps, a first fully fledged demonstration will be completed. Following the completed of the fully integrated demonstration, we will also investigate and experiment the enrichment of the manufacturing data in two directions:

- Publication of additional KPIs, such as: (A) Defect rates per operation and/or per process, (B) State for the individual operations and the whole process (idle, stopped, in setup, in production), (C) Production rate per product type, (D) Defect rate per material type.
- Collection and/or simulation of data from more than one plants.

The enrichment of the data outlined above would enable the demonstration of more interesting scenarios.

SENSAP will attempt to install and deploy the system in a real manufacturing plant based on liaison collaboration with one of its clients (i.e. a client that has already deployed ITK systems).

5 DIGITAL AGRICULTURE - PHENONET

5.1 Description

'Phenonet' describes the network of wireless sensor nodes collecting information over a field of experimental crops. The term “Phenomics” describes the study of how the genetic makeup of an organism determines its appearance, function, growth and performance. Plant phenomics is a cross-disciplinary approach, studying the connection from cell to leaf to whole plant and from crop to canopy (CSIROa 2013).

Analysing the size, growth and performance of plants in a greenhouse or field site can be time-consuming and laborious. More specifically, when a field site is located in a remote area, it becomes quite expensive to send people out to the field. The ability to collect this information from remote locations and send it back to the laboratory in real time is an invaluable tool for plant scientists (CSIROa 2013, CSIROb 2013).

CSIRO has developed smart wireless sensors nodes that work autonomously and independently cooperating with each other to set up an ad hoc network to record environmental conditions and wirelessly transfer data to a data store.

5.1.1 Current Architecture and Data Model

5.1.1.1 Architecture

The Phenonet project is supported by a production system with commercial quality grade software and unit tests developed for researchers in CSIRO and government organizations in Australia. Phenonet platform is currently being used in daily basis and enjoys high level of uptime and very stable code. The high-level architecture of Phenonet project is depicted in Figure 45 consists of five stages.

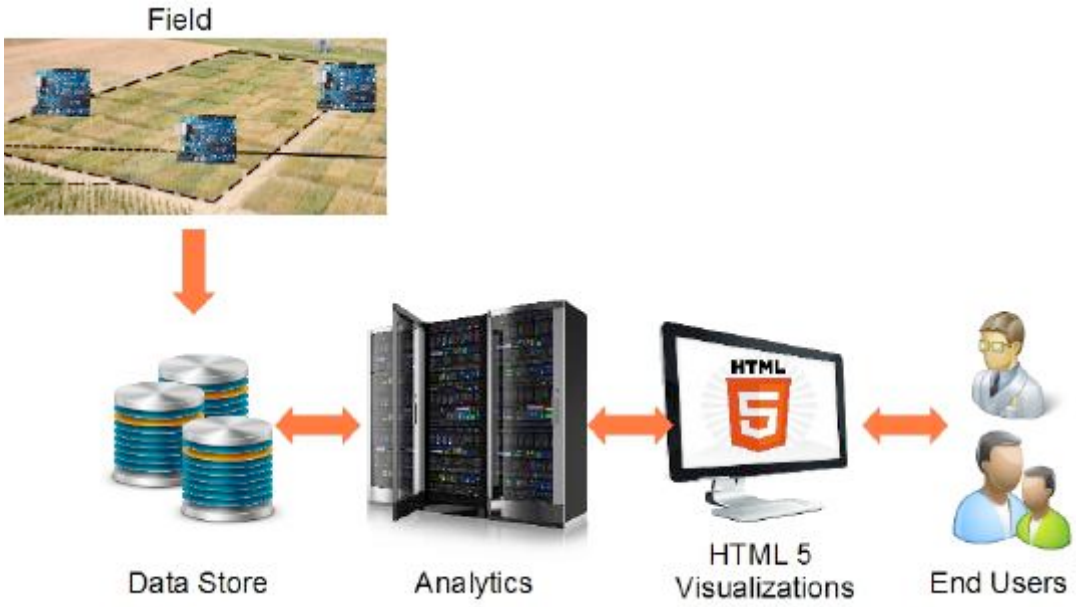


Figure 45: Phenonet Architecture

- **Field**: The field is an experimental plot comprising different types of crops varieties. Wireless sensors are installed in the experimental plots that measure various environmental features such as soil temperature, crop canopy temperature, humidity, wind speed etc. Using this information, the crops growth, performance, size, etc. are continuous sensed/computed in real-time.
- **Data Store**: Data Storage highlights the need to have all captured data and information about the data (metadata), to be stored in a safe location. At storage state, we are targeting both sensor measurements and metadata information. Examples of metadata information include; sensor types, serial numbers, MAC address, experimental treatment, crop sowing date, genotype, replicate number etc. Each sensor stream is identified using a globally unique identifier (GUID). This layer in current Phenonet relies on python scripts to upload data into the system.
- **Data Analysis**: Is the brain behind all the calculations, data modelling, data cleansing and linear aggregation models used by Phenonet project. This component directly contacts Data Store layer when it requires data from a particular stream. Internally, Data Analysis component also performs extensive caching and applies proprietary algorithms and mechanisms to ensure a highly responsive interaction with the system is maintained at all times. Data Analysis component is accessible through HTTP RESTful API. The response to any request received by this component is in format of JSON object. This layer is developed in Scala.
- **HTML5 Visualization**: This component is responsible to generate RESTful network requests and send them to data analysis component. The response is then rendered by the frontend and appropriate visualization components. This

layer is written in CoffeeScript and uses HTML5 for rendering and visualization.

- **End user**: Ranges from a plant biologist to a farmer. The system also provides tools and mechanisms to share data analysis and visualizations with other group of users.

5.1.1.2 Data Model

The Phenonet data model is depicted in Figure 46.

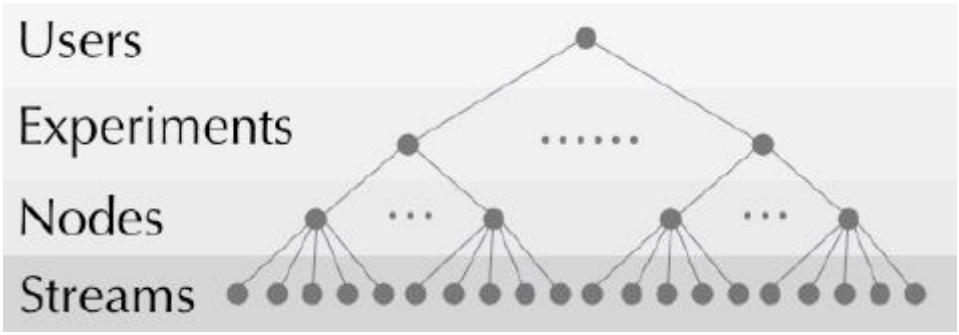


Figure 46: Phenonet Data Model

In Phenonet, a user is a logical entity (e.g. a project or a research group), which owns a group of experiments. An experiment has only one owner. Each experiment is a group of nodes and each node belongs to a single experiment. A node can also have its location associated with it, such as latitude and longitude values. A node itself is a group of streams and a stream is a series of timestamp and real number pairs with a unit of measurement. Metadata can be attached at every hierarchical layer. In general the following polices are enforced on the data model.

- Any user can have zero or more experiments
- Any experiment can have zero or more nodes
- Any node can have zero or more streams. Each node can also have latitude, longitude and altitude values.
- Any stream is a set of (timestamp, value) pairs. Each stream has one unit of measurement.

The mapping of a typical field experiment is illustrated in Figure 47. In this example, the experiment is an ordered arrangement of 2 m wide by 6 m length plots. The plots are subdivided into experimental units and are mapped to the node level in Phenonet. On some of these experimental units, measurements of soil moisture are made at multiple depths. A measurement of soil moisture at a particular depth is mapped to the stream level, associated to a node. In summary, the stream maps to the physical or virtual sensor that monitors a phenomenon while nodes and experiments are used for grouping at different levels.

