



Project title:

Development of sensor-based Citizens' Observatory
Community for improving quality of life in cities

Acronym: **CITI-SENSE** Grant Agreement No: **308524**

EU FP7- ENV-2012
Collaborative project

Deliverable D7.1

D7.1: CITI-SENSE Platform and architecture 1.0

Work Package 7

Date: 30.05.2014

Version: 1.00

Leading Beneficiary:	SINTEF
Editor(s):	Arne J. Berre (SINTEF), Richard Rombouts (Snowflake)
Author(s) (alphabetically):	Andrei Tamin (U-Hopper), Richard Rombouts (Snowflake), Alberto Fernandez (Sensing&Control), Alberto Olivares (Snowflake), Debbie Wilson (Snowflake), Alex McDonald (Snowflake), Mike Kobernus (NILU), Nicolas Ferry (SINTEF), Dumitru Roman (SINTEF), Boris Pokric (Dunavnet)
Dissemination level:	Public

Versioning and contribution history

Version	Date issued	Description	Contributors
0.1	31.03.2013	Initial table of content	Arne J. Berre
0.2	25.04.2013	Updated structure after Cambridge discussions	Nicolas Ferry
0.3	28.06.2013	Input on data access services with Confluence sharing	Debbie, Alex
0.4	16.08.2013	Input on visualisation widgets with Confluence sharing	Andrei
0.5	27.09.2013	Integration of input on GEOSS	Arne J. Berre
0.6	27.10.2013	Integration of input from Belgrade meeting	Arne J. Berre
0.7	14.11.2013	Updates on analysis from WP2/WP3 requirements and input from KICT	Arne J. Berre
0.8	27.11.2013	Update on data access interfaces	Richard Rombouts
0.9	13.12.2013	Update on architectural viewpoints	Arne J. Berre
0.92	15.02.2014	Update on sensor services	Arne J. Berre
0.93	10.03.2014	Update on life cycle services and requirements analysis	Arne J. Berre
0.94	15.03.2014	Update on access to CITI-SENSE WFS-T services	Richard Rombouts
0.95	21.04.2014	Update on access to SensApp server	Nicolas Ferry
0.96	28.04.2014	Update on use of Visualisation components	Boris Pokric
1.00	30.05.2014	Finalisation	Arne J. Berre

Peer review summary

Internal review 1			
Reviewer	Mirjam Fredriksen (NILU)		
Received for review	30.05.2014	Date of review	03.06.2014

Internal review 2			
Reviewer	Leonardo Santiago (Ateknea)		
Received for review	30.05.2014	Date of review	04.06.2014

Executive Summary

This D7.1 deliverable provides the foundation and guidelines for the CITI-SENSE Platform and Architecture. This deliverable on the CITI-SENSE Platform and Architecture is being incrementally developed in multiple versions. It is being complemented by further detailed technical information being continuously updated through the CITI-SENSE Confluence system wiki.

The D7.1. CITI-SENSE Platform and Architecture version 1.0 presents the foundation and initial version of the CITI-SENSE platform with the following content: Chapter 2 on Life cycle and Architectural services, INSPIRE and GEOSS chapter introduces the foundation for the CITI-SENSE architecture with a basis in the life cycle services of INSPIRE and the architectural approach of GEOSS, and related standards from CEN/TC287, ISO/TC211 and OGC. Chapter 3 analyses the CITI-SENSE Platform and Architecture requirements described in D7.2 . The Work package 2 and 3 based requirements from the CITI-SENSE Empowerment Initiatives which were analysed in D7.2 are here further analysed with respect to mappings to generic platform use cases. The baseline for the use case template is found in Annex. The chapter 4 describes the CITI-SENSE architecture following the 5 viewpoints of the ISO RM-ODP standard. Chapter 5 describes Data and Product Services for the usage of the Data and Server platform. These are further documented in Annexe A for the usage of the WFS-T server. Chapter 6 on Sensor services describes the SensApp sensor storage services. These are further documented in Annex B for the usage of SensApp. Chapter 8 describes components for the development of Visualisation and User Interaction services.

Further services and extensions in planning, such as the CITI-SENSE Data model, Management and security services Ubiquitous public access services, GEOSS integration, Social sensors and Linked (Open) Data planned for D7.3 is introduced in chapter 8.

Table of contents

EXECUTIVE SUMMARY	3
TABLE OF CONTENTS	4
ABBREVIATIONS AND ACRONYMS.....	8
1 INTRODUCTION.....	9
1.1 INCREMENTAL DEVELOPMENT	10
2 LIFE CYCLE AND ARCHITECTURAL SERVICES, INSPIRE AND GEOSS.....	11
2.1 SERVICE DESCRIPTIONS AND CEN/TC287 TR 15449 AND ISO 19119.....	11
2.2 USE OF RM-ODP VIEWPOINTS	12
2.2.1 The Enterprise Viewpoint	12
2.2.2 The Computational Viewpoint.....	12
2.2.3 The Information Viewpoint	12
2.2.4 The Engineering Viewpoint.....	13
2.2.5 The Technology Viewpoint	13
2.3 RELATIONSHIPS BETWEEN VIEWPOINTS	13
2.4 SERVICE MODELS, PROCESSES AND SERVICE ORIENTED ARCHITECTURES	14
2.5 THE MODEL-DRIVEN APPROACH	14
2.6 SYSTEM-OF-SYSTEMS ENGINEERING	14
2.7 LIFECYCLE SERVICES AND INSPIRE	15
2.7.1 Lifecycle Service Components	15
2.7.2 Value Chain of SDI Knowledge Generation.....	16
2.7.3 Overview of Stakeholders.....	17
2.7.4 Requirements for a Next Generation of SDI Services	18
2.8 ARCHITECTURAL SERVICES AND ISO 19119	18
2.9 CITI-SENSE GENERIC LIFECYCLE SERVICES	20
3 CITI-SENSE PLATFORM AND ARCHITECTURE REQUIREMENTS.....	21
3.1 CLIENTS	21
3.1.1 Client Applications	21
3.2 WP2 AND WP3 REQUIREMENTS – USE CASES	21
3.3 WORK PACKAGE 2 – CITY CASES	22
3.4 WORK PACKAGE 3A - PUBLIC PARK.....	23
3.5 WORK PACKAGE 3B – SCHOOLS PILOT	23
3.6 CITI-SENSE PLATFORM REQUIREMENTS ANALYSIS	24
4 CITI-SENSE ARCHITECTURE.....	27
4.1 RM-ODP VIEWPOINTS	27
4.2 ENTERPRISE VIEWPOINT	28
4.2.1 Sensing Systems	28
4.2.2 Platform	29
4.2.3 Consumers	29
4.3 COMPUTATIONAL VIEWPOINT.....	31
4.4 INFORMATION VIEWPOINT	40
4.5 ENGINEERING VIEWPOINT.....	42
4.6 TECHNOLOGY VIEWPOINT	44
5 DATA AND PRODUCT SERVICES	47
5.1 USAGE AND REQUIREMENTS	47
5.1.1 Data Providers	47
5.1.2 Data Consumers.....	48
5.2 LOGICAL SERVICE INTERFACES AND INFORMATION MODEL.....	49

5.2.1	Sensor Registration	50
5.2.2	Sensor Data Ingest	51
5.2.3	Data Storage and Processing.....	52
5.2.4	Data Publishing Services	52
5.3	IMPLEMENTATION	55
5.3.1	Cloud deployment.....	55
5.3.2	Sensor Registration	55
5.3.3	Sensor Data Ingest	56
5.3.4	Data Storage and Processing.....	57
5.3.5	Data Publication.....	58
6	SENSOR SERVICES.....	60
6.1	USAGE AND REQUIREMENTS.....	60
6.1.1	Sensors.....	60
6.2	LOGICAL SERVICE INTERFACES AND INFORMATION MODEL	60
6.2.1	Sensor Inputs	66
6.3	IMPLEMENTATION TECHNOLOGIES	66
6.3.1	Retrieving data in CSV format	67
6.3.2	On the relationship of SenML and SensorML	69
7	VISUALISATION AND USER INTERACTION SERVICES	73
7.1	USAGE AND REQUIREMENTS	73
7.2	LOGICAL SERVICE INTERFACES AND INFORMATION MODEL.....	73
7.3	IMPLEMENTATION TECHNOLOGIES.....	73
7.4	WIDGETS FOR MAPS.....	74
7.4.1	Google Maps + Google Maps JavaScript API	74
7.4.2	OpenStreetMap + OpenLayers JavaScript API	75
7.4.3	Kartograph.....	75
7.5	WIDGETS FOR CHARTS	75
7.5.1	Google chart tools - interactive JavaScript library.....	75
7.5.2	highcharts.js - interactive JavaScript charts	76
7.5.3	envision.js - A HTML5 time series charts	76
7.5.4	Crossfilter - Fast Multidimensional Filtering library for Coordinated Views.....	76
7.5.5	Raphael.js - JavaScript library	77
7.6	HEATMAPS	77
7.6.1	heatmap.js – A HTML5 canvas heatmap library	77
7.7	MOBILE WIDGETS.....	78
7.7.1	JavaScript (web-based) solutions.....	78
7.7.2	Native solutions	78
7.8	CROSS PLATFORM ISSUES	78
7.9	VISUALISATION TECHNOLOGIES FROM OGC, AND RELATED GEOSS PROJECTS ETC.	79
8	FURTHER SERVICES FOR CONSIDERATION.....	81
8.1	CITI-SENSE DATA MODEL.....	81
8.2	MANAGEMENT AND SECURITY SERVICES	81
8.3	UBIQUITOUS PUBLIC ACCESS SERVICES.....	81
8.4	GEOSS INTEGRATION.....	81
8.5	SOCIAL SENSORS.....	81
8.6	LINKED (OPEN) DATA SERVICES.....	81
8.7	COMMUNICATION AND COMPOSITION SERVICES	82
8.7.1	Event Services	82
8.8	PARTICIPATION AND EMPOWERMENT SERVICES.....	82
9	CONCLUSION AND FURTHER PLATFORM EVOLUTION.....	83
10	ANNEX A: ACCESS TO CITI-SENSE WFS-T SERVER.....	84
10.1	SETUP.....	84
10.2	SAMPLES	84
10.3	SENSORS DATA ACCESS VIA WFS.....	86
10.3.1	Inserting data using the WFS-T service	88

10.3.2	How does this data look like in the database?.....	92
11	ANNEX B: ACCESS TO SENSAPP SERVER	93
11.1	SETUP.....	93
11.2	INSTALLING SENSAPP DEPENDENCIES	93
11.3	DEPLOYING SENSAPP IN THE DEVELOPMENT ENVIRONMENT	93
12	ANNEX C: VISUALISATION COMPONENTS.....	94
12.1	WIDGET FRAMEWORK FOR MEASUREMENTS (DNET).....	94
12.1.1	Installation.....	94
12.2	WIDGET FRAMEWORK FOR QUESTIONNAIRES (U-HOPPER)	94
13	ANNEX I: USE CASE TEMPLATE AND USE CASES.....	97
13.1	USE CASE METHODOLOGY	97
13.2	USE CASE TEMPLATE	101
13.3	USE CASES FOR THE CITI-SENSE PLATFORM.....	102
	Use-case-name (table template)	103
	Observe - Collect static sensor data	103
	Observe - Collect mobile sensor data.....	104
	Observe – Questionnaire	105
	Publish (observations/resources).....	105
	Discover (observations/resources)	106
	Query-Access	106
	Transform.....	107
	Visualise.....	108
	Analyze	108
	Notify	109
	Security and Rights Management	109
	Manage sensors.....	110
	Manage resources: humans/sensors/data/services.....	111
14	REFERENCES.....	112

Index of figures

Figure 2-1	ISO RM-ODP viewpoints	12
Figure 2-2	RM-ODP viewpoints and Interoperability description support.....	13
Figure 2-3	Life cycle services defined by INSPIRE	16
Figure 2-4	Added value chain of SDI knowledge generation.....	17
Figure 2-5	Relationships of services in both a layered and a bus architecture	19
Figure 2-6	CITI-SENSE Generic life cycle services.....	20
Figure 3-1	CITI-SENSE Platform use cases.....	25
Figure 4-1	Enterprise viewpoint	28
Figure 4-2	Computational Viewpoint.....	31
Figure 4-3	CITI-SENSE Platform data flow.....	32
Figure 4-4	CITI-SENSE Architecture as a service oriented architecture	33
Figure 4-5	OGC Web Services Architecture	34
Figure 4-6	CITI-SENSE Platform	38
Figure 4-7	Architecture for sensor and data management	39
Figure 4-8	CITI-SENSE adaptation of GEOSS architecture.....	40

Figure 4-9 Information viewpoint.....	41
Figure 4-10 Engineering viewpoint	43
Figure 4-11 Technology viewpoint	45
Figure 5-1 The CITI-SENSE Platform and its components	49
Figure 5-2 Spatial data services	50
Figure 5-3 WFS Data ingestion.....	56
Figure 5-4 WFS Cloud services support	57
Figure 5-5 An example of the GO Publisher WFS landing page	59
Figure 7-1 SensApp platform architecture	61
Figure 7-2 Sensor local app interface	63
Figure 7-3 Sensor log interface.....	64
Figure 7-4 Bluetooth Smart and Bluetooth Smart Ready	64
Figure 7-5 BLE Device – UV Sensor	65
Figure 7-6 BL Device – Weather Meter	66
Figure 7-7 CSV export	68
Figure 8-1 Google Map widgets	74
Figure 8-2 Kartograph widgets.....	75
Figure 8-3 Google chart widgets	75
Figure 8-4 Highchart widgets	76
Figure 8-5 Envision time series widgets	76
Figure 8-6 Crossfilter widgets	76
Figure 8-7 Raphael vector graphics widget	77
Figure 8-8 Heatmap widget.....	77

Index of Tables

Table 0-1 Abbreviations and Acronyms	8
Table 2-1 Added-value chain of SDI knowledge generation.....	17
Table 3-1 Requirements summary from CITI-SENSE pilots from D7.2 analysis.....	21
Table 3-2 Requirements to use case mapping for CITI-SENSE pilots from D7.2 analysis	25
Table 4-1 Consumers of sensor data.....	29
Table 4-2 Open Standards in the CITI-SENSE architecture.....	41
Table 5-1 Use of push and pull interfaces for observation data providers	51
Table 6-1 Example of data aggregation.....	67
Table 14-1 References.....	112

Abbreviations and Acronyms

Table 0-1 Abbreviations and Acronyms

Abbreviation / Acronym	Description
DAE	Digital Agenda for Europe
EIF	European Interoperability Framework
EIS	European Interoperability Strategy
FP7	Seventh Research Framework Programme of the European Union
GEOSS	Global Earth Observation System of Systems
GFM	General Feature Model
GML	Geography Markup Language
HTML	Hyper Text Markup Language
IaaS	Infrastructure as a Service
ICT	Information and Communication Technology
INSPIRE	Infrastructure for Spatial Information in the European Community
IoT	Internet of Things
ISO	International Organization for Standardization
LOD	Linked Open Data
O&M	Observations and Measurements
OGC	Open Geospatial Consortium
OWL	Web Ontology Language
RDF	Resource Description Framework
REST	Representational State Transfer
SaaS	Software as a Service
SDI	Spatial Data Infrastructure
SOA	Service-oriented Architecture
SPARQL	SPARQL Protocol And RDF Query Language
SPARQL	SPARQL Protocol And RDF Query Language
SSNO	Semantic Sensor Networks Ontology
SSNO	Semantic Sensor Networks Ontology
URI	Uniform Resource Identifier
VGI	Volunteered Geographic Information

1 Introduction

This deliverable, D7.1. “Platform and Architecture version 1.0” provides the foundation for the CITI-SENSE architecture together with D7.2 “Core ontology network for city observatories applications”. D7.1 is providing the basis of the CITI-SENSE platform with a focus on the main services.