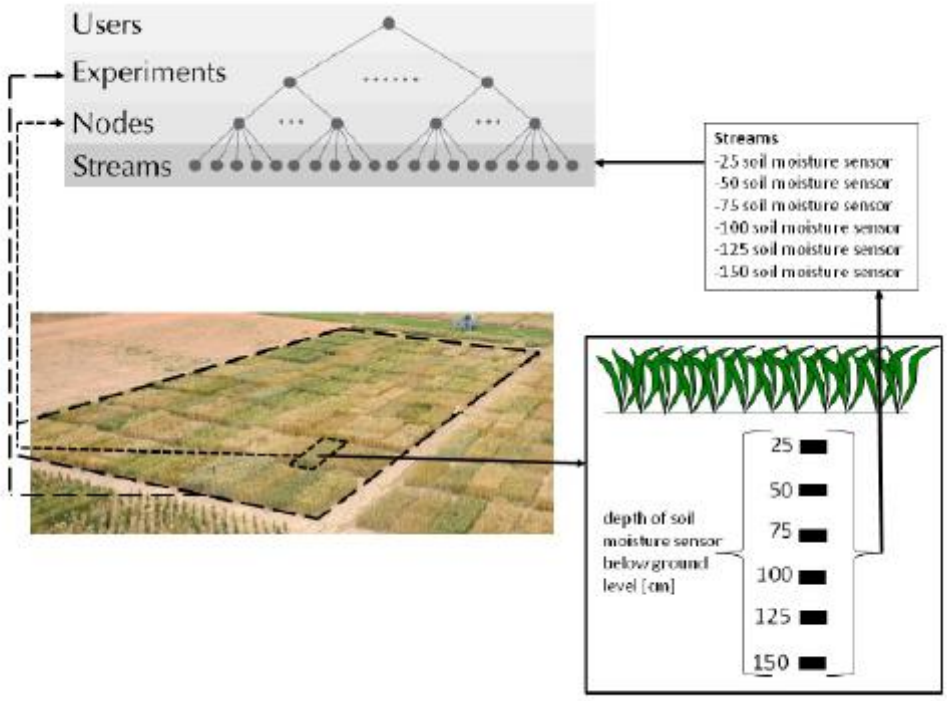


Figure 47: A typical field experiment mapped to the Phenonet data model

The metadata associated to each of the levels is critical for providing contextual information. In the above example, for the experiment level, metadata could include information such as the year when the experiment was run; the date the experiment was sown; description about the objectives of the experiment and even descriptions about the experimental site like e.g. soil type. At the node level the most important metadata fields are the genotype and the relative location of the experimental unit within the experimental plot (in most cases for this application a row/column notation is used). Treatments applied to individual experimental units can also be appended as metadata at the node level. At the stream level, in this example the depth, the sensor type and the sensor serial number are the most important metadata fields, while sensor information like the date of calibration or settings of the sensor can also be critical.

5.2 Scenario and implementation strategy

5.2.1 Implementation Strategy

In order to show flexibility and application of OpenIoT Middleware, Phenonet project team would work closely with OpenIoT partner to connect Phenonet project to OpenIoT and vice versa. This integration practice offers benefits for both parties. In the case of Phenonet project, it offers various integration points, which can be utilized to add new sensor types and new types of data analysis tools. In the case OpenIoT, this integration is yet another proof of concept that commercial grade 3rd party solution can be ported/migrated to use OpenIoT middleware, and by doing so, 3rd party software can take advantage of the features offered by the OpenIoT platform such as semantically rich sensor annotations, sensor discovery etc.

Mapping of prioritized implementation of Phenonet scenario to OpenIoT services is depicted in
Figure 48. Green colour denotes services which are implemented first, then come services coloured in yellow and then come services coloured in blue.

As a proof of concept, we see Phenonet architecture to be extended as depicted in Figure 48. Phenonet will take advantage of the following core services and tools provided by the OpenIoT platform.

1. **xGSN**: XGSN will be used as the sensor streaming middleware. xGSN wrappers will be developed to interface with the Phenonet data store to obtain real-time access to Phenonet sensor data. xGSN will also be responsible to annotate incoming sensor data streams from the Data Store/Field.
2. **Scheduler and Service Delivery and Utility Manager (SDUM):** These core services will be used to compose and deploy a Phenonet experiment on the OpenIoT platform. A service composition in OpenIoT will be mapped to Phenonet experiments that comprise a set of sensors with a particular analytic operation such as average, sum.
3. **Cloud Data Store and Discovery Service (LSM-Lite):** The data from existing data store will be pushed into LSM along with sensor annotations allowing discovery of sensor data.
4. **Request Definition and Presentation:** The request definition and presentation tools will be used to design and deploy an experiment with the help of the discovery, scheduler and SDUM core services.

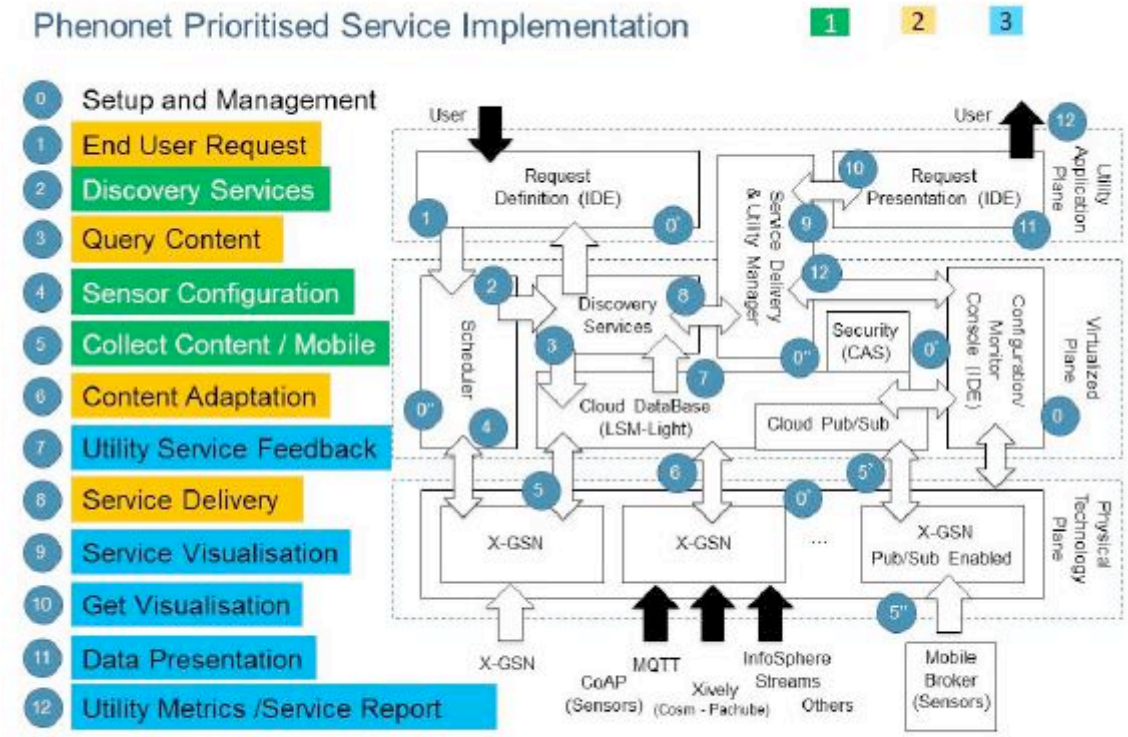


Figure 48: Mapping of Phenonet to OpenIoT services

Within the Phenonet-OpenIoT context, an experiment will be composed by the end-user (a farmer or a scientist) by discovering relevant sensor data required for the experiment. For example, to compose an experiment as depicted in Figure 49, an end-user will search for soil moisture sensors at different depth and compose an experiment/service for each of the discovered sensor. The location of the sensor will map to the node location of existing Phenonet application. The metadata for a given experiment will be added to the description of the experiment when composing the experiment using the Request Definition tool. Similarly, HTML5 visualizations of Phenonet will be replaced with OpenIoT Request Presentation tools that allow users to visualize the experiment's outcomes.