The further evolution from this are the D7.3 “Platform and architecture interim “, D7.4 “Platform and architecture v2.0 “ for M24, D7.5 “Platform and architecture v3.0“ for M36 and D7.6 “Platform and architecture v4.0“ for M48.

This chapter 1 is the introduction to the CITI-SENSE Platform and Architecture. It provides a short introduction to the different chapters in this report.

Chapter 2 on Life cycle and Architectural services, INSPIRE and GEOSS introduces the foundation for the CITI-SENSE architecture with a basis in the life cycle services of INSPIRE and the architectural approach of GEOSS, and related standards from CEN/TC287, ISO/TC211 and OGC. It introduces a multi viewpoint description approach with a foundation in the five RM-ODP viewpoints. These viewpoints reflects different aspects of the platform description.

Chapter 3 analyses the CITI-SENSE Platform and Architecture requirements coming from the various user pilots and work packages as described in D7.2 and related to the platform life cycle model from chapter 2. The WP based requirements which were analysed in D7.2 are here further analysed with respect to mappings to generic platform use cases. The baseline for the use case template is found in a Annex D “Use Case template”.

The chapter 4 describes the CITI-SENSE architecture following the five RM-ODP viewpoints introduced in chapter 2. The Enterprise viewpoint describes the main stakeholders and use cases for the CITI-SENSE platform. The Computational viewpoint introduces the various functional services that are being supported by the CITI-SENSE platform. The Information viewpoint describes the main information model elements and data representations that are being handled by the CITI-SENSE platform. The Engineering viewpoint describes some of the infrastructure elements of the CITI-SENSE platform. The Technology viewpoint describes actual technologies and implementation elements related to the CITI-SENSE platform. All of the various aspects of the architecture are being further described in subsequent chapters.

Chapter 5 describes Data and Product Services of the CITI-SENSE platform with a particular focus on usage of the Spatial and Environmental Data Services platform. The structure for this and the following chapters is a simplified version of the RM-ODP viewpoints with the Enterprise viewpoint covered in a usage and requirement section, the Computational and Information viewpoint covered in logical service interface and information model section, and the engineering and technology viewpoint is covered in an implementation section. The Spatial and Environmental Data Services are further documented in Annex A “Access to CITI-SENSE WFS-T server”.

Chapter 6 on Sensor services describes in particular for the usage of the sensor access platform, in particular to the mobile sensor storage platform SensApp. This is further documented in Annex B “Access to SensApp server” for the usage of SensApp.

Chapter 7 describes components for the development of Visualisation and User Interaction services. The more detailed usage of some of the visualisation components, in particular a widget set for measurement visualisations, is further described in Annex C “Visualisation Components”.

Chapter 8 describes further services for consideration. This includes in particular services for event handling with publish/subscribe protocols and notification management, and also further relationships to various forms of social media platforms in particular for the support for participation and empowerment services.

Annex A “Access to CITI-SENSE WFS-T server”, contains an introduction to the usage of the CITI-SENSE Data access server based on the WFS-T standard,.

Annex B “Access to SensApp server” contains a description of the access to the SensApp server for sensor data storage support in particular for mobile sensors, or for situations when no pre-existing sensor platform exists.

Annex C “Visualisation Components” contains a description for the use of visualisation components in particular for HTML5-based measurement widgets.

Annex D “Use case template and use cases” contains a description of the use case methodology, use case template and use cases for the CITI-SENSE platform and architecture.

1.1 Incremental Development

This document, and the later versions of this “Platform and Architecture” document are being incrementally updated with new information for versions every 6 month in the project, also relating to further details and documentation available in the platform part of the CITI-SENSE Confluence system.

2 Life Cycle and Architectural Services, INSPIRE and GEOSS

This chapter provides a foundation for the CITI-SENSE architecture with a basis in the emerging standards and approaches in the area of environmental and geospatial data services.

2.1 Service Descriptions and CEN/TC287 TR 15449 and ISO 19119

Spatial Data Infrastructures, such as the architecture and platform needed by CITI-SENSE can be regarded as a set of interconnected, distributed, information systems. Their complexity calls for a structured approach to address properly the many facets. ISO/IEC 10746-1 Open Distributed Processing - Reference Model (RM-ODP) [7] provides an overall conceptual framework for building open distributed processing systems in an incremental manner. The viewpoints of the RM-ODP standards have been widely adopted: they constitute the conceptual basis for the ISO 19100 series of geomatics standards), and they also have been employed in the OMG object management architecture [1].

Different Spatial Data Infrastructure standard organisations and initiatives like CEN/TC287 TR 15449 SDI [5], ISO/TC211, OGC and GEOSS have decided on an architectural description approach based on ISO RM-ODP, and it has been decided to adopt this approach also in CITI-SENSE.

An architecture is a set of components, connections and topologies defined through a series of viewpoints. The architecture for CITI-SENSE will have multiple users, developers, operators and reviewers. Each group will view the system from their own perspective. The purpose of an architectural description is to provide an understanding of the system from multiple viewpoints. The Architecture description helps to ensure that each viewpoint will be consistent with the requirements and with the other viewpoints.

According to RM-ODP, the content of such service specifications is described from the perspective of the following five viewpoints, which enable the separation of concerns:

- **The enterprise viewpoint** addresses service aspects from an organisational, business and user perspective.
- **The computational viewpoint** addresses service aspect from a system architect perspective.
- **The information viewpoint** addresses service aspects from a geospatial information expert perspective.
- **The engineering viewpoint** addresses service aspects from a system designer perspective.
- **The technology viewpoint** addresses service aspects from a system builder and implementer perspective.

Figure 1 illustrates the main focus of each of the five RM-ODP viewpoints, and the main question that each viewpoint addresses.

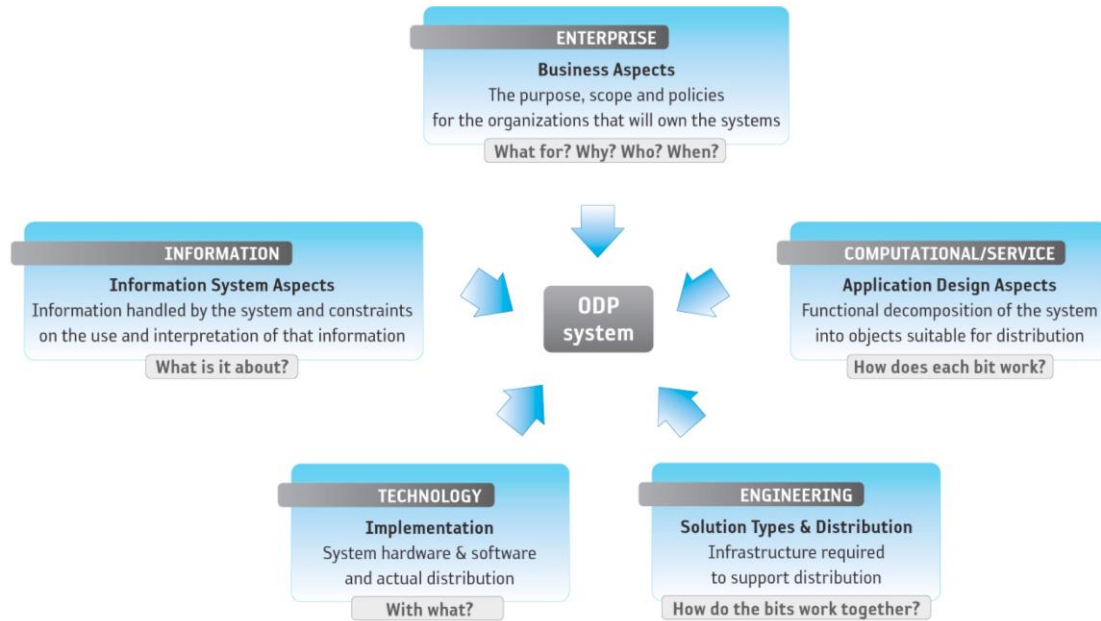


Figure 2-1 ISO RM-ODP viewpoints

2.2 Use of RM-ODP Viewpoints

2.2.1 The Enterprise Viewpoint

The enterprise viewpoint is focused on the purpose, scope and policies of an enterprise or business and how they relate to the specified system or service. An enterprise specification of a service is a model of that service and the environment with which the service interacts. It covers the role of the service in the business and the human-user roles and business policies related to the service. *In the context of the service centric view there is a particular focus on the use cases and external functionally related to the particular services.*

The enterprise viewpoint can typically be described by a set of goal oriented use cases.

2.2.2 The Computational Viewpoint

The computational viewpoint is focused on the interaction patterns between the components (services) of the system, described through their interfaces. A computational specification of a service is a model of the service interface seen from a client, and the potential set of other services that this service requires to have available, with the interacting services described as sources and sinks of information.

2.2.3 The Information Viewpoint

The information viewpoint is focused on the semantics of information and information processing. An information specification of an ODP system is a model of the information that it holds and of the information processing that it carries out. *In the context of the service centric view there is a particular focus on the information being used and provided by the particular services.*

2.2.4 The Engineering Viewpoint

The engineering viewpoint is focused on the design of distribution-oriented aspects, i.e. the infrastructure required to support distribution. An engineering specification of an ODP system defines a networked computing infrastructure that supports the system structure defined in the computational specification and provides the distribution transparencies that it defines. ODP defines the following distribution transparencies: access, failure, location, migration, relocation, replication, persistence and transaction. Security may also be a mechanism.

2.2.5 The Technology Viewpoint

The technology viewpoint is focused on the implementation of the ODP system in terms of a configuration of technology objects representing the hardware and software components of the implementation. It is constrained by cost and availability of technology objects (hardware and software products) that would satisfy this specification. These may conform to platform-specific standards that are effectively templates for technology objects.

2.3 Relationships between Viewpoints

There are important relationships between the viewpoints. In particular for the architecture of CITI-SENSE, it is important to see the relationship to the use of services from the enterprise viewpoint (WP2, WP3 and WP4), the information being provided as input and output to services from the information viewpoint (WP6), the logical service architecture itself from the computation viewpoint, the different mechanisms and architectural patterns used for distribution and different architectural styles around services in the engineering viewpoint(WP7), and the actual technologies being used in the technology viewpoint (WP8).

The various services and architectural components within the CITI-SENSE architecture is in the following being described according to the various ODP viewpoints. This is aligned also with the recommendations from the CEN/TC287 TR 15449 on Spatial Data Infrastructures [5], following these viewpoints as illustrated in Figure 2-2, where description support for different kinds of interoperability also is illustrated.

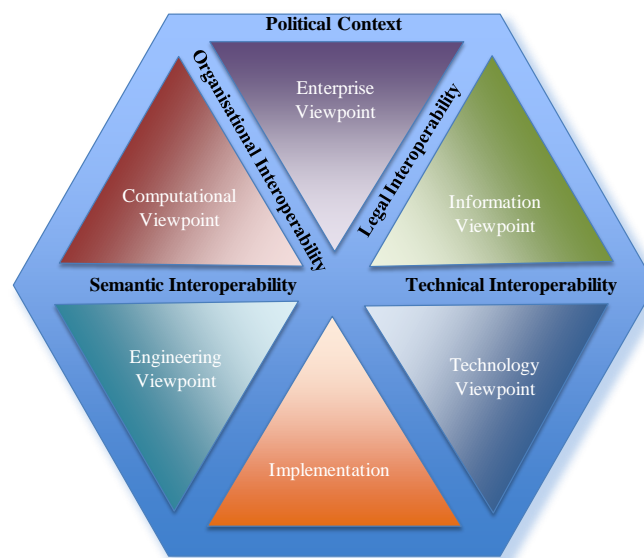


Figure 2-2 RM-ODP viewpoints and Interoperability description support

2.4 Service models, Processes and Service Oriented Architectures

The spatial data in an SDI provides a model of the real world. On top of the data, services can be built to make the data accessible through the web and to use it in an information system by e.g. viewing, downloading, or processing them. A Service Oriented Architecture (SOA) enables new and existing enterprise systems to share services, information and data across technical platforms, departments and ultimately across organizational and regional boundaries. In current systems services are also used as an integration approach for multiple architectural styles, including both document styles and synchronous RPC (Remote procedure call) styles of services, and also include integration with event management and an event driven architectural styles as well as support for interaction with sensors through services. Process support as with process management and workflow systems, including service composition, orchestration and choreography, can be viewed as the sequencing of behaviour in the implementation of services.