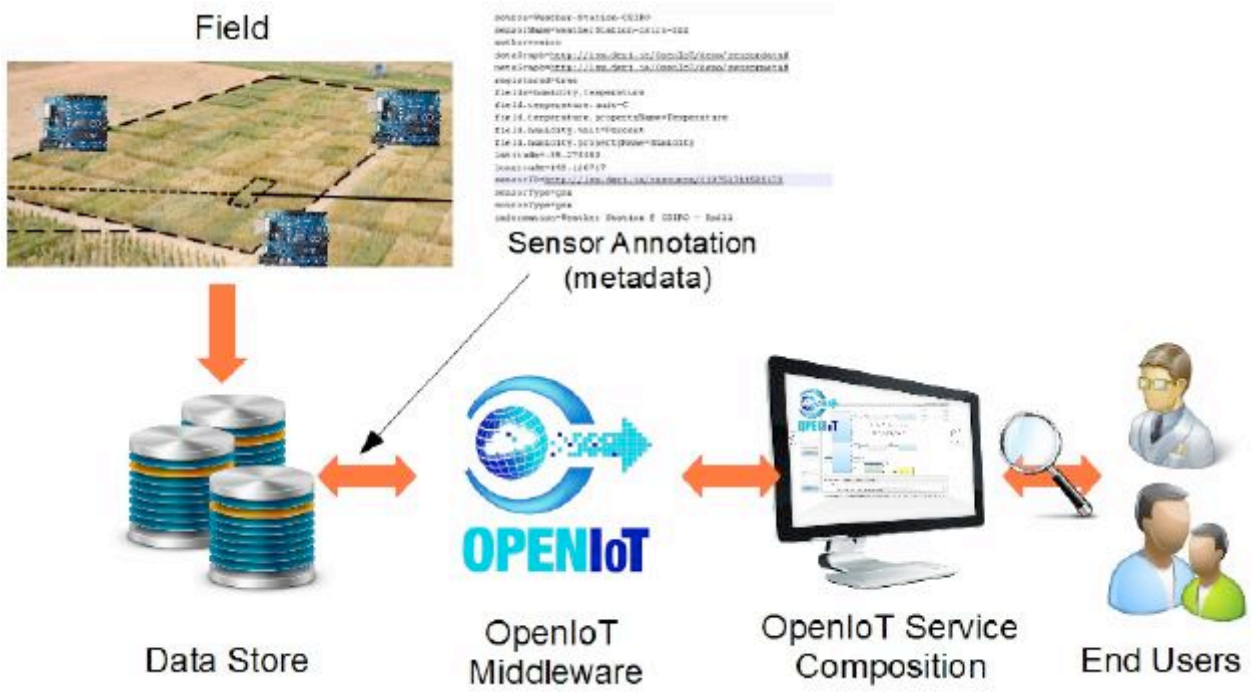


Figure 49: Proposed Phenonet Implementation Architecture on OpenIoT

5.2.1.1 Implementation Scenarios

The dataset that will be used to implement the Phenonet proof-of-concept application on the OpenIoT platform will be made available to OpenIoT consortium and public at large. Meanwhile, the sample data set can be found in Appendix 1.

The following section will describe in details the purpose of these experiments, the types of sensors used in these experiments and their characteristics and the benefits delivered by these experiments. This information was recorded from plant biologist conducting the experiments. The experiment under focus is *Kirkegaard and Danish*.

5.2.1.1.1 Experiment Description

- **Kirkegaard and Danish Experiment**: Kirkegaard and Danish is an experiment for evaluating effect of sheep grazing on crop re-growth by looking at root activity, water use, crop growth rate, and crop yield. In this experiment soil moisture sensors have been installed at multiple depths from 10 cm to 2 m below the soil surface. This enables the end-user to track the extraction of water from the soil by the roots throughout the crop growing season. We can then obtain an indirect measurement of root activity.
- **Experimental Units (Nodes):** The naming convention at the node level is as follows:
    - 'Genotype_Column#Row#_Treatment'
        - Genotype - the particular genotype / variety e.g. Revenue.
        - Treatment:
            - Grazed: Animals are allowed to eat the plant leaves to control the plant growth. After sometime, the animals are moved off the crop allowing the crops to grow
            - Ungrazed: In the case of Ungrazed, the focus is the crop, no animals. no animal to be fed there.
            - High N : High in nitrogen
        - Where:
            - Column - field coordinate in direction of tractor travel when sowing the crop. Sometimes described as "how wide the trial is". Used at the node level.
            - Row - field coordinate at right angles to column. Sometimes described as "how deep the trial goes". AGT use 'Prange' and commonly have 12x Prange in a trial. Used at the node level.
    - **Naming E.g.** 'EGA Gregory_C02R03'
        - Genotype = EGA Gregory
        - Column = 02
        - Row = 03
- **Sensor Nodes (Streams):** GBHeavy is the hardware name of the sensor used and is depicted in Figure 50. GBHeavy measure the soil water status at the particular depth of interest. More technical details about this sensor is available here[10]:
    - **Naming E.g**.' GBHeavy100'
        - installed at 100cm depth

[10] http://mea.com.au/upload/NEW/PRODUCT_Brochures/B03_Gypsum_Blocks_Web_0_1.pdf

Specifications for GB Heavy	
Sensor Type	Gypsum block
Measurement Range	50 to 500 kPa
Output Signal	AC resistance
Specifications for GB Light	
Sensor Type	Granular matrix
Measurement Range	0 to 200 kPa
Output Signal	AC resistance

Figure 50: Soil Moisture Sensor

5.2.1.1.2 Experiment Purpose

The purpose of the experiment is to evaluate the effect of sheep grazing on crop re-growth by looking at root activity, water use, crop growth rate, and crop yield. The information about crop growth obtained in real time can effectively help the researchers to provide estimates on the potential yield of a variety.

The experiment tries to compare trade-offs between grazed and un-grazed setup for a particular variety of crop. This experiment is supposed to last for 9 months. The animals come in the first 6 months and after that they are removed from the site. To compute the root activity and water usage, a soil moisture sensor is deployed. Sensors at one depth level don't tell the entire story. We need multiple depths to see the behaviour of the root system and the water available to the crop at any particular time.

- *Efficiently and effectively manage the water resource:* The soil profile information could be used in a farming production system as follows. If there is not enough water left in the soil profile, the farmer may decide to move the live stocks into the site (crop farm). The movement of live stock into the site will cause the water usage to be reduced as live stock feed on the leaves. This will delay the use of water use until the vital grain filling period. When soil moisture is high, the farmer may decide to move the live stocks off the site so that the plants can consume the water effectively and produce higher yields.
- *Efficiently and effectively administer the timing of using fertilizers*: The farmer may also use the soil profile information obtained in real time to arrive at a decision on when to apply nitrogen to the soil to aid crop growth. Nitrogen is

applied generally when soil moisture is high, so it supports the growth of the crop, otherwise nitrogen is wasted.

- *Dual Purpose Cropping System*: Increase crop yield by efficiently and effectively utilising the resources (water and fertilizer) while also allowing live stock to feed on healthy crop leaves enriching live stock growth.

The above setup is not only for experimenting but farmers benefit from it by using the data to plan nitrogen fertilizer, timing of moving animals (animals/live stock eat leaves of the plants, which is, but eating the flower which yields the grains is not good) in and out before they start eating the reproductive part of the plants (called the flower) and planning water resources.

The key focus of the experiments is to increase crop yield and produce high quality harvest. Observing and understanding different types of crop performance and growth under varying conditions like soil, grazed/un-grazed, water and nitrogen content can greatly help in increasing the quality and quantity of crop yield.

5.3 Current status and demonstration – Stage 1

5.3.1 Demonstration

The OpenIoT-Phenonet implementation (Stage 1) will focus on migrating and demonstrating the following OpenIoT features for the Phenonet Use case presented earlier.

1. **Sensor Configuration**: The sensor configuration task will involve the following activities
    1. Extending the OpenIoT ontology to describe Phenonet Sensors more specifically the GBHeavy Sensor platform
    2. Registering the extended ontology with the cloud store (LSM)
    3. Developing a metadata description for the sensor that will be used by xGSN to push data into cloud store (LSM)

2. **Collect Content**: The collect content task will involve the development of xGSN wrappers that will allow xGSN to fetch the data from the Phenonet data store. The current Phenonet system provides restful API to get raw and aggregated data streams as follows:
    1. **Phenonet Data Access API**

Table 3: API Specification - Overview

Resource	Method	Description
/data_download	POST	Download raw or aggregated sensor data

2. POST /data_download

This request is for downloading raw or aggregated sensor data from one or more streams. The aggregation level is specified per request.

Table 4: Phenonet API Specification- Detailed

Parameter	Required	Default	Description	Format
level	No	raw	Aggregation level	level is text and can be set to one of the following values: raw, 1-minute, 5-minute, 15-minute, 1-hour, 3-hour, 6-hour, 1-day, 1-month, 1-year
sd	No		Start Date (inclusive)	Date in iso format, e.g., 2012-01-30 for 30th of Jan, 2012
ed	No		End Date (inclusive)	Date in iso format, e.g., 2012-12-20 for 20th of Dec, 2012
st	No		Start Time (inclusive)	For Specifying Time of Day (ToD) in raw, 1-minute, 5-minute, 15-minute, 1-hour aggregation levels. For format is 01:01:00 [00-23]:[00-59]:[0-59]
ed	No		End Time (inclusive)	For Specifying Time of Day (ToD) in raw, 1-minute, 5-minute, 15-minute, 1-hour aggregation levels. The format is 02:09:59 [00-23]:[00-59]:[0-59]
sid	No		stream id(s)	sid or ["sid1","sid2","sid3",...]

The xGSN wrapper will consume the restful API and obtain the data for appropriate streams.

3. **Data Discovery**

   Once data is pushed by xGSN, the data will be discoverable using the existing OpenIoT tools namely the scheduler and Request Definition Interface. The Request Definition interface will be used to demonstrate the discovery of sensor data along with simple proof-of-concept experiment/service composition.