2.5 The Model-Driven Approach

A model driven approach has been established in INSPIRE and GEOSS, with a foundation from ISO/TC211 and OGC in particular for the data-centric view, but is now also emerging from services. Recent work in international standards such as SoaML (from OMG) and current activities such as USDL (from W3C), provides new facilities for the modelling and specification also of services in a platform neutral way.

2.6 System-of-Systems Engineering

The notion of “System of Systems” (SoS) and “System of Systems Engineering” (SoSE) emerged in many fields of applications, to address the common problem of integrating many independent, autonomous systems, frequently of large dimensions, in order to satisfy a global goal while keeping them autonomous. A system of systems is defined as a large-scale integrated system that are heterogeneous and consist of sub-systems that are independently operable, but are networked together for a common goal.

In spite of a large scale integrated system, the System of Systems components can operate independently to produce products or services satisfying their customer objectives. The component systems may be connected by implementing one or more interoperability arrangements that do not require tight coupling or strong integrations. In keeping with the definition, System of Systems key concepts are:

- *large-scale systems*: the subjection of subsystems to a central task often introduces new challenging problems that for small systems may reduce the advantage. Therefore for small systems other solutions may be more effective and efficient.
- *heterogeneous*: homogeneous systems may be merged in an integrated system without the need of SoS engineering tasks.
- *independently operable*: the realization of a System of Systems must not affect the normal and usual working of the composing systems. The SoS engineering implements agreements supplementing without supplanting the existing.
- *networked together*: the composing systems need to communicate to achieve a common goal.

These concepts help to understand when a System of Systems approach is a valuable solution. This allows it to maintain its inherent operational character even as system components join or disengage from it. Since System of Systems is a construct of both legacy and new systems, an important feature is the attention to flexibility and holistic aspects.

SoS engineering deals with planning, analyzing, organizing, and integrating the capabilities of a mix of existing and new systems into a capability greater than the sum of the capabilities of the constituent parts.

In order to achieve interoperability between the components, a System of Systems must deal with various issues. The information crosses trust boundaries, where each system is controlled and managed independently, and involves social, political and business considerations. By an architectural point-of-view, the quantitative and qualitative differences in the data exchange across the disparate systems must be dealt with. For example, the architecture typically involves different technology stacks, design models and component life cycles. In a System of Systems, the systems under consideration are loosely coupled, i.e. minimal assumptions can be made about the interface between two interacting systems. Thus, when dealing with loosely coupled systems, a system's interface can be described in terms of the data model and the role (producer or consumer) the system plays in the information exchange.

Standardisation as a means of achieving interoperability, is a key activity in a SoS engineering process. Due to the desire for adaptability in all areas of system of systems, these standards should, at a minimum, be developed as standards that are "open" to any entity participating or impacted by the System of Systems. Adaptability is necessary in a system of systems, since the membership or configuration is or can be dynamic, and the relationships between the individual systems in the System of Systems may not always be known.

2.7 Lifecycle Services and INSPIRE

2.7.1 Lifecycle Service Components

The lifecycle-based perspective for the identification of services comprises both a service-centric and a data-centric view. The service-centric view could be applied to any service-oriented system. Only the Data Centric View contains instantiations, which are specific for the geospatial and SDI domains. Likewise, GeoPortals are a specific type of geospatial applications. The following Figure 2-3 illustrates different network services supported in INSPIRE with a support also for the life cycle of decision support from registration to discovery and view and further download and invocation.

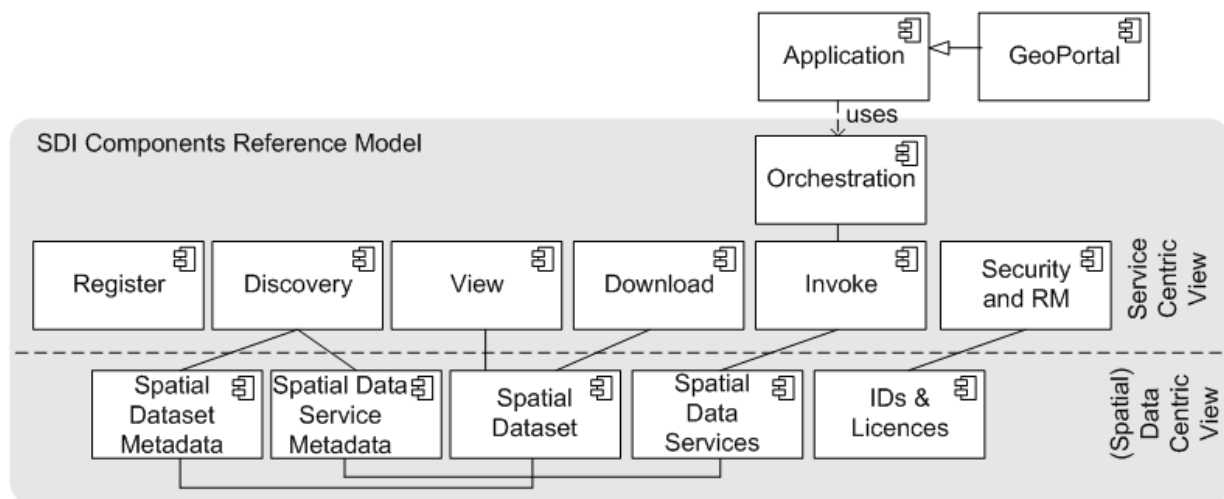


Figure 2-3 Life cycle services defined by INSPIRE

The primary organizing structure is determined by the following generic core lifecycle components:

- **Register (Publish):** for describing and publishing resources.
- **Discovery:** for searching for and discovery of resources.
- **View:** for visualising of resources.
- **Download:** for downloading and exchanging resources.
- **Invoke:** for interacting with resources.
- **Orchestration and Composition:** for providing aggregated resources including in particular workflows for service composition.
- **Security and Rights Management:** for managing access rights to resources.

On a secondary level, these components encompass both a data-centric and service-centric view. First, the roles are introduced, which are involved in the generation of knowledge about the environment and define the overall added-value chain. Then, common requirements for future SDI services are presented. In doing so, a bridge is provided between practical SDI applications and the wider political framework. This approach could equally be applied to other geospatial and non-geospatial domains beyond the SDI domain.

2.7.2 Value Chain of SDI Knowledge Generation

When analyzing the requirements of SDI services for the terrestrial, atmospheric and marine sphere, six roles may be identified each of which contributes to the generation of SDI knowledge and are therefore part of the value chain.

- **Observer**, being the initial source of information about the environment. This may reach from sensors measuring weather conditions to citizens observing species occurrences.
- **Publisher**, making a resource, such as an observation, discoverable to a wider audience, e.g. by providing required resource descriptions (metadata).
- **Discoverer**, being the entity that finds a resource, e.g. species occurrence data, based on all available descriptions.
- **Service Provider**, making information or an SDI model accessible to (and usable by) the

wider audience, e.g. by offering a standard based service for data download.

- **Service Orchestrator**, being responsible for combining existing services in a way that they create information for a distinct purpose, i.e. an SDI application focusing on a particular sphere, such as terrestrial biodiversity.
- **Decision Maker**, consuming an SDI application in order to retrieve decision supporting material and making a final decision based on the information available, e.g. designating a new protected area.

Consequently, the process workflow can be summarized as in Figure 2-4 Note that workflow services may themselves get published in order to serve as building blocks for more complex SDI solutions.

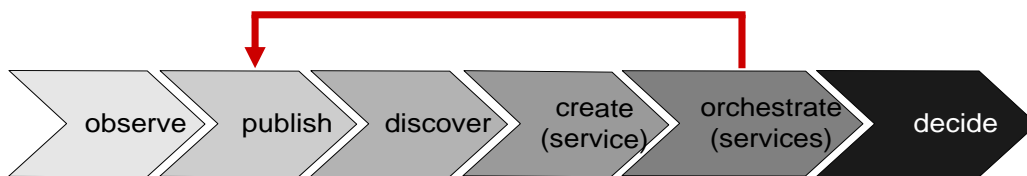


Figure 2-4 Added value chain of SDI knowledge generation

2.7.3 Overview of Stakeholders

The roles identified are played by a variety of individuals and organizations:

- **Citizens** of a particular social, political, or national community;
- **SDI agencies** on sub-national, national and European level;
- **Public authorities** of national and regional and other level;
- **Industries** from the primary, secondary and service sector;
- **Platform providers** offering frameworks on which applications may be run;
- **Infrastructure providers** offering physical components and essential services;
- **Sensor network owners** holding the sensor and basic communication hardware.

The main roles of each of the players are shown in Table 2-1 Added-value chain of SDI knowledge generation, which provides an overview of the mappings between these stakeholders and the different roles in the value chain of SDI knowledge generation. Notably, citizens can play all roles, they may even discover available information and provide new services (mash-ups).

Table 2-1 Added-value chain of SDI knowledge generation

	observe	provide	discover	create	orchestrate	decide
Citizens	x	x	x	x	x	x
SDI agencies	x	x		x		x
Public authorities		x		x		x
Industries			x	x	x	x
Platform providers				x		
Infrastructure providers				x		

Sensor network
owners

x x x x

2.7.4 Requirements for a Next Generation of SDI Services

The requirements for a next generation of SDI services can be summarized as follows:

- publication, discovery, access and visualization of SDI data sets;
- planning, publication, discovery, access and visualization of measurements;
- publication, discovery, access and visualization of objective, semi-objective and subjective observations by end users;
- transformation of data sets and fusion of observations;
- publication, discovery and access to SDI models and simulations;
- composition and invocation of workflows;
- support and enforcement of data and service policies based on identity, licenses, trust chains, etc.;
- publication, discovery, access, visualization and annotation support for controlled vocabularies, taxonomies, and ontologies;
- integration with the Semantic Web and Web 2.0; and
- interoperability with existing and planned infrastructures in the context of:
 - the most relevant initiatives at international level, such as INSPIRE, GMES, SEIS, GEOSS,
 - relevant well-established communities, including research and e-government infrastructures,
 - the mode relevant policies on international level, above all related to Public Sector Information (PSI) .

Dedicated components (SDI services) should support these requirements. They should be designed and developed leveraging existing architectural approaches and technical specifications, and re-using or extending existing tools. Attention should be paid to open international standards and communities-of-practice specifications, and to open source components in order to make the resulting system more flexible and scalable.

2.8 Architectural Services and ISO 19119

The lifecycle-based services and relevant applications can further be described in terms of their architectural components and services/services.

Figure 2-5 shows the relationships between different service categories, in the context of a complete end-to-end ICT architecture, both as a layered architecture and as a bus architecture. Here, the taxonomy follows the approach to define both generic domain-independent and specific services, such as geospatial and SDI specific services, in each of the following six groups which are colour-coded.

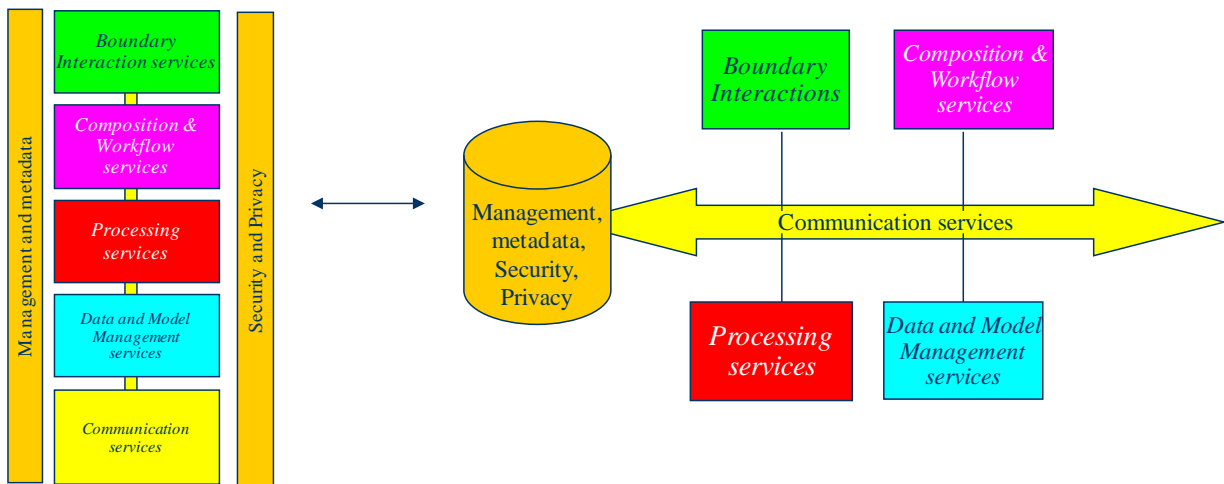


Figure 2-5 Relationships of services in both a layered and a bus architecture

The names/types have been generalised in the new version of the ISO 19119 standard [5] to be able to support a slightly broader set of services.