4. **OpenIoT Integrated Platform Setup**

   The integrated platform setup involves setting up and deploying the OpenIoT services on a local infrastructure at CSIRO. This task includes obtaining the latest version of the source from GitHub, compiling and deploying them without any bugs. This process has been ongoing since October and CSIRO currently has a successfully running integrated OpenIoT platform that uses DERI's LSM cloud store infrastructure.

5.3.2 Current Status

The following components of OpenIoT have been successfully installed, configured and integrated at CSIRO

1. Scheduler and SDUM
2. LSM-Lite Client (partially working)
3. Connection to LSM on DERI server
4. Request Definition and presentation UI
5. Ide.core UI
6. xGSN (with test wrapper for Netatmo Weather Station)

We are now working on Task 1, 2 and 3 with the OpenIoT second review as the target date for Stage 1 OpenIoT-Phenonet demonstrations. The Gantt chart in Figure 51 below presents the current and proposed timeframes for proof of concept demonstration completion.

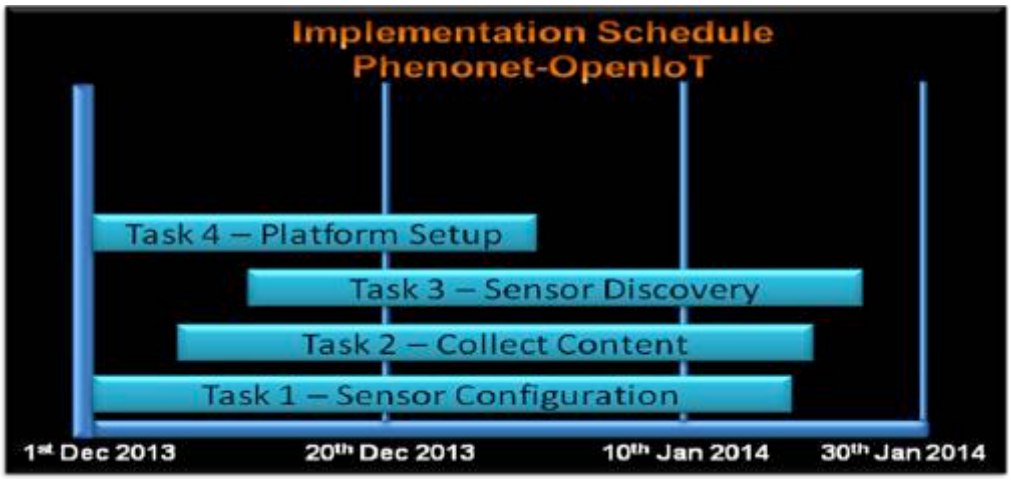


Figure 51: OpenIoT-Phenonet Implementation – Stage 1 Schedule

5.3.3 Current Issues

The LSM on DERI has been successfully tested. As of this document writing, the LSM-Server and LSM-Client version was received on 6th December 2013. Given the priority of the other three tasks to be completed by January 2014, we will not use a local LSM server version for the Stage 1 demonstration.

The current OpenIoT platform only allows the creation of sensor types called gsn. This is a major issue and limitation as sensors like soil moisture, weather, etc. cannot be discovered. This issue has a reported fix time of 15th December. Depending on successfully fix and no further issues, we hope we can incorporate this into the Stage 1 demo.

5.4 Future work

The future work with the OpenIoT-Phenonet integration will include implementing and demonstrating the Phenonet platform over the entire range of OpenIoT tools and services for the scenarios mentioned in Section 5.2.1.1.

The plan is to implement and demonstrate the Phenonet use case previously described over the OpenIoT platform. The following services will be demonstrated by the respective deadlines.

- OpenIoT-Phenonet- Stage 2 Implementation and Demonstration (Deadline May 2014)
    - All features of Stage 1 implementation
    - End User Request
    - Query Content
    - Content Adaptation
    - Service Delivery
- OpenIoT-Phenonet- Stage 3 Implementation and Demonstration (Deadline Nov 2014)
    - All features of Stage 2 implementation
    - Utility service feedback
    - Service Visualization
    - Data Presentation
    - Utility Metrics
    - Security

5.5 Section Acknowledgement

We would like to thank Dr. David Deery, Dr. Jose Jimenez-Berni and their team at Plant Industry, CSIRO for their contributions to this document.

6 CONCLUSIONS

This deliverable reports on the first release of Proof-of-Concept Validating Applications (a), on partner experience and lessons while implementing diverse use cases on the basis of OpenIoT software platform.

The document summarises the OpenIoT software platform, core middleware and components that are used to develop OpenIoT use cases and map the use cases scenarios onto OpenIoT software platform.

The implemented use cases include: (a) Smart Cities – Campus Guide; (b) Smart Cities – Silver Angel; (c) Smart Industries – Intelligent manufacturing – Materials Flow and Manufacturing Performance Traceability; (d) Smart Industries – Digital Agriculture – Phenonet. All the reported use cases describe how the OpenIoT use cases are/can/will be mapped to OpenIoT platform services according to 3-stage priorities. The use cases include description of test data sets, whether they're already available for public use, or what are the plans to release them for public use via the OpenIoT Github. The availability of such data set will allow the provision of on-line demos through the OpenIoT open source project, which can be critical for the development of the OpenIoT developers and contributors community, but also for the take up of the project as a whole. The use case sections also describe experience, issues, difficulties, lessons while developing use cases on the basis of the OpenIoT software platform.

It is worth mentioning that the use cases are at different stages of maturity and have implemented different sets of services as described in the document. Furthermore, the three use cases feature different technical approaches and involved diverse sensors and functionalities. This is to some extend purposeful in order to explore and validate different aspects and modules of OpenIoT. It is also intended to showcases that OpenIoT is not an «all or nothing» proposition, given that different integrators are likely to use appropriate subsets of the OpenIoT functionalities. However, all the OpenIoT use cases will implement full sets of OpenIoT services and will fully mapped onto the OpenIoT software platform as planned.

7 REFERENCES

[EPCIS] EPCglobal, EPC Information Services (EPCIS) Specification (version 1.0.1), available at: http://www.gs1.org/gsmp/kc/epcglobal/epcis (retrieved November 2013)

[CSIROa] CSIRO, 2013 Phenonet: wireless sensors in agriculture. Available from: http://www.csiro.au/Outcomes/ICT-and-Services/National-Challenges/Wireless-sensors-in-agriculture.aspx. Accessed on Dec 2013

[CSIROb] CSIRO, 2013 The High Resolution Plant Phenomics Centre. Available from: http://www.csiro.au/Outcomes/Food-and-Agriculture/HRPPC/Sensors-in-the-field.aspx. Accessed on Dec 2013

[Le-Phouc 2011] D. Le-Phuoc, M. Dao-Tran, J.X. Parreira, M. Hauswirth, A native and adaptive approach for unified processing of linked streams and linked data, in: ISWC, 2011, pp. 370–388