- **Boundary Interaction Services** are services for the management of sensors and user interfaces, graphics, multimedia and for the presentation of compound documents. Boundary Interaction services have been defined to not only include human interaction services, but also other system boundaries like sensor and actuator services. Specific services focus on providing capabilities for managing the interface between humans and Geographic Information Systems and location-based sensors and actuators. This class includes also graphic representation of features, as described in EN ISO 19117.
- **Workflow/Task Services** are services for support of specific tasks or work-related activities conducted by humans. These services support use of resources and development of products involving a sequence of activities or steps that may be conducted by different persons. The specific services focus on workflow for tasks associated with geographic and SDI information — involving processing of orders for buying and selling of geographic information and services. These services are described in more detail in EN ISO 19119.
- **Processing Services** perform large-scale computations involving substantial amounts of data. Examples include services for providing the time of day, spelling checkers and services that perform coordinate transformations (e.g. accepting a set of coordinates expressed using one reference system and converting them to a set of coordinates in a different reference system). A processing service does not include capabilities for providing persistent storage of data or transfer of data over networks. Specific services focus on processing of geographic information. EN ISO 19116 is an example of a processing service. Other examples include services for coordinate transformation, metric translation and format conversion.
- **Model/Information Management Services** are services for management of the development, manipulation and storage of metadata, conceptual schemas and datasets. The specialization of this class of services focuses on management and administration of geographic information, including conceptual schemas and data. Specific services within this class are identified in EN ISO 19119. These services are based on the content of those standards in the EN ISO 19100 series that standardize the structure of geographic information and the procedures for its administration, including: EN ISO 19107, EN ISO 19108, EN ISO 19109, EN ISO 19110, EN ISO 19111, EN

ISO 19112, EN ISO 19113, EN ISO 19114 and EN ISO 19115. Examples of such services are a query and update service for access and manipulation of geographic information and a catalogue service for management of feature catalogues.

- **Communication Services** are services for encoding and transfer of data across communications networks. The specific services focus on the transfer of geographic information across a computer network. Requirements for Transfer and Encoding services are found in EN ISO 19118.
- **System Management and Security Services** are services for the management of system components, applications and networks. These services also include management of user accounts and user access privileges. The specific services focus on user management and performance management, and on Geo Rights Management.

These six categories of services have been considered to be sufficient for most of the identified service types and services, with the escape mechanism that many new instances will be put into the processing category. There are also situations where tools and applications are composite and contain components that will span multiple categories, and also for this reason the lifecycle-based classification has been found useful as an additional classification.

The CITI-SENSE Platform and architecture will specify appropriate platform services within all of the six categories.

2.9 CITI-SENSE Generic Lifecycle Services

The initial analysis of the requirements of the various workpackages within CITI-SENSE shows that the first focus is on the collection of various forms of observation data – for static and mobile sensors, as well as from questionnaires filled out by humans.

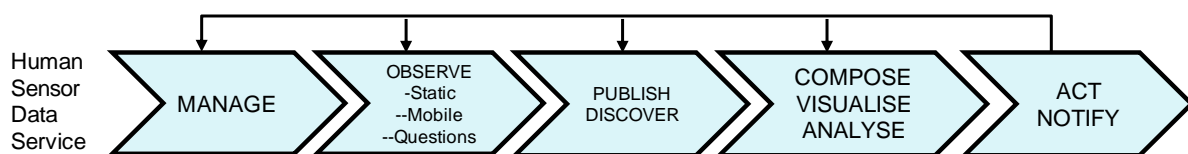


Figure 2-6 CITI-SENSE Generic life cycle services.

The life cycle approach will be able to manage and support a number of different resources, including human (citizen), sensor, data and service resources. Each of these resource types will be handled through the following phases as illustrated in Figure 2-6: Manage resources, Observe, Publish/Discover, Compose/Visualise/Analyse and Act/Notify. The CITI-SENSE platform and architecture will initially have a focus on the first phases, but is enabled to also provide support for the further phases later in the project.

3 CITI-SENSE Platform and Architecture Requirements

3.1 Clients

3.1.1 Client Applications

The purpose of the CITI-SENSE platform is ultimately to support a range of client applications. The functionality of these applications is not defined within this architecture since the goal is to support a dynamic and growing range of applications. However, there are several types of activity which these applications are expected to perform. These include:

- Analysis of data for decision support. This will be used by government users to support operational decision making or policy making.
- Visualisation of data. Consumer applications such as Google Earth allow a range of geographic data sources to be visualised. This type of application will support the understanding of the CITI-SENSE data by citizens.
- Value add and presentation. It is expected that SMEs will create mashups of CITI-SENSE data with other sources in order to provide their clients with value added services. These services may combine CITI-SENSE data with data from other sources.
- Semantic analysis. Academic and research users of CITI-SENSE will apply complex analysis to CITI-SENSE data to support research goals.

3.2 WP2 and WP3 Requirements – Use Cases

The empowerment initiatives of WP2 and WP3 provides the foundation for the requirements on the CITI-SENSE platform.

The CITI-SENSE deliverable D7.2 made an analysis of the WP2 and WP3 requirements and collected these into 20 identified requirements from R1 to R20 as shown in the table below.

- Work package 2 is devoted to develop and test methods for citizen’s empowerment in the field of urban air quality. The city case studies will be deployed in 8 cities.
- Work package 3 has as aim to support citizen’s participation in the management of public places in order to help ensuring a good environmental quality. This objective is concretised in two different pilots: public places/parks and several schools.

The aggregated requirements as defined by the work packages 2 and 3 can be found in Table 3-1.

Table 3-1 Requirements summary from CITI-SENSE pilots from D7.2 analysis

ID	Requirement	Source
R1	Collect data from static outdoor sensors: NO, NO ₂ , CO, O ₃ ,PM, Noise, Temperature, Humidity, PAH	WP2
R2	Collect data from personal sensors (indoor/outdoor): NO, CO, O ₃ , Temperature, Humidity	WP2
R3	Collect data from SmartPhone Data: GPS, Accelerometer, elevation	WP2

ID	Requirement	Source
R4	Collect data from perception data: surveys with questionnaires	WP2
R5	Collect data from air pollution monitoring and meteorological stations	WP2
R6	Collect Forecasting data on air pollution and meteorological data	WP2
R7	Collect data from User profile	WP2
R8	Collect 360-degree photoscape (video or photo)	WP2
R9	Collect data from sensors (measures each minute): mean radiant temperature (Tmrt), wind speed, air temperature and relative humidity.	WP3/Public park
R10	Collect data from calculated value (for each measuring period): Heat index, Wind chill, Outdoor Wet Bulb Globe Temperature (WBGT), PET (Physiological Environmental Temperature)	WP3/Public park
R11	Collect data from sound measurements (MP3) for each measuring period	WP3/Public park
R12	Collect data from calculated value (each minute): LAeq, L90 LAmax and LAmin	WP3/Public park
R13	Collect data from sound events: in MP3 recorded in R11, events: moment in the recording along with a label identifying the event.	WP3 – Public park
R14	Collect data from urban Landscape perception (answer to questionnaires about Local landscape perception, Participation and perception measurements and other data and photographs).	WP3 – Public park
R15	Collect data from UV Exposure	WP3 – Public park
R16	Collect data from sensor data: Temperature, Relative humidity, CO ₂ , NO ₂ , Dust, Noise, VOC, Radon.	WP3 – Schools Pilot
R17	Collect Location, School, operation hours, Time period	WP3 – Schools Pilot

The initial focus in WP2 and WP3 has been in particular on the collection of observation data, it is foreseen that further life cycle services related to analysis and decisions might emerge later in the project, and the CITI-SENSE platform and architecture is thus preparing to support also further life cycle services.

3.3 Work package 2 – City Cases

There are various types of information inputs coming from the city case studies of work package 2, the following information will be collected and available during the case studies execution:

- Static outdoor sensors.
- Personal (indoor/outdoor) sensors.
- Smartphone data
- Perception data collected during short surveys
- Data from routine air pollution monitoring stations and meteorological stations
- Forecasting of air pollution and meteorological data
- Consent for use of data;
- Video or photo

3.4 Work package 3a - Public Park

WP3 will design, implement and evaluate a pilot case in a public park in Vitoria (Spain). The main aim of this pilot is to empower citizens in the process of designing public places from an environmental point of view including comfort criteria.

The following aspects will be evaluated:

- Thermal comfort
- Acoustic comfort
- UV Radiation
- Urban landscape perception
- General satisfaction
- Other variables (safety, cleanness)
- **Other data**

The usage of this already available data will complement data from the mobile sensors and questionnaires collected during the measuring period. Thus, definition of adequate databases previously to the measuring period will:

- Provide extra information to the citizen/participant in the empowerment initiative and/or contribute to the 'on-line' perception evaluation.
- Contribute to an 'off-line' evaluation of citizens thermal comfort, acoustics and UV radiation exposure after the measuring/experience period.

3.5 Work Package 3b – Schools Pilot

Visualization of real time data for a number of different measurements:
Temperature, Relative humidity, CO2, NO2, Dust, Noise, VOC, Radon

Retrieval of the observation data both in XML and JSON representation.

General queries include

- Get sensors on school = return XML, JSON.
- Get locations on school = return XML, JSON.
- Get parameter types measured on school = return XML, JSON.
- Get measured data in this time period where parameter x's value is greater than y = return XML, JSON.
- Get sensor data, Query with all possible combinations, return types= 1) plots 2)export to excel friendly format
 - Location
 - Parameter type/types
 - School
 - [operation hours J/N]

- Time period
- Flag

The main stakeholders in this pilot are the Pupils, Teachers, Technical staff and Headmaster.

The main purpose of the applications is to give the stakeholders tools/information that will enable them to work more actively and systematically with issues that will improve **indoor air quality**. It is also an objective to raise awareness of the importance of indoor air environment on health, wellbeing and learning.

3.6 CITI-SENSE Platform Requirements Analysis

The initial set of requirements derived from WP2 and WP3 are mainly focused on the life cycle phase of collection of observations for different kinds of sensor types and data. It is, however, envisaged that requirements for further life cycle phases related to analysis and decision support might emerge later in the project.

Other technical CITI-SENSE work packages providing support for the pilot cases in WP2 and WP3 also has explicit or implicit requirements for the CITI-SENSE platform.

Work Package 4 – Citizens Observatories – has requirements for being able to support Citizens observatories portal services, and is also related to requirements for a strategy of GEOSS integration.

Work Package 5 – Empowerment - has requirements for being able to support Citizens empowerment and is currently analysing to what extent different kinds of social media support will be valuable to utilise in that context.

Work Package 6 – Products and services - has requirements for being able to access existing data products, and also for being able to develop application services both for mobile apps and for web applications.

Work Package 8 – Sensor platforms – has requirements for various sensor platforms to be able to store and retrieve data from the CITI-SENSE platform.

The CITI-SENSE platform aims to meet these requirements through a support for the full life cycle of services described in chapter 2 of this report. The following use cases are identified as generic platform use cases in order to achieve this.

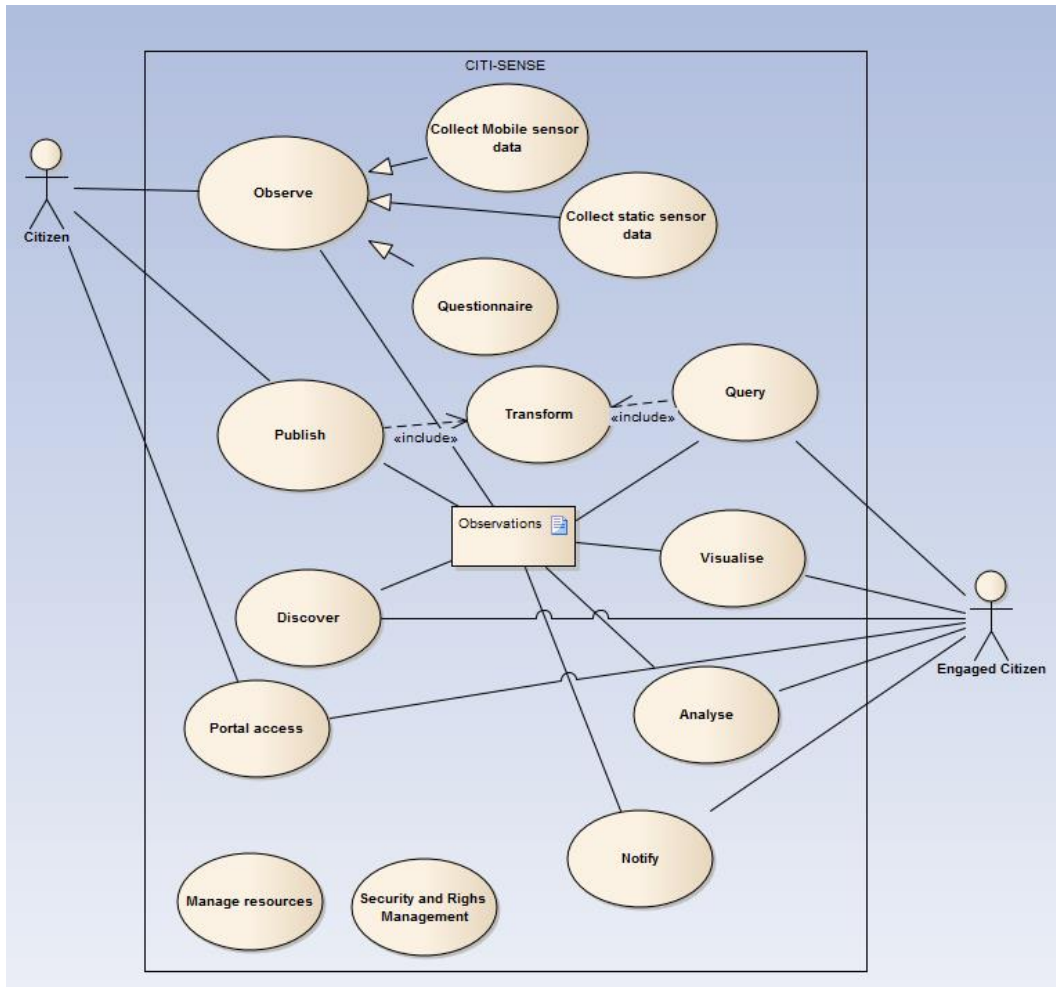


Figure 3-1 CITI-SENSE Platform use cases

The following use cases are the generic use cases, derived from a life-cycle based analysis, independent of particular observation types, which the platform will be able to support.

1. Observe – Collect observation data
2. Observe - Collect mobile sensor data
3. Observe - Collect static sensor data
4. Observe – Questionnaire
5. Publish
6. Discover
7. Query-Access
8. Transform
9. Visualise
10. Analyse (Compose, Process, Fusion (Product/Applications/Apps)...)
11. Notify (Event management, Publish/Subscribe)
12. Manage resources (humans, sensors, data, services)
13. Security and Rights Management

Table 3-2 Requirements to use case mapping for CITI-SENSE pilots from D7.2 analysis

ID	Requirement	Platform use case
R1	Collect data from static outdoor sensors: NO, NO ₂ , CO, O ₃ ,PM, Noise,	Collect static sensor

ID	Requirement	Platform use case
	Temperature, Humidity, PAH	data
R2	Collect data from personal sensors (indoor/outdoor): NO, CO, O ₃ , Temperature, Humidity	Collect mobile sensor data
R3	Collect data from SmartPhone Data: GPS, Accelerometer, elevation	Collect mobile sensor data
R4	Collect data from perception data: surveys with questionnaires	Questionnaire
R5	Collect data from air pollution monitoring and meteorological stations	Collect static sensor data
R6	Collect Forecasting data on air pollution and meteorological data	Observe
R7	Collect data from User profile	Collect mobile sensor data
R8	Collect 360-degree photoscape (video or photo)	Collect mobile sensor data
R9	Collect data from sensors (measures each minute): mean radiant temperature (T _{mrt}), wind speed, air temperature and relative humidity.	Collect mobile sensor data
R10	Collect data from calculated value (for each measuring period): Heat index, Wind chill, Outdoor Wet Bulb Globe Temperature (WBGT), PET (Physiological Environmental Temperature)	Collect mobile sensor data
R11	Collect data from sound measurements (MP3) for each measuring period	Collect mobile sensor data
R12	Collect data from calculated value (each minute): LA _{eq} , L ₉₀ LA _{max} and LA _{min}	Collect mobile sensor data
R13	Collect data from sound events: in MP3 recorded in R11, events: moment in the recording along with a label identifying the event.	Collect mobile sensor data
R14	Collect data from urban Landscape perception (answer to questionnaires about Local landscape perception, Participation and perception measurements and other data and photographs).	Collect mobile sensor data
R15	Collect data from UV Exposure	Collect mobile sensor data
R16	Collect data from sensor data: Temperature, Relative humidity, CO ₂ , NO ₂ , Dust, Noise, VOC, Radon.	Collect mobile sensor data
R17	Collect Location, School, operation hours, Time period	Collect mobile sensor data

The CITI-SENSE platform aims to meet these requirements through a support for the full life cycle of services described in chapter 2 of this report. The initial priority is on use cases to collect and visualise observation data, but further life cycle service support is foreseen and prepared for future phases of the project.

4 CITI-SENSE Architecture

The CITI-SENSE project aims to develop “citizens' observatories” to empower citizens to contribute to and participate in environmental governance, to enable to support and influence community and societal priorities and associated decision making.

To meet the project's objective, a community-based environmental monitoring and information system will be developed. This system is based on three key pillars 1) technological platforms for distributed monitoring, 2) information and communication technologies and 3) societal involvement.

The CITI-SENSE Architecture and platform focuses on the second pillar: information and communication technologies.

4.1 RM-ODP Viewpoints

Most complex system specifications are so extensive that no single individual can fully comprehend all aspects of the specifications. Furthermore, we all have different interests in a given system and different reasons for examining the system's specifications. A business executive will ask different questions of a system make-up than would a system implementer. The concept of RM-ODP viewpoints framework, therefore, is to provide separate viewpoints into the specification of a given complex system. These viewpoints each satisfy an audience with interest in a particular set of aspects of the system. Associated with each viewpoint is a viewpoint language that optimizes the vocabulary and presentation for the audience of that viewpoint¹.

RM-ODP defines five viewpoints. Each viewpoint represents an abstraction of the whole system related to a particular set of concern, as further described in section 2.2.

This chapter describes the CITI-SENSE architecture following the 5 RM-ODP viewpoints.

¹ <http://rm-odp.wikispaces.com/ODP+Viewpoints>

4.2 Enterprise Viewpoint

Enterprise viewpoint

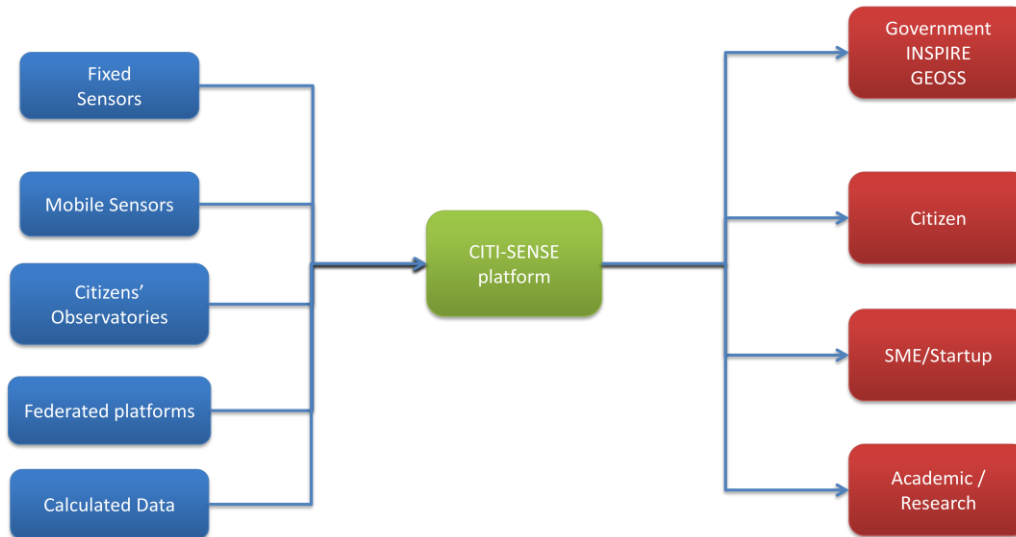


Figure 4-1 Enterprise viewpoint

4.2.1 Sensing Systems

At the data collection end of the system (the blue boxes in Figure 4-1) there are variety of sensing systems in use within CITI-SENSE, including static outdoor sensors, personal indoor/outdoor sensors, mobile sensors in smart phones, personal observations and calculated data. Many sensor types are part of 'conventional' sensor networks. Mobile sensors will allow citizens to contribute data to CITI-SENSE and act as a 'citizen observatory'. In addition to conventional sensor data, CITI-SENSE also provides a platform for exchanging "observational data", which adds a more personal dimension to the logical sensor data.

In addition to the original sensor data, several partners are also creating derived or value-added data products from these data and need a mechanism to store and publish these data.

There are three different types of data provider within the project:

- 1) Providers that already have an existing platform for receiving, storing, post-processing and publishing data;
- 2) Providers that can collect or create the data but have no mechanism for storing, post-processing or publishing data;
- 3) Providers that have developed an algorithm, service or application that can create derived or value-added data products from the original observation data. They need to be able to access the original data, have a mechanism for storing and publishing the derived or value-added data. Additionally then may want to run the algorithm, service or application on the CITI-SENSE architecture.

The CITI-SENSE technical architecture must be capable of supporting all types of data providers.

4.2.2 Platform

Providers with an existing platform for storing and processing data will join those systems to the CITI-SENSE architecture by federating their system with the CITI-SENSE services. Information from sensors will be transmitted to the federated store using existing protocols and may also be post-processed at the federated node. The data from that node will then be submitted to the CITI-SENSE hub where it can be combined with data from other sensors and distributed through a common set of discovery, download and other web services. The SensApp platform developed by SINTEF will form one such federated node within CITI-SENSE.

Providers with no existing storage and processing capability will connect their sensors directly to services on the CITI-SENSE hub. From here the sensor data can be stored and processed using the services within the CITI-SENSE hub.

Where a provider needs to run an algorithm or process over data they will create a processing service. These services will retrieve data from the CITI-SENSE hub, process that data, and submit the result back to the CITI-SENSE hub as new data. For example, a process could retrieve air quality readings from the hub, process them to create a simple summary such as a “red”, “amber” or “green” indicator of air quality. The summary would then be submitted in much the same way as a sensor reading would be.

4.2.3 Consumers

The consumers of sensor data from CITI-SENSE fall into 4 categories, as shown in Table 4-1.

Table 4-1 Consumers of sensor data

Type of Organisation	Use of CITI-SENSE	Description
Government	Decision making	These users will take information from CITI-SENSE for use in policy or operational decision making. These users will have software which can carry out analysis and processing of data specific to their applications such as GIS systems.
Academic / Research	Research and analysis	These consumers will take data for analysis. They will typically require rich data content which can be used in conjunction with data from other sources.
SMEs / Startups	Value added services / commercial exploitation	These consumers will add a layer of services on top of CITI-SENSE which will process the data for specific applications. For example, they will create visualisations, summaries or reports based on CITI-SENSE data which will then be provided to end users.
Citizen	Insight and understanding	Citizens will be accessing data via web browsers or apps and so will usually consume prepared representations of the sensor data. Some of these representations will be

		provided by CITI-SENSE itself. For example KML representations to show data in Google Earth and similar applications. Citizens will also access CITI-SENSE data indirectly via services provided by SMEs.
--	--	--

The set of Platform use cases from the previous chapter aims at addressing a full INSPIRE lifecycle. The initial requirements in CITI-SENSE have been focusing on the observation phase, but it is assumed that later requirements emerging after the initial pilots also will include requirements for later phases.

4.3 Computational Viewpoint

The Computational viewpoint is concerned with the functional decomposition of the system into different parts, typically represented as services, that interact at interfaces with operations and messages - enabling system distribution. This section describes the service contracts and interfaces for the overall CITI-SENSE architecture.

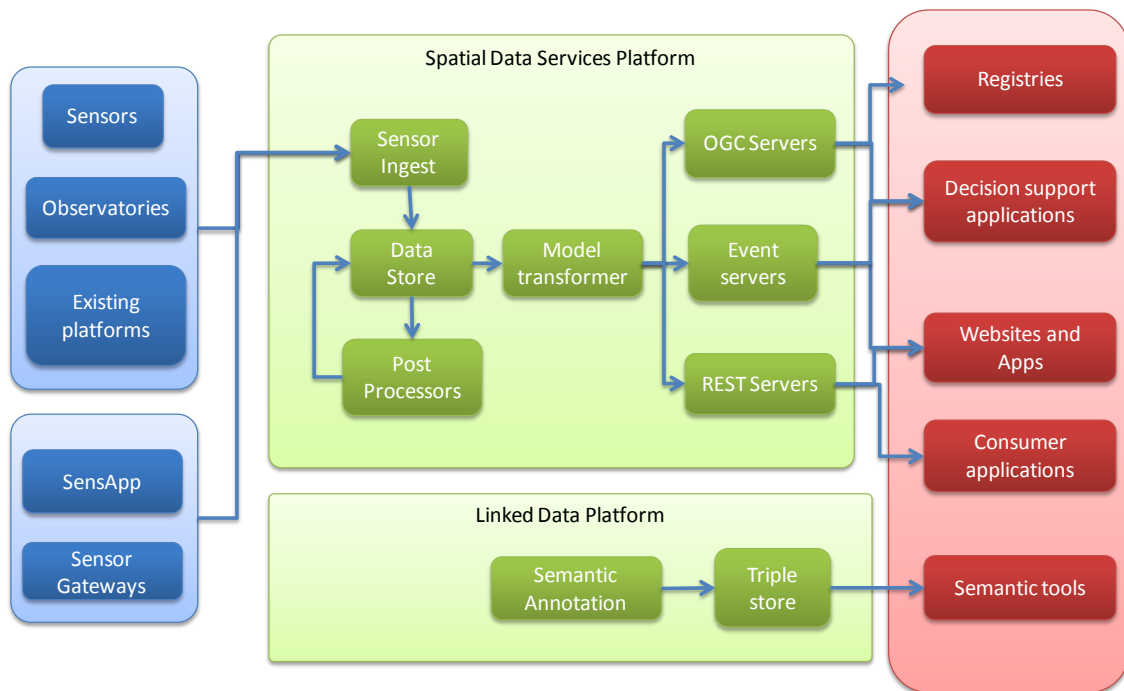


Figure 4-2 Computational Viewpoint

At a high-level the CITI-SENSE architecture has a relatively small number of computational components:

1. Data sensing - which includes the act of collecting, processing and disseminating;
2. Data ingestion - feeding data from the sensors into the data platforms; this includes transforming data between various data models;
3. Data storage - storing data in a data repository;
4. Data publication - making data available to data consumers; this includes transforming data between data models;
5. Data consuming - loading data into consumer friendly applications or services.

The elements from the computational viewpoint are illustrated in Figure 4-2. The following figures are some of the initial architecture diagrams that have been the initial basis for the work in CITI-SENSE.

An overall data flow focuses on the value network from sensor platforms to shared storage services by ingestion is shown in Figure 4-3.