APPENDIX I – SMALL SAMPLE DATASET FOR DIGITAL AGRICULTURE – PHENONET USE CASE

Timestamp	Soil Moisture - Value
12 Dec 2013 - 23:00:00	-935.33
12 Dec 2013 - 21:00:00	-935.33
12 Dec 2013 - 19:00:00	-931.13
12 Dec 2013 - 17:00:00	-922.86
12 Dec 2013 - 15:00:00	-910.78
12 Dec 2013 - 13:00:00	-899.08
12 Dec 2013 - 11:00:00	-880.38
12 Dec 2013 - 09:00:00	-873.16
12 Dec 2013 - 07:00:00	-880.38
12 Dec 2013 - 05:00:00	-887.74
12 Dec 2013 - 03:00:00	-895.26
12 Dec 2013 - 01:00:00	-906.84
11 Dec 2013 - 23:00:00	-914.76
11 Dec 2013 - 21:00:00	-914.76
11 Dec 2013 - 19:00:00	-910.78
11 Dec 2013 - 17:00:00	-902.94
11 Dec 2013 - 15:00:00	-891.49
11 Dec 2013 - 13:00:00	-880.38
11 Dec 2013 - 11:00:00	-866.08
11 Dec 2013 - 09:00:00	-855.72
11 Dec 2013 - 07:00:00	-859.14
11 Dec 2013 - 05:00:00	-869.6
11 Dec 2013 - 03:00:00	-876.75
11 Dec 2013 - 01:00:00	-887.74
10 Dec 2013 - 23:00:00	0
10 Dec 2013 - 21:00:00	-899.08
10 Dec 2013 - 19:00:00	-895.26
10 Dec 2013 - 17:00:00	-884.04
10 Dec 2013 - 15:00:00	-876.75
10 Dec 2013 - 13:00:00	-866.08
10 Dec 2013 - 11:00:00	-855.72
10 Dec 2013 - 09:00:00	-845.66
10 Dec 2013 - 07:00:00	-845.66
10 Dec 2013 - 05:00:00	-855.72
10 Dec 2013 - 03:00:00	-862.59
10 Dec 2013 - 01:00:00	-866.08
09 Dec 2013 - 23:00:00	-869.6
09 Dec 2013 - 21:00:00	-873.16
09 Dec 2013 - 19:00:00	-869.6
09 Dec 2013 - 17:00:00	-859.14
09 Dec 2013 - 15:00:00	-842.37
09 Dec 2013 - 13:00:00	-820.17
09 Dec 2013 - 11:00:00	-796.47
09 Dec 2013 - 09:00:00	-785.22
09 Dec 2013 - 07:00:00	-790.8
09 Dec 2013 - 05:00:00	-802.24
09 Dec 2013 - 03:00:00	-814.09
09 Dec 2013 - 01:00:00	-826.37
08 Dec 2013 - 23:00:00	-835.88
08 Dec 2013 - 21:00:00	-845.66
08 Dec 2013 - 19:00:00	-842.37
08 Dec 2013 - 17:00:00	-829.51
08 Dec 2013 - 15:00:00	-811.08
08 Dec 2013 - 13:00:00	-790.8
08 Dec 2013 - 11:00:00	-771.69
08 Dec 2013 - 09:00:00	-758.71
08 Dec 2013 - 07:00:00	-769.05
08 Dec 2013 - 05:00:00	-779.74
08 Dec 2013 - 03:00:00	-790.8
08 Dec 2013 - 01:00:00	-802.24
07 Dec 2013 - 23:00:00	-811.08

07 Dec 2013 - 21:00:00	-817.11
07 Dec 2013 - 19:00:00	-811.08
07 Dec 2013 - 17:00:00	-799.34
07 Dec 2013 - 15:00:00	-782.47
07 Dec 2013 - 13:00:00	-763.84
07 Dec 2013 - 11:00:00	-743.81
07 Dec 2013 - 09:00:00	-731.91
07 Dec 2013 - 07:00:00	-741.39
07 Dec 2013 - 05:00:00	-751.17
07 Dec 2013 - 03:00:00	-761.26
07 Dec 2013 - 01:00:00	-774.35
06 Dec 2013 - 23:00:00	0
06 Dec 2013 - 21:00:00	-790.8
06 Dec 2013 - 19:00:00	-788
06 Dec 2013 - 17:00:00	-774.35
06 Dec 2013 - 15:00:00	-758.71
06 Dec 2013 - 13:00:00	-738.99
06 Dec 2013 - 11:00:00	-722.72
06 Dec 2013 - 09:00:00	-711.6
06 Dec 2013 - 07:00:00	-720.46
06 Dec 2013 - 05:00:00	-729.59
06 Dec 2013 - 03:00:00	-738.99
06 Dec 2013 - 01:00:00	-751.17
05 Dec 2013 - 23:00:00	-763.84
05 Dec 2013 - 21:00:00	-774.35
05 Dec 2013 - 19:00:00	-779.74
05 Dec 2013 - 17:00:00	-771.69
05 Dec 2013 - 15:00:00	-761.26
05 Dec 2013 - 13:00:00	-763.84
05 Dec 2013 - 11:00:00	-763.84
05 Dec 2013 - 09:00:00	-758.71
05 Dec 2013 - 07:00:00	-774.35
05 Dec 2013 - 05:00:00	-790.8
05 Dec 2013 - 03:00:00	-808.11
05 Dec 2013 - 01:00:00	-823.25

04 Dec 2013 - 23:00:00	-832.68
04 Dec 2013 - 21:00:00	-835.88
04 Dec 2013 - 19:00:00	-832.68
04 Dec 2013 - 17:00:00	-823.25
04 Dec 2013 - 15:00:00	-805.16
04 Dec 2013 - 13:00:00	-785.22
04 Dec 2013 - 11:00:00	-761.26
04 Dec 2013 - 09:00:00	-736.61
04 Dec 2013 - 07:00:00	-731.91
04 Dec 2013 - 05:00:00	-743.81
04 Dec 2013 - 03:00:00	-758.71
04 Dec 2013 - 01:00:00	-771.69
03 Dec 2013 - 23:00:00	-785.22
03 Dec 2013 - 21:00:00	-793.62
03 Dec 2013 - 19:00:00	-790.8
03 Dec 2013 - 17:00:00	-774.35
03 Dec 2013 - 15:00:00	-756.18
03 Dec 2013 - 13:00:00	-734.25
03 Dec 2013 - 11:00:00	-705.13
03 Dec 2013 - 09:00:00	-680.57
03 Dec 2013 - 07:00:00	-684.52
03 Dec 2013 - 05:00:00	-698.79
03 Dec 2013 - 03:00:00	-713.79
03 Dec 2013 - 01:00:00	-729.59
02 Dec 2013 - 23:00:00	-741.39
02 Dec 2013 - 21:00:00	-746.24
02 Dec 2013 - 19:00:00	-741.39
02 Dec 2013 - 17:00:00	-724.99
02 Dec 2013 - 15:00:00	-707.27
02 Dec 2013 - 13:00:00	-682.54
02 Dec 2013 - 11:00:00	-652.58
02 Dec 2013 - 09:00:00	-628.78
02 Dec 2013 - 07:00:00	-635.37
02 Dec 2013 - 05:00:00	-650.81
02 Dec 2013 - 03:00:00	-667.16

02 Dec 2013 - 01:00:00	-680.57
01 Dec 2013 - 23:00:00	-694.64
01 Dec 2013 - 21:00:00	-700.89
01 Dec 2013 - 19:00:00	-694.64
01 Dec 2013 - 17:00:00	-680.57
01 Dec 2013 - 15:00:00	-661.6
01 Dec 2013 - 13:00:00	-635.37
01 Dec 2013 - 11:00:00	-606.93
01 Dec 2013 - 09:00:00	-584.03
01 Dec 2013 - 07:00:00	-586.78
01 Dec 2013 - 05:00:00	-602.48
01 Dec 2013 - 03:00:00	-614.53
01 Dec 2013 - 01:00:00	-628.78
30 Nov 2013 - 23:00:00	-642.12
30 Nov 2013 - 21:00:00	-649.05
30 Nov 2013 - 19:00:00	-647.3
30 Nov 2013 - 17:00:00	-630.41
30 Nov 2013 - 15:00:00	-611.46
30 Nov 2013 - 13:00:00	-585.4
30 Nov 2013 - 11:00:00	-550.79
30 Nov 2013 - 09:00:00	-521.61
30 Nov 2013 - 07:00:00	-516.4
30 Nov 2013 - 05:00:00	-525.87
30 Nov 2013 - 03:00:00	-539.13
30 Nov 2013 - 01:00:00	-556.83
29 Nov 2013 - 23:00:00	-572
29 Nov 2013 - 21:00:00	-586.78
29 Nov 2013 - 19:00:00	-590.96
29 Nov 2013 - 17:00:00	-577.27
29 Nov 2013 - 15:00:00	-548.41
29 Nov 2013 - 13:00:00	-521.61
29 Nov 2013 - 11:00:00	-507.31
29 Nov 2013 - 09:00:00	-511.31
29 Nov 2013 - 07:00:00	-525.87
29 Nov 2013 - 05:00:00	-539.13

29 Nov 2013 - 03:00:00	-548.41
29 Nov 2013 - 01:00:00	-555.61
28 Nov 2013 - 23:00:00	-563.04
28 Nov 2013 - 21:00:00	-573.31
28 Nov 2013 - 19:00:00	-579.95
28 Nov 2013 - 17:00:00	-574.62
28 Nov 2013 - 15:00:00	-553.19
28 Nov 2013 - 13:00:00	-525.87
28 Nov 2013 - 11:00:00	-490.18
28 Nov 2013 - 09:00:00	-462.74
28 Nov 2013 - 07:00:00	-464.35
28 Nov 2013 - 05:00:00	-476.02
28 Nov 2013 - 03:00:00	-489.27
28 Nov 2013 - 01:00:00	-506.33
27 Nov 2013 - 23:00:00	-523.73
27 Nov 2013 - 21:00:00	-533.51
27 Nov 2013 - 19:00:00	-531.3
27 Nov 2013 - 17:00:00	-512.32
27 Nov 2013 - 15:00:00	-489.27
27 Nov 2013 - 13:00:00	-462.74
27 Nov 2013 - 11:00:00	-434.28
27 Nov 2013 - 09:00:00	-411.88
27 Nov 2013 - 07:00:00	-413.76
27 Nov 2013 - 05:00:00	-423.42
27 Nov 2013 - 03:00:00	-432.89
27 Nov 2013 - 01:00:00	-442.84
26 Nov 2013 - 23:00:00	-452.55
26 Nov 2013 - 21:00:00	-461.14
26 Nov 2013 - 19:00:00	-464.35
26 Nov 2013 - 17:00:00	-451.02
26 Nov 2013 - 15:00:00	-432.19
26 Nov 2013 - 13:00:00	-411.26
26 Nov 2013 - 11:00:00	-388.55
26 Nov 2013 - 09:00:00	-371.82
26 Nov 2013 - 07:00:00	-372.32