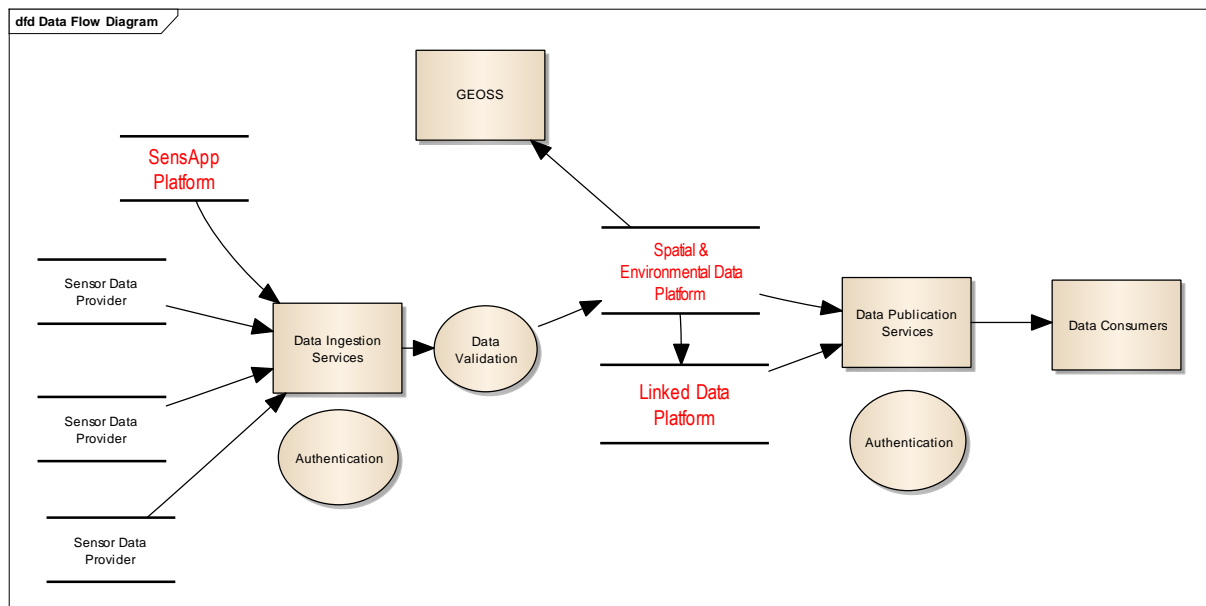
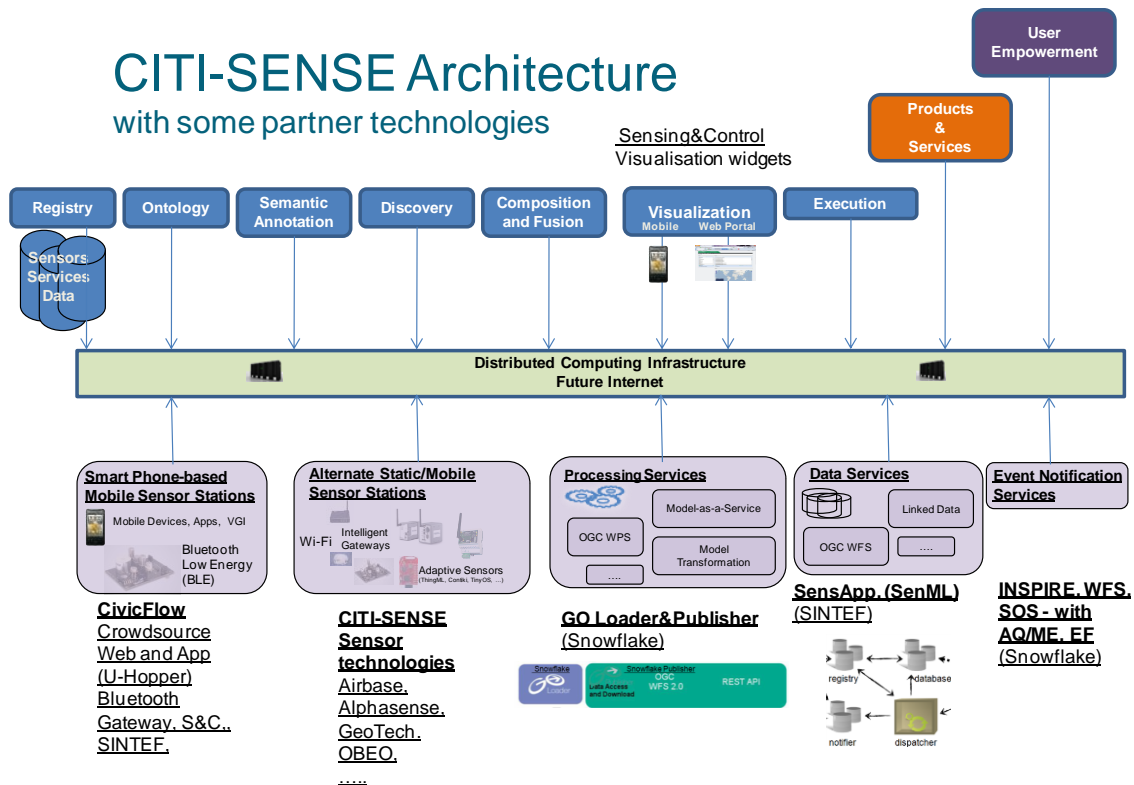


Figure 4-3 CITI-SENSE Platform data flow

The CITI-SENSE project will support a number of sensor data providers, for both mobile and static sensor platforms, which often have their own sensor data storage platform. The SensApp platform is offered in particular for the support of mobile smart phone related sensors, for the cases when a sensor data platform is not already in place. The Data ingestion services support both a pull and a push interface for the ingestion of sensor data. It will be possible with security support in terms of authentication, access control and encryption for cases when this is needed. The Spatial and Environmental Data platform will be the main platform for long term storage of sensor data.



The CITI-SENSE Architecture in Figure 4-4 shows a number of service areas relevant for the CITI-SENSE platform. The architecture illustrates also the service-oriented architecture approach of CITI-SENSE, with partner technologies to be potentially provided in the various service areas.

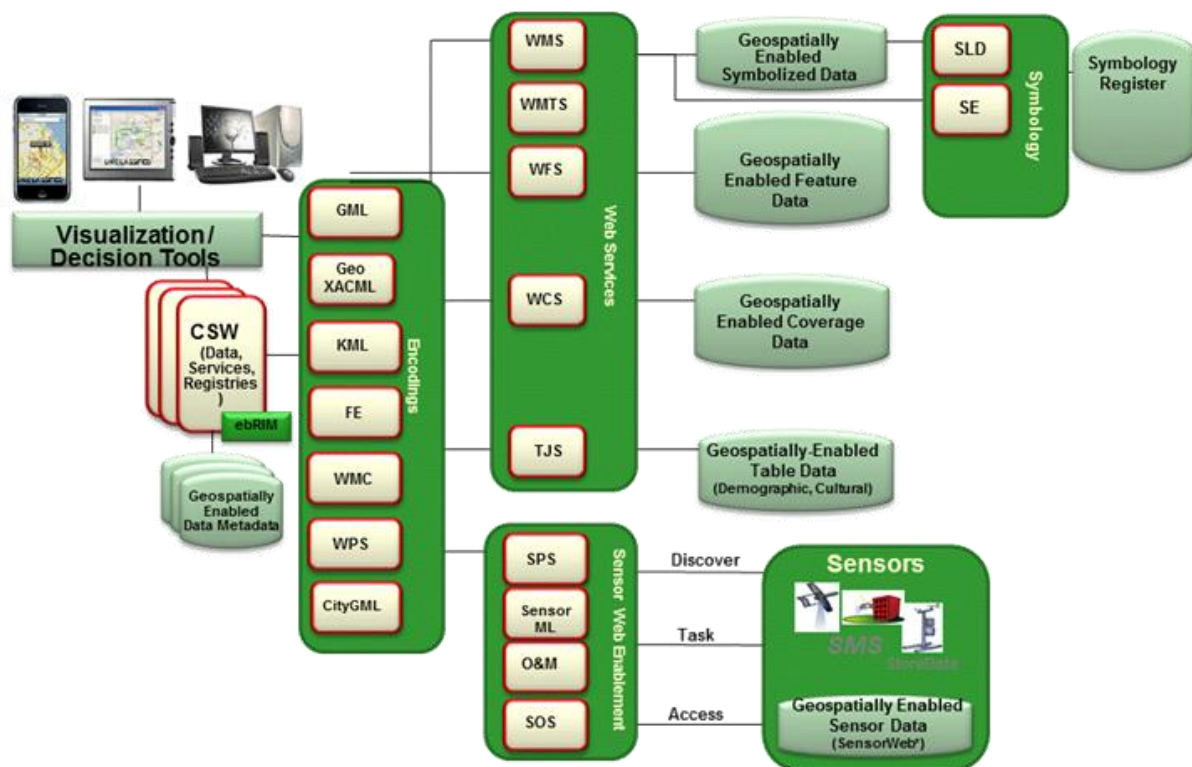


Figure 4-5 OGC Web Services Architecture

The Figure 4-5 above shows various available services and standard in the OGC Web Services Architecture for spatial supported systems. The main data server providers in CITI-SENSE represented by Snowflake and SINTEF have both been active within the development of the OGC Web Services Architecture, and the related standards within OGC and ISO/TC211. The CITI-SENSE spatial data services platform will be based on the standard specification of WFS (Web Feature Service) which is one of the core standards within the OGC Web Services Architecture.

The reader is referred to the OGC website² and the OGC Reference Model³ for a complete listing of OGC service standards and more information about each standard.

Here follows a short summary of the various standards, and how they might be used within the CITI-SENSE Architecture and platform.

- **Discovery** (of resources in general, including services): enabled by the **Catalog Service for the Web Standard CSW**, which defines common interfaces to discover, browse, and query metadata about data, services, and other potential resources for several bindings: Z39.50, CORBA, and HTTP. OGC has defined several profiles of CSW with the latest one being the OpenSearch profile. *This is a relevant standard for CITI-SENSE Discovery of resources, in the approach that will also be harmonised with the GEOSS Discovery and Access Broker (DAB).*
- **Data Access**
 - o **Web Map Service WMS**: also published as ISO 19128, provides three operations (GetCapabilities, GetMap, and GetFeatureInfo) in support of the creation and display

² www.opengeospatial.org and <http://www.opengeospatial.org/standards/is>

³ <http://www.opengeospatial.org/standards/orm>

of registered and superimposed map-like views of information that come simultaneously from multiple remote and heterogeneous sources. *This is a possible service for CITI-SENSE Map visualisations.*

- **Web Map Tiling Service WMTS:** provides for serving spatially referenced data using tile images with predefined content, extent, and resolution. WMTS trades the flexibility of custom map rendering as provided by WMS for the scalability possible by serving a fixed set of tiles. The fixed set of tiles also enables the use of standard network mechanisms for scalability such as distributed cache systems. WMTS includes both resource (REST) and procedure oriented architectural styles (KVP and SOAP). *This is a possible service for CITI-SENSE Map visualisations requiring tile images.*
 - **Web Feature Service WFS:** also published as ISO 19142, allows a client to retrieve and update geospatial data encoded in Geography Markup Language (GML) from multiple Web Feature Services. The specification defines interfaces for data access and manipulation operations on geographic features. Via these interfaces, a Web user or service can combine, use and manage geodata from different sources. A Transactional WFS (WFS-T) includes an optional Transaction operation to insert, update, or delete a feature. *The WFS and WFS-T will be the two main ISO/OGC standards supported by the CITI-SENSE Spatial Data Services platform.*
 - **Web Coverage Service WCS:** supports supports electronic retrieval of geospatial data as coverage, that is, digital geospatial information representing space/time-varying phenomena. WCS provides access to coverage data in forms that are useful for client-side rendering, as input into scientific models, and for other clients. Similar to WMS and WFS, WCS allows clients to choose portions of a server's information holdings based on spatial constraints and other query criteria. Unlike WMS, WCS defines a rich syntax for requests against Coverages; and returns data with its original semantics (instead of pictures) that may be interpreted, extrapolated, etc., and not just portrayed. Unlike WFS, WCS focuses on coverages as a specialized class of features with correspondingly streamlined functionality. *This is a possible service for CITI-SENSE Map visualisations requiring support for coverages.*
- **Sensor Web Enablement (SWE) for sensor discovery, tasking and access**
- **SWE** is designed to enable all types of Web and/or Internet-accessible sensors, instruments, and imaging devices to be accessible and, where applicable, controllable via the Web. The vision is to provide a standards-based foundation for plug-and-play web-based sensor networks. The SWE suite is enabled by the **SWE Service Model**, an Implementation Standard providing data types and mechanisms reused by the following SWE services
 - The **Sensor Observation Service SOS** Implementation Standard defines a web service interface for requesting, filtering, and retrieving observations and sensor system information. Observations may be from in-situ sensors (e.g., water monitoring devices) or dynamic sensors (e.g., imagers on Earth-observation satellites).
 - The **Sensor Planning Service SPS** Implementation Standard defines an interface to task sensors or models. Using SPS, sensors can be reprogrammed or calibrated, sensor missions can be started or changed, simulation models executed and controlled. The feasibility of a tasking request can be checked and alternatives may be provided.

- The **Sensor Alert Service SAS** Best Practice Document defines a web service interface for publishing and subscribing to alerts from sensors. Sensor nodes advertise with an SAS. If an event occurs the node will send it to the SAS via the publish operation. A consumer (interested party) may subscribe to events disseminated by the SAS. If an event occurs the SAS will alert all clients subscribed to this event type.

The CITI-SENSE sensor services will use a mapping to the Observations and Measurement standard and the SensorML representation supported in the Sensor Observation Service (SOS) with a mapping to WFS. For mobile sensors a more lightweight approach with SenML and integration with CSV files will be supported.

- **Data Visualization**

- **Styled Layer Descriptor SLD:** deals with how WMS can be extended to allow user-defined symbolization of feature and coverage data. This profile defines how the Symbology Encoding standard can be used with WMS. SLD allows for user-defined layers and named or user-defined styling in WMS. If a WMS is to symbolize features using a user-defined symbolization, the source of the feature data must be identified. The features may be in a remote WFS or WCS, or from a specific default feature/coverage store. WMS servers using remote feature data are also called Feature Portrayal Services (FPS), while those using remote coverage data are Coverage Portrayal Services (CPS).
- **Symbology Encoding SE:** specifies the format of a map-styling language for producing georeferenced maps with user-defined styling. SE is an XML language for styling information used to portray Feature and Coverage data. SE may be used together with SLD. As SE is a grammar for styling map data independent of any service interface specification it can be used flexibly by a number of services that style georeferenced information or store styling information that can be used by other services.

- **Processing**

- **Web Processing Service WPS:** defines an interface that facilitates the publishing of geospatial processes, and the discovery of and binding to those processes by clients. Processes include any algorithm, calculation or model that operates on spatially referenced data. A WPS may offer calculations as simple as subtracting one set of spatially referenced numbers from another (e.g., determining the difference in influenza cases between two different seasons), or as complicated as a global climate change model. The data required by the WPS can be delivered across a network using OGC Web Services. A WPS process may be an atomic function that performs a specific geospatial calculation. Chaining of WPS processes facilitates the creation of repeatable workflows
- Interoperability standards for catalog search are key to service oriented architectures. The OGC Catalogue Service for the Web (CSW) is a binding defined in the OpenGIS Catalogue Services Implementation Standard (CAT). The Catalog standard defines common interfaces to discover, browse, and query metadata about data, services, and other potential resources for several bindings, including a geospatial extension to OpenSearch.

The services can interact through both request/reply and publish/subscribe patterns:

- The services listed above are enabled by a variety of Encodings such as Geography Markup Language, Filter Encoding (used by the Web Feature Service), Web Map Context, KML, etc.
- These services follow a Request/Reply pattern. The OGC service architecture also accommodates for the Publish/Subscribe pattern via the PubSub 1.0 standard - an interface specification that supports the core components and concepts of the Publish/Subscribe message exchange pattern with OGC Web Services. The Publish/Subscribe pattern complements the Request/Reply pattern specified by many existing OGC Web Services. The Publish/Subscribe specification may be used either in concert with, or independently of, existing OGC Web Services to publish data of interest to interested Subscribers. This specification defines functionality independently of binding technology (e.g., KVP, SOAP, REST). Extensions to this specification may realize these core concepts with specific binding technologies.
 - o The support of both Request/Reply and Publish/Subscribe patterns is needed to fulfil the requirements of many services that have a pubsub element to them.
- Several of the services listed have been successfully prototyped and demonstrated to work in support of various use cases in scenarios in a series of OGC Interoperability Program rapid prototyping projects (testbeds and pilots). More information about those demonstrations and their outcomes are available online at <http://www.opengeospatial.org/projects/initiatives>.

The CITI-SENSE Platform and Architecture will continuously be harmonised with the provided services of the OGC and ISO/TC211 service architectures, and also provide feedback to current and future standardisation activities for further impact of CITI-SENSE project results.

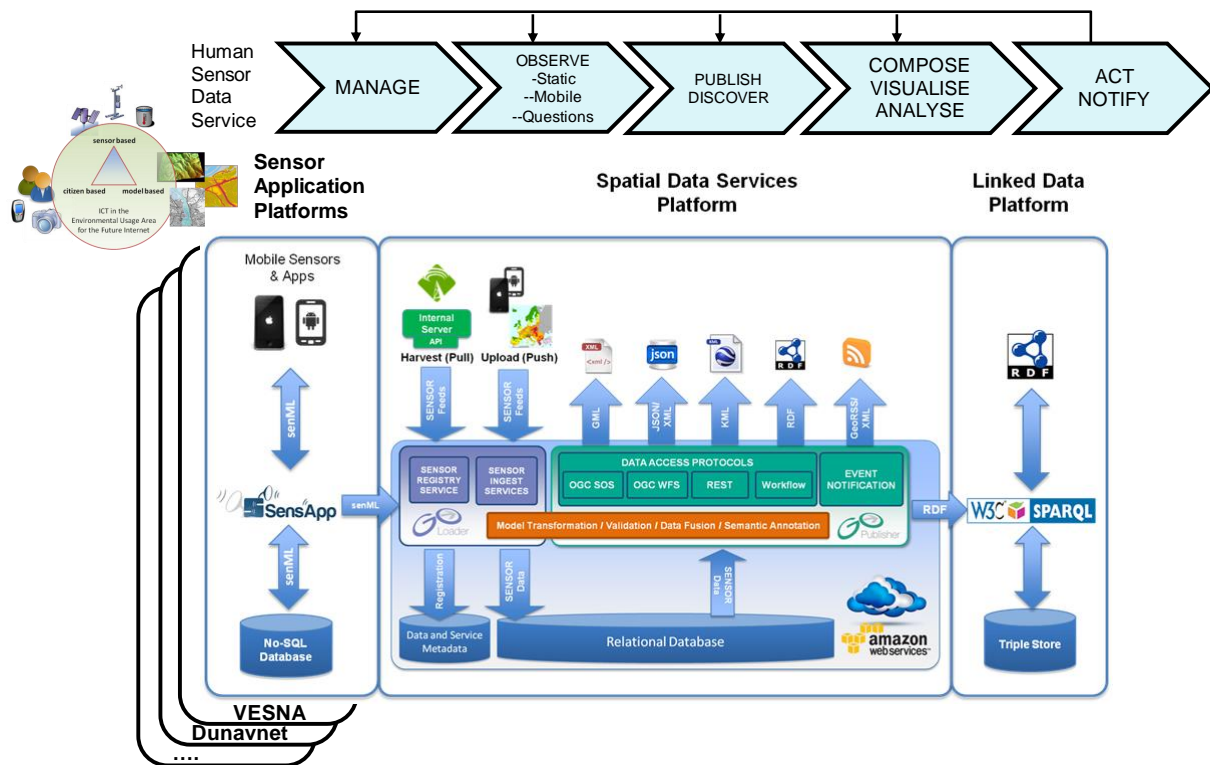


Figure 4-6 CITI-SENSE Platform

The proposed technical architecture for CITI-SENSE consist of 3 distinct platforms, as illustrated in Figure 4-6, which are developed on different technologies with similar but different end-users in mind.

1. **Sensor Application Platform (SensApp)** - allows for various sensor data provider networks to interact with the CITI-SENSE architecture.
2. **Spatial and Environmental Data Services Platform (SEDS)** - responsible for consuming, storing and publishing environmental data by adopting ISO and OGC Open Standards.
3. **Linked Data Platform** - this platform stores and manages environmental data and makes it available using Linked Data and Semantic Web principles.

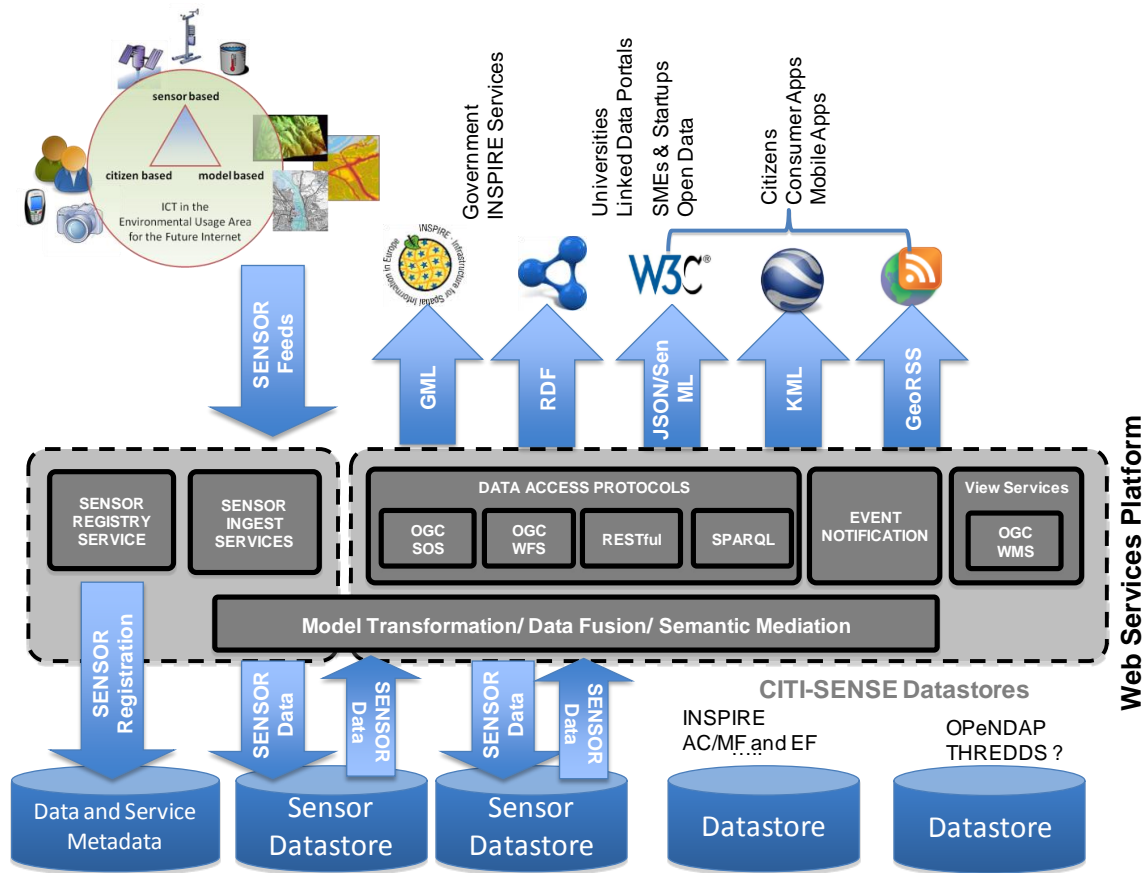


Figure 4-7 Architecture for sensor and data management

The CITI-SENSE architecture will support multiple data stores both for sensor data and for environmental and geospatial data, as illustrated in Figure 4-7. There will also be support for multiple input and output formats and standards, but the aim is to identify a minimum set of standards that will be used and supported.

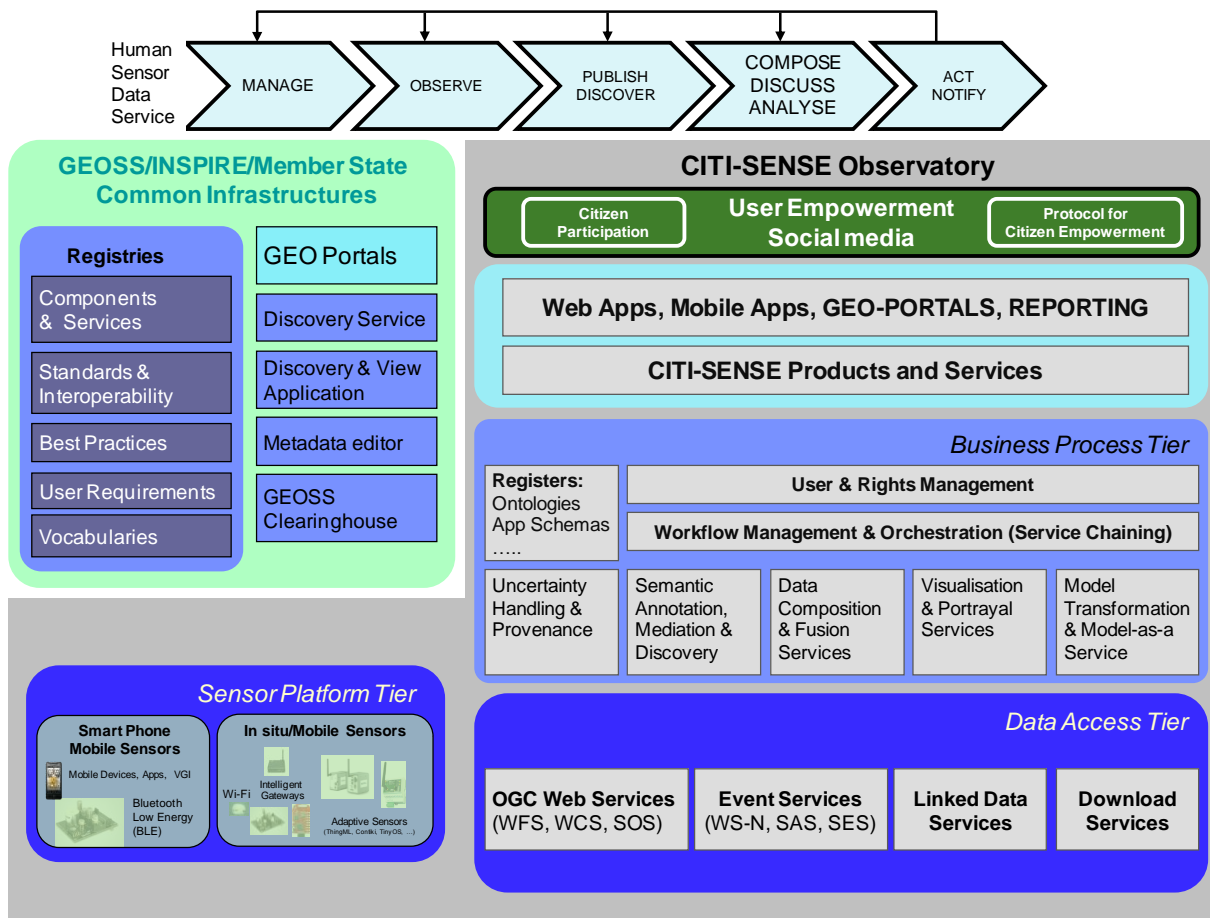


Figure 4-8 CITI-SENSE adaptation of GEOSS architecture

The CITI-SENSE project will contribute to the GEOSS Common Infrastructure (GCI), but also the CITI-SENSE architecture will be aligned with the GEOSS Architecture as shown in Figure 4-8.

4.4 Information Viewpoint

This section describes the information and data being managed and represented by these services.