26 Nov 2013 - 05:00:00	-378.95
26 Nov 2013 - 03:00:00	-386.37
26 Nov 2013 - 01:00:00	-394.11
25 Nov 2013 - 23:00:00	-401.6
25 Nov 2013 - 21:00:00	-407.58
25 Nov 2013 - 19:00:00	-408.8
25 Nov 2013 - 17:00:00	-400.43
25 Nov 2013 - 15:00:00	-388.55
25 Nov 2013 - 13:00:00	-374.33
25 Nov 2013 - 11:00:00	-356.1
25 Nov 2013 - 09:00:00	-342.13
25 Nov 2013 - 07:00:00	-342.55
25 Nov 2013 - 05:00:00	-348.1
25 Nov 2013 - 03:00:00	-353.39
25 Nov 2013 - 01:00:00	-359.32
24 Nov 2013 - 23:00:00	-364.5
24 Nov 2013 - 21:00:00	-368.85
24 Nov 2013 - 19:00:00	-368.36
24 Nov 2013 - 17:00:00	-359.32
24 Nov 2013 - 15:00:00	-348.1
24 Nov 2013 - 13:00:00	-333.96
24 Nov 2013 - 11:00:00	-319.49
24 Nov 2013 - 09:00:00	-310.99
24 Nov 2013 - 07:00:00	-313.07
24 Nov 2013 - 05:00:00	-316.96
24 Nov 2013 - 03:00:00	-320.95
24 Nov 2013 - 01:00:00	-325.42
23 Nov 2013 - 23:00:00	-329.24
23 Nov 2013 - 21:00:00	-332.37
23 Nov 2013 - 19:00:00	-331.98
23 Nov 2013 - 17:00:00	-326.56
23 Nov 2013 - 15:00:00	-316.24
23 Nov 2013 - 13:00:00	-303.91
23 Nov 2013 - 11:00:00	-292.19
23 Nov 2013 - 09:00:00	-285.35
23 Nov 2013 - 07:00:00	-287.1
23 Nov 2013 - 05:00:00	-289.78
23 Nov 2013 - 03:00:00	-292.5
23 Nov 2013 - 01:00:00	-294.96
22 Nov 2013 - 23:00:00	-297.15
22 Nov 2013 - 21:00:00	-298.1
22 Nov 2013 - 19:00:00	-296.83
22 Nov 2013 - 17:00:00	-291.59
22 Nov 2013 - 15:00:00	-283.9
22 Nov 2013 - 13:00:00	-275.51
22 Nov 2013 - 11:00:00	-270.96
22 Nov 2013 - 09:00:00	-271.49
22 Nov 2013 - 07:00:00	-273.35
22 Nov 2013 - 05:00:00	-275.51
22 Nov 2013 - 03:00:00	-277.7
22 Nov 2013 - 01:00:00	-279.92
21 Nov 2013 - 23:00:00	-282.18
21 Nov 2013 - 21:00:00	-283.9
21 Nov 2013 - 19:00:00	-284.48
21 Nov 2013 - 17:00:00	-281.61
21 Nov 2013 - 15:00:00	-275.23
21 Nov 2013 - 13:00:00	-270.17
21 Nov 2013 - 11:00:00	-268.87
21 Nov 2013 - 09:00:00	-269.39
21 Nov 2013 - 07:00:00	-270.69
21 Nov 2013 - 05:00:00	-272.81
21 Nov 2013 - 03:00:00	-274.96
21 Nov 2013 - 01:00:00	-277.15
20 Nov 2013 - 23:00:00	-278.53
20 Nov 2013 - 21:00:00	-278.53
20 Nov 2013 - 19:00:00	-278.53
20 Nov 2013 - 17:00:00	-274.96
20 Nov 2013 - 15:00:00	-269.13
20 Nov 2013 - 13:00:00	-262.52
20 Nov 2013 - 11:00:00	-257.41

20 Nov 2013 - 09:00:00	-256.69
20 Nov 2013 - 07:00:00	-258.13
20 Nov 2013 - 05:00:00	-259.34
20 Nov 2013 - 03:00:00	-260.8
20 Nov 2013 - 01:00:00	-262.28
19 Nov 2013 - 23:00:00	0
19 Nov 2013 - 21:00:00	-264.52
19 Nov 2013 - 19:00:00	-264.27
19 Nov 2013 - 17:00:00	-261.04
19 Nov 2013 - 15:00:00	-256.69
19 Nov 2013 - 13:00:00	-252.25
19 Nov 2013 - 11:00:00	-248.61
19 Nov 2013 - 09:00:00	-247.72
19 Nov 2013 - 07:00:00	-248.61
19 Nov 2013 - 05:00:00	-249.74
19 Nov 2013 - 03:00:00	-250.87
19 Nov 2013 - 01:00:00	-252.02
18 Nov 2013 - 23:00:00	-252.94
18 Nov 2013 - 21:00:00	-253.63
18 Nov 2013 - 19:00:00	-253.17
18 Nov 2013 - 17:00:00	-251.56
18 Nov 2013 - 15:00:00	-248.83
18 Nov 2013 - 13:00:00	-246.16
18 Nov 2013 - 11:00:00	-243.55
18 Nov 2013 - 09:00:00	-242.69
18 Nov 2013 - 07:00:00	-243.12
18 Nov 2013 - 05:00:00	-243.98
18 Nov 2013 - 03:00:00	-244.85
18 Nov 2013 - 01:00:00	-245.94
17 Nov 2013 - 23:00:00	-246.83
17 Nov 2013 - 21:00:00	-247.27
17 Nov 2013 - 19:00:00	-247.27
17 Nov 2013 - 17:00:00	-245.72
17 Nov 2013 - 15:00:00	-243.55
17 Nov 2013 - 13:00:00	-241.19

17 Nov 2013 - 11:00:00	-238.88
17 Nov 2013 - 09:00:00	-237.63
17 Nov 2013 - 07:00:00	-237.84
17 Nov 2013 - 05:00:00	-238.46
17 Nov 2013 - 03:00:00	-239.29
17 Nov 2013 - 01:00:00	-240.13
16 Nov 2013 - 23:00:00	-240.77
16 Nov 2013 - 21:00:00	-241.19
16 Nov 2013 - 19:00:00	-241.4
16 Nov 2013 - 17:00:00	-240.56
16 Nov 2013 - 15:00:00	-239.29
16 Nov 2013 - 13:00:00	-238.46
16 Nov 2013 - 11:00:00	-237.84
16 Nov 2013 - 09:00:00	-238.04
16 Nov 2013 - 07:00:00	-238.25
16 Nov 2013 - 05:00:00	-238.88
16 Nov 2013 - 03:00:00	-239.71
16 Nov 2013 - 01:00:00	-240.56
15 Nov 2013 - 23:00:00	-241.4
15 Nov 2013 - 21:00:00	-242.04
15 Nov 2013 - 19:00:00	-242.26
15 Nov 2013 - 17:00:00	-241.62
15 Nov 2013 - 15:00:00	-239.92
15 Nov 2013 - 13:00:00	-237.84
15 Nov 2013 - 11:00:00	-235.99
15 Nov 2013 - 09:00:00	-235.17
15 Nov 2013 - 07:00:00	-235.99
15 Nov 2013 - 05:00:00	-236.6
15 Nov 2013 - 03:00:00	-237.42
15 Nov 2013 - 01:00:00	-238.46
14 Nov 2013 - 23:00:00	-239.5
14 Nov 2013 - 21:00:00	-240.56
14 Nov 2013 - 19:00:00	-241.19
14 Nov 2013 - 17:00:00	-240.56
14 Nov 2013 - 15:00:00	-239.29

14 Nov 2013 - 13:00:00	-238.04
14 Nov 2013 - 11:00:00	-236.4
14 Nov 2013 - 09:00:00	-235.78
14 Nov 2013 - 07:00:00	-236.6
14 Nov 2013 - 05:00:00	-237.63
14 Nov 2013 - 03:00:00	-238.88
14 Nov 2013 - 01:00:00	-239.92
13 Nov 2013 - 23:00:00	-241.4
13 Nov 2013 - 21:00:00	-242.9
13 Nov 2013 - 19:00:00	-243.98
13 Nov 2013 - 17:00:00	-244.2
13 Nov 2013 - 15:00:00	-243.55
13 Nov 2013 - 13:00:00	-242.47
13 Nov 2013 - 11:00:00	-241.62
13 Nov 2013 - 09:00:00	-240.98
13 Nov 2013 - 07:00:00	-242.47
13 Nov 2013 - 05:00:00	-244.41
13 Nov 2013 - 03:00:00	-246.38
13 Nov 2013 - 01:00:00	-248.61
12 Nov 2013 - 23:00:00	-251.1
12 Nov 2013 - 21:00:00	-253.63
12 Nov 2013 - 19:00:00	-255.75
12 Nov 2013 - 17:00:00	-256.22
12 Nov 2013 - 15:00:00	-252.71
12 Nov 2013 - 13:00:00	-248.16
12 Nov 2013 - 11:00:00	-242.69
12 Nov 2013 - 09:00:00	-242.47
12 Nov 2013 - 07:00:00	-244.63
12 Nov 2013 - 05:00:00	-246.83
12 Nov 2013 - 03:00:00	-249.28
12 Nov 2013 - 01:00:00	-252.25
11 Nov 2013 - 23:00:00	-254.8
11 Nov 2013 - 21:00:00	-258.13
11 Nov 2013 - 19:00:00	-261.29
11 Nov 2013 - 17:00:00	-264.52
11 Nov 2013 - 15:00:00	-268.09
11 Nov 2013 - 13:00:00	-272.28
11 Nov 2013 - 11:00:00	-276.87
11 Nov 2013 - 09:00:00	-281.9
11 Nov 2013 - 07:00:00	-287.4
11 Nov 2013 - 05:00:00	-294.03
11 Nov 2013 - 03:00:00	-326.56
11 Nov 2013 - 01:00:00	-333.56
10 Nov 2013 - 23:00:00	-341.29
10 Nov 2013 - 21:00:00	-350.28
10 Nov 2013 - 19:00:00	-360.25
10 Nov 2013 - 17:00:00	-371.82
10 Nov 2013 - 15:00:00	-382.09
10 Nov 2013 - 13:00:00	-389.09
10 Nov 2013 - 11:00:00	-388
10 Nov 2013 - 09:00:00	-383.15
10 Nov 2013 - 07:00:00	-381.03
10 Nov 2013 - 05:00:00	-390.2
10 Nov 2013 - 03:00:00	-401.02
10 Nov 2013 - 01:00:00	-413.76
09 Nov 2013 - 23:00:00	-427.42
09 Nov 2013 - 21:00:00	-445.78
09 Nov 2013 - 19:00:00	-448.76
09 Nov 2013 - 17:00:00	-429.45
09 Nov 2013 - 15:00:00	-402.78
09 Nov 2013 - 13:00:00	-372.82
09 Nov 2013 - 11:00:00	-345.08
09 Nov 2013 - 09:00:00	-332.37
09 Nov 2013 - 07:00:00	-343.81
09 Nov 2013 - 05:00:00	-358.85
09 Nov 2013 - 03:00:00	-373.32
09 Nov 2013 - 01:00:00	-391.86
08 Nov 2013 - 23:00:00	-410.64
08 Nov 2013 - 21:00:00	-430.82
08 Nov 2013 - 19:00:00	-450.26

08 Nov 2013 - 17:00:00	-448.76
08 Nov 2013 - 15:00:00	-428.09
08 Nov 2013 - 13:00:00	-399.26
08 Nov 2013 - 11:00:00	-365.46
08 Nov 2013 - 09:00:00	-337.58
08 Nov 2013 - 07:00:00	-338.4
08 Nov 2013 - 05:00:00	-351.61
08 Nov 2013 - 03:00:00	-365.94
08 Nov 2013 - 01:00:00	-381.56
07 Nov 2013 - 23:00:00	-396.38
07 Nov 2013 - 21:00:00	-405.17
07 Nov 2013 - 19:00:00	-401.02
07 Nov 2013 - 17:00:00	-383.68
07 Nov 2013 - 15:00:00	-362.12
07 Nov 2013 - 13:00:00	0
07 Nov 2013 - 11:00:00	-310.3
07 Nov 2013 - 09:00:00	-285.35
07 Nov 2013 - 07:00:00	-284.19
07 Nov 2013 - 05:00:00	-295.27
07 Nov 2013 - 03:00:00	-307.24
07 Nov 2013 - 01:00:00	-319.49
06 Nov 2013 - 23:00:00	-331.58
06 Nov 2013 - 21:00:00	-339.63
06 Nov 2013 - 19:00:00	-336.77
06 Nov 2013 - 17:00:00	-321.69
06 Nov 2013 - 15:00:00	-303.25
06 Nov 2013 - 13:00:00	-282.47
06 Nov 2013 - 11:00:00	-257.89
06 Nov 2013 - 09:00:00	-239.08
06 Nov 2013 - 07:00:00	-241.4
06 Nov 2013 - 05:00:00	-251.33
06 Nov 2013 - 03:00:00	-261.29
06 Nov 2013 - 01:00:00	-271.49
05 Nov 2013 - 23:00:00	-280.2
05 Nov 2013 - 21:00:00	-284.77
05 Nov 2013 - 19:00:00	-280.77
05 Nov 2013 - 17:00:00	-268.35
05 Nov 2013 - 15:00:00	-252.71
05 Nov 2013 - 13:00:00	-235.78
05 Nov 2013 - 11:00:00	-217.02
05 Nov 2013 - 09:00:00	-201.44
05 Nov 2013 - 07:00:00	-199.73
05 Nov 2013 - 05:00:00	-206.69
05 Nov 2013 - 03:00:00	-214.05
05 Nov 2013 - 01:00:00	-221.32
04 Nov 2013 - 23:00:00	-227.68
04 Nov 2013 - 21:00:00	-230.2
04 Nov 2013 - 19:00:00	-226.54
04 Nov 2013 - 17:00:00	-216.31
04 Nov 2013 - 15:00:00	-203
04 Nov 2013 - 13:00:00	-188.72
04 Nov 2013 - 11:00:00	-173.81
04 Nov 2013 - 09:00:00	-160.79
04 Nov 2013 - 07:00:00	-159.03
04 Nov 2013 - 05:00:00	-164.63
04 Nov 2013 - 03:00:00	-170.39
04 Nov 2013 - 01:00:00	-176.06
03 Nov 2013 - 23:00:00	-180.57
03 Nov 2013 - 21:00:00	-181.49
03 Nov 2013 - 19:00:00	-177.21
03 Nov 2013 - 17:00:00	-167.88
03 Nov 2013 - 15:00:00	-156.44
03 Nov 2013 - 13:00:00	-144.66
03 Nov 2013 - 11:00:00	-133.62
03 Nov 2013 - 09:00:00	-122.76
03 Nov 2013 - 07:00:00	-118.52
03 Nov 2013 - 05:00:00	-120.62
03 Nov 2013 - 03:00:00	-123.16
03 Nov 2013 - 01:00:00	-125.83
02 Nov 2013 - 23:00:00	-128.4

02 Nov 2013 - 21:00:00	-129.58
02 Nov 2013 - 19:00:00	-127.23
02 Nov 2013 - 17:00:00	-121.64
02 Nov 2013 - 15:00:00	-114.22
02 Nov 2013 - 13:00:00	-106.12
02 Nov 2013 - 11:00:00	-98.15
02 Nov 2013 - 09:00:00	-91.88
02 Nov 2013 - 07:00:00	-90.22
02 Nov 2013 - 05:00:00	-91.2
02 Nov 2013 - 03:00:00	-92.19
02 Nov 2013 - 01:00:00	-93.31
01 Nov 2013 - 23:00:00	-94.06
01 Nov 2013 - 21:00:00	-93.49
01 Nov 2013 - 19:00:00	-90.46
01 Nov 2013 - 17:00:00	-85.55
01 Nov 2013 - 15:00:00	-79.78
01 Nov 2013 - 13:00:00	-74.55
01 Nov 2013 - 11:00:00	-70.13
01 Nov 2013 - 09:00:00	-67.07
01 Nov 2013 - 07:00:00	-66.19
01 Nov 2013 - 05:00:00	-65.82
01 Nov 2013 - 03:00:00	-65.46
01 Nov 2013 - 01:00:00	-65.05
31 Oct 2013 - 23:00:00	-64.37
31 Oct 2013 - 21:00:00	-63.35
31 Oct 2013 - 19:00:00	-61.61
31 Oct 2013 - 17:00:00	-59.43
31 Oct 2013 - 15:00:00	-56.72
31 Oct 2013 - 13:00:00	-54.58
31 Oct 2013 - 11:00:00	-52.95
31 Oct 2013 - 09:00:00	-51.72
31 Oct 2013 - 07:00:00	-51.28
31 Oct 2013 - 05:00:00	-51.13
31 Oct 2013 - 03:00:00	-51.08
31 Oct 2013 - 01:00:00	-51.08

30 Oct 2013 - 23:00:00	-50.89
30 Oct 2013 - 21:00:00	-50.54
30 Oct 2013 - 19:00:00	-50.1
30 Oct 2013 - 17:00:00	-49.51
30 Oct 2013 - 15:00:00	-48.1
30 Oct 2013 - 13:00:00	-46.92
30 Oct 2013 - 11:00:00	-46
30 Oct 2013 - 09:00:00	-45.12
30 Oct 2013 - 07:00:00	-44.73
30 Oct 2013 - 05:00:00	-44.58
30 Oct 2013 - 03:00:00	-44.53
30 Oct 2013 - 01:00:00	-44.44
29 Oct 2013 - 23:00:00	-44.24
29 Oct 2013 - 21:00:00	-43.9
29 Oct 2013 - 19:00:00	-42.83
29 Oct 2013 - 17:00:00	-42.73
29 Oct 2013 - 15:00:00	-42.39
29 Oct 2013 - 13:00:00	-42.1
29 Oct 2013 - 11:00:00	-41.8
29 Oct 2013 - 09:00:00	-42.15
29 Oct 2013 - 07:00:00	-41.9
29 Oct 2013 - 05:00:00	-41.85
29 Oct 2013 - 03:00:00	-41.9
29 Oct 2013 - 01:00:00	-42.05
28 Oct 2013 - 23:00:00	-42.05
28 Oct 2013 - 21:00:00	-42.05
28 Oct 2013 - 19:00:00	-42.1
28 Oct 2013 - 17:00:00	-42.19
28 Oct 2013 - 15:00:00	-42.1
28 Oct 2013 - 13:00:00	-42.39
28 Oct 2013 - 11:00:00	-42.73
28 Oct 2013 - 09:00:00	-42.63
28 Oct 2013 - 07:00:00	-42.49
28 Oct 2013 - 05:00:00	-42.44
28 Oct 2013 - 03:00:00	-42.29

28 Oct 2013 - 01:00:00	-42.24
27 Oct 2013 - 23:00:00	-42.05
27 Oct 2013 - 21:00:00	-42
27 Oct 2013 - 19:00:00	-41.85
27 Oct 2013 - 17:00:00	-42.1
27 Oct 2013 - 15:00:00	-42
27 Oct 2013 - 13:00:00	-42.1
27 Oct 2013 - 11:00:00	-42.58
27 Oct 2013 - 09:00:00	-42.49
27 Oct 2013 - 07:00:00	-42.19
27 Oct 2013 - 05:00:00	-42.05
27 Oct 2013 - 03:00:00	-42.05
27 Oct 2013 - 01:00:00	-42
26 Oct 2013 - 23:00:00	-41.9
26 Oct 2013 - 21:00:00	-41.85
26 Oct 2013 - 19:00:00	-41.75
26 Oct 2013 - 17:00:00	-42
26 Oct 2013 - 15:00:00	-41.85
26 Oct 2013 - 13:00:00	-42
26 Oct 2013 - 11:00:00	-42.44
26 Oct 2013 - 09:00:00	-42.34
26 Oct 2013 - 07:00:00	-42.15
26 Oct 2013 - 05:00:00	-41.95
26 Oct 2013 - 03:00:00	-41.9
26 Oct 2013 - 01:00:00	-41.8
25 Oct 2013 - 23:00:00	-41.71
25 Oct 2013 - 21:00:00	-41.61
25 Oct 2013 - 19:00:00	-41.51
25 Oct 2013 - 17:00:00	-41.61
25 Oct 2013 - 15:00:00	-41.56
25 Oct 2013 - 13:00:00	-41.75
25 Oct 2013 - 11:00:00	-42.15
25 Oct 2013 - 09:00:00	-42.05
25 Oct 2013 - 07:00:00	-41.8
25 Oct 2013 - 05:00:00	-41.66

25 Oct 2013 - 03:00:00	-41.56
25 Oct 2013 - 01:00:00	-41.46
24 Oct 2013 - 23:00:00	-41.41
24 Oct 2013 - 21:00:00	-41.32
24 Oct 2013 - 19:00:00	-41.27
24 Oct 2013 - 17:00:00	-41.41
24 Oct 2013 - 15:00:00	-41.32
24 Oct 2013 - 13:00:00	-41.41
24 Oct 2013 - 11:00:00	-41.61
24 Oct 2013 - 09:00:00	-41.8
24 Oct 2013 - 07:00:00	-41.61
24 Oct 2013 - 05:00:00	-41.66
24 Oct 2013 - 03:00:00	-41.71
24 Oct 2013 - 01:00:00	-41.66
23 Oct 2013 - 23:00:00	-41.61
23 Oct 2013 - 21:00:00	-41.46
23 Oct 2013 - 19:00:00	-41.32
23 Oct 2013 - 17:00:00	-41.22
23 Oct 2013 - 15:00:00	-41.36
23 Oct 2013 - 11:00:00	-41.66
23 Oct 2013 - 09:00:00	-41.46
23 Oct 2013 - 07:00:00	-41.56
23 Oct 2013 - 05:00:00	-41.71
23 Oct 2013 - 03:00:00	-41.56
23 Oct 2013 - 01:00:00	0
22 Oct 2013 - 23:00:00	-41.61
22 Oct 2013 - 21:00:00	-41.61
22 Oct 2013 - 19:00:00	-41.66
22 Oct 2013 - 17:00:00	-41.75
22 Oct 2013 - 13:00:00	0
22 Oct 2013 - 11:00:00	-45.36
22 Oct 2013 - 09:00:00	-45.22
22 Oct 2013 - 07:00:00	-45.12
22 Oct 2013 - 05:00:00	-45.02
22 Oct 2013 - 03:00:00	-45.02

22 Oct 2013 - 01:00:00	-44.83
21 Oct 2013 - 23:00:00	-44.88
21 Oct 2013 - 21:00:00	-44.73
21 Oct 2013 - 19:00:00	-44.63
21 Oct 2013 - 17:00:00	-44.73
21 Oct 2013 - 15:00:00	-44.92
21 Oct 2013 - 13:00:00	-44.97
21 Oct 2013 - 11:00:00	-44.88
21 Oct 2013 - 09:00:00	-44.73
21 Oct 2013 - 07:00:00	-44.29
21 Oct 2013 - 05:00:00	-44.14
21 Oct 2013 - 03:00:00	-44.14
21 Oct 2013 - 01:00:00	-44.05
20 Oct 2013 - 23:00:00	-44
20 Oct 2013 - 21:00:00	-44
20 Oct 2013 - 19:00:00	-44
20 Oct 2013 - 17:00:00	-44.44
20 Oct 2013 - 15:00:00	-44.14
20 Oct 2013 - 13:00:00	-44
20 Oct 2013 - 11:00:00	-43.9
20 Oct 2013 - 09:00:00	-43.51
20 Oct 2013 - 07:00:00	-43.02
20 Oct 2013 - 05:00:00	-42.93

20 Oct 2013 - 03:00:00	-42.83
20 Oct 2013 - 01:00:00	-42.78
19 Oct 2013 - 23:00:00	-42.68
19 Oct 2013 - 21:00:00	-42.63
19 Oct 2013 - 19:00:00	-42.63
19 Oct 2013 - 17:00:00	-43.02
19 Oct 2013 - 15:00:00	-42.88
19 Oct 2013 - 13:00:00	-42.78
19 Oct 2013 - 11:00:00	-42.49
19 Oct 2013 - 09:00:00	-42.15
19 Oct 2013 - 07:00:00	-41.71
19 Oct 2013 - 05:00:00	-41.66
19 Oct 2013 - 03:00:00	-41.66
19 Oct 2013 - 01:00:00	-41.66
18 Oct 2013 - 23:00:00	-41.61
18 Oct 2013 - 21:00:00	-41.61
18 Oct 2013 - 19:00:00	-41.75
18 Oct 2013 - 17:00:00	-42.29
18 Oct 2013 - 15:00:00	-42.05
18 Oct 2013 - 13:00:00	-41.9
18 Oct 2013 - 11:00:00	-41.71
18 Oct 2013 - 09:00:00	-41.41
18 Oct 2013 - 07:00:00	-41.02