The information viewpoint is further described in detail in D7.2 produced by task T7.1. The D7.2 deliverable provides a detailed study and recommendations of many existing ontologies, standards and models, many of which are incorporated in the CITI-SENSE architecture. The information viewpoint is illustrated in Figure 4-9.

The existing SenApp platform also contains storage where data from the SensApp sensor network is held. In SensApp this is implemented using a noSQL database. It will interface to the CITI-SENSE platform through web services transmitting SenML. The noSQL interface of the database is therefore internal to the SensApp platform.

Information viewpoint

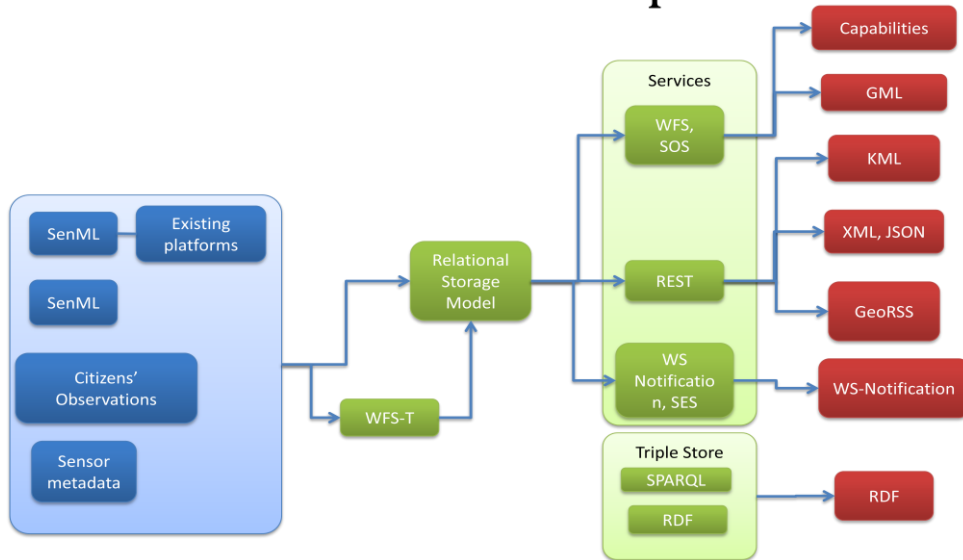


Figure 4-9 Information viewpoint

One of the key cornerstones when designing the CITI-SENSE architecture is the implementation of international open standards where possible. There are many benefits to using open standards, but the main one is that it allows the architecture to be interoperable. As discussed earlier, in the CITI-SENSE architecture data is exchanged between many different components and services. To connect these components and services, it is important that they have a common interface so that data can flow effortlessly.

A comprehensive overview of relevant information model and ontology standards can be found in D7.2.

The openness of a standard plays a vital role as well, as this protects against technology-lock in, where the system is reliable on a particular, often proprietary, technology. Open standards are developed and managed by international communities where people from the public, private and academic sectors work together. In the CITI-SENSE architecture mainly open technology standards developed by the World Wide Web Consortium (W3C)⁴, OASIS⁵ and the Open Geospatial Consortium (OGC)⁶ are used. Many of these standards have been adopted by ISO.

Open Standards that have been adopted in the CITI-SENSE architecture are shown in Table 4-2 Open Standards in the CITI-SENSE architecture:

Table 4-2 Open Standards in the CITI-SENSE architecture

SenML	W3C draft specification that defines media types for representing simple sensor measurements and device parameters.
WFS/WFS-T	OGC interface standard for allowing request for geographical features across the

⁴ www.w3c.org

⁵ www.oasis-open.org

⁶ www.opengeospatial.org

	web using platform-independent calls. WFS-T is a part of the WFS specification which allows for transactions on geographical features, such as inserts, updates and deletes
SOS	OGC standard web service to query real-time sensor data and sensor data time series. SOS is part of a family of the Sensor Web Enablement (SWE) standards.
WS Notification	A group of specifications that allows event-driven programming between web services developed by OASIS.
SES	OGC discussion paper for Sensor Event Service used to provide an publish / subscribe based access to sensor data and measurements. SES is an enhancement of the current Sensor Alert Service (SAS) standard.
GML	OGC standard for expressing geographical features using the XML grammar. GML serves as a modelling language for geographic systems as well as an open interchange format for geographic transactions on the Internet.
KML	OGC standard is an XML notation for expressing geographic annotation and visualization within Internet-based, two-dimensional maps and three-dimensional Earth browsers, originally developed by Keyhole Inc. and extensively used in Google Earth and Maps.
RDF	An official W3C Recommendation for Semantic Web data models

4.5 Engineering Viewpoint

This section describes the architectural elements and underlying enablers and services being used by these services. It identifies the components that need to be developed (engineered), as illustrated in Figure 4-10.

Engineering viewpoint

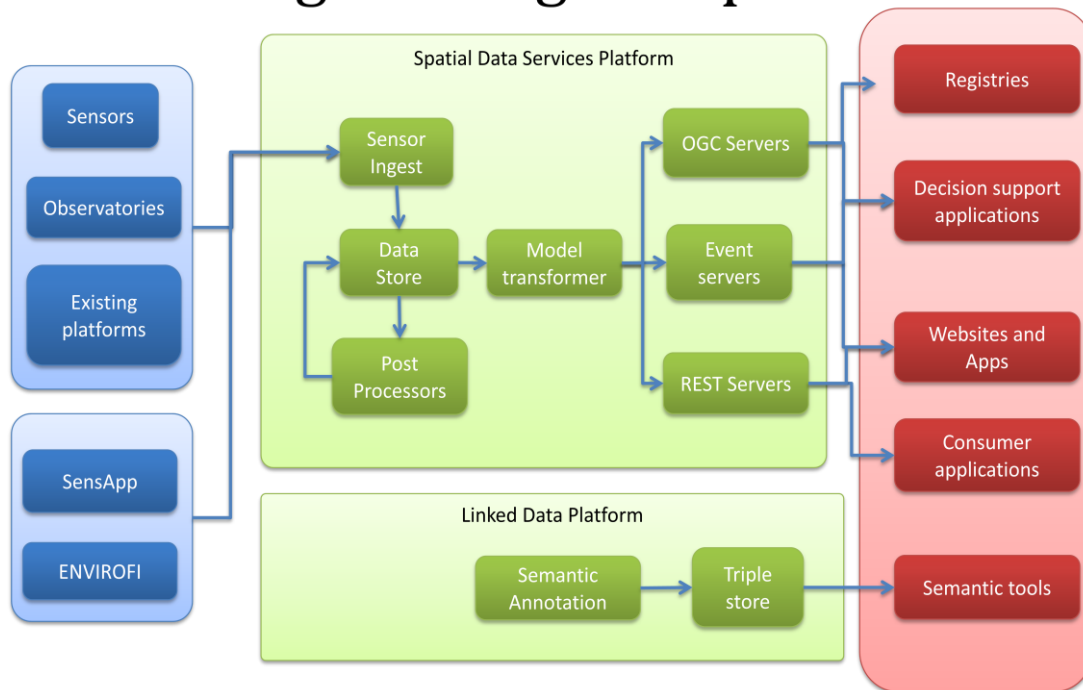


Figure 4-10 Engineering viewpoint

Sensor Ingest

This component allows the ingestion of all types of sensor data into the Spatial and Environmental Data Services Platform.

Data Store

This component stores and manages the data loaded by the Sensor Ingest component. To meet user requirements to query historical data, all data must be stored, managed and retained for a period of time.

Post Processors

In some cases the loaded sensor data needs to be post-processed. This component provides functionality such as data aggregation and harmonisation, but also more complex post-processing processes.

Model transformer

Data is exchanged between a number of distributed federated sensor network providers and a heterogeneous group of data consumers, each with a different set of requirements and data models. The Model Transformer component is responsible for transforming the datasets between the various required data models and web service interfaces.

OGC servers



These include web services that are underpinned by OGC open interface standards, such as WFS and SensorML, or OGC data encoding standards like GML and KML.

Event Servers

This component contains the various event services that will be implemented in the CITI-SENSE architecture. An Event Service provides operations to register sensors at the service application and let clients subscribe for observations available at the service.

REST Servers

Servers that contain web services that provide data using the Representational State Transfer architecture. Data will be encoded using JSON encoding.

Semantic Annotation

Semantic Annotation, or tagging, is about attaching names, attributes, comments, descriptions, etc. to a document or to a selected part in a text. It provides additional information (metadata) about an existing piece of data.

Triple Store

Stores the linked data in a purpose-built database for the storage and retrieval of triples. A triple being a data entity composed of subject-predicate-object, like "Bob is 35" or "Bob knows Fred".

4.6 Technology Viewpoint

This section describes the actual technology and implementation elements related to these services, as shown in Figure 4-11.

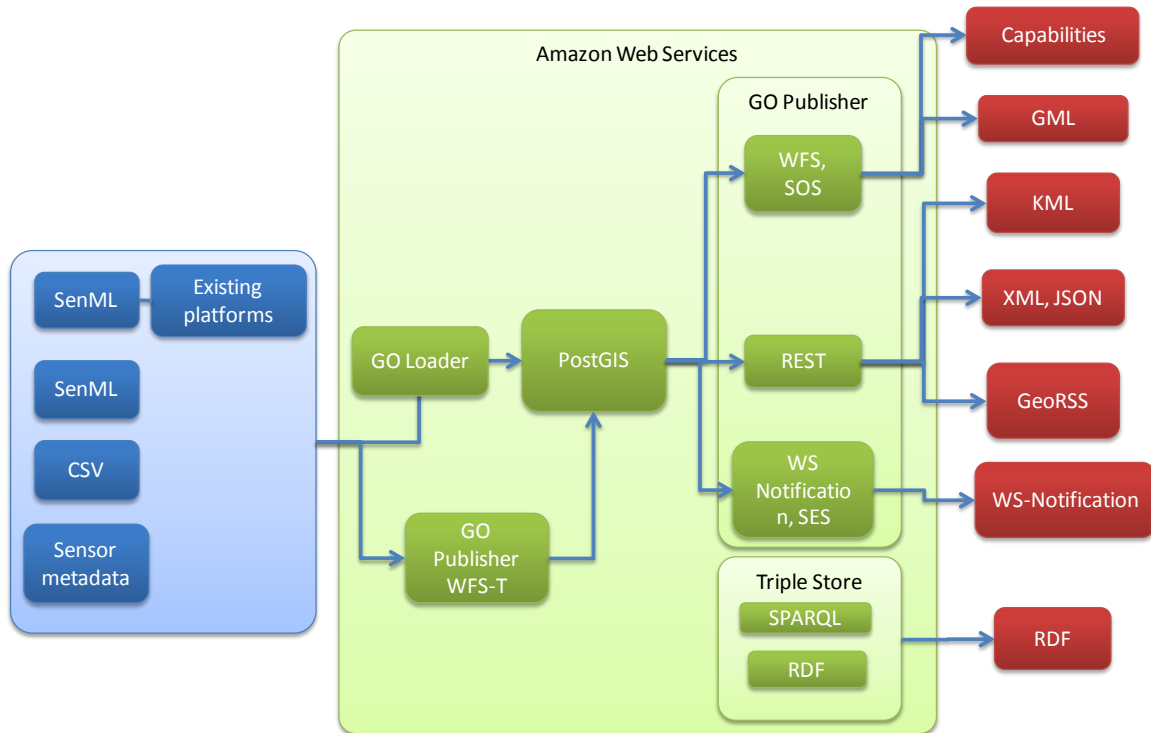


Figure 4-11 Technology viewpoint

In the realisation of the CITI-SENSE architecture a combination of technologies are used. The technologies that have been selected have a proven track record and have been trialed and tested in many implementations across the globe.

Amazon Web Services

Amazon Web Services (AWS) is a collection of remote computing services that together make up a cloud computing platform, offered over the Internet by Amazon.com. The most central and well-known of these services are Amazon EC2 and Amazon S3.

In the CITI-SENSE architecture, the Spatial and Environmental Data Service Platform and Linked Data Platform will be deployed on AWS.

PostGIS

PostGIS is the spatial extension of the open source, object-relational database PostgreSQL. It has been continuously developed by a large community of people over the last 15 years and further developments continue to take place to this day. PostgreSQL, and PostGIS, provide the enterprise class database required by a project like CITI-SENSE where vast amounts of data will be stored and managed.

GO Loader

GO Loader is an off-the-shelf software product developed by Snowflake Software Ltd., which allows for data to be exchanged between any given XML schema to any given relational SQL database schema. The underlying streaming data processing technology enables large amounts of data to be exchanged at high speeds.

GO Loader supports the building and loading of enterprise databases from content delivered in XML and its geographic equivalent GML. Based on Open Standards, GO Loader's 'schema aware' technology means it can automatically adapt itself to any XML dataset. GO Loader has a very fast load time allowing large datasets to be handled with ease.

GO Loader is also a complete data modelling tool with a translator inside. The modelling, translation and data management functionality helps to model, store and use the data in a manner tailored to business needs GO Loader also features GIS integration, multiple database support, easy automation and updating and archiving. GO Loader is used within the WFS-T in order to support the loading of features into the database. It is also used to populate the cloud based relational database.

Additionally, the GO Loader product provides a graphical user interface which can be used to set up a transformation configuration between a source XML schema and target database schema without any coding needed.

GO Publisher

GO Publisher is an off-the-shelf software product developed by Snowflake Software Ltd., which allows for data to be exchanged between any given relational SQL database and any given XML application schema. In this sense, GO Publisher is the mirror image of the GO Loader product.

Just like GO Loader, GO Publisher provides a graphical user interface to quickly build a transformation configuration without any coding required.

After completing the transformation configuration, GO Publisher can be used to build and deploy an ISO compliant Web Feature Service (WFS), or a RESTful web service.

GO Publisher supports various data encodings, including XML, GML, KML and JSON. GO Publisher enables users to publish data within an existing database and make it compliant to a number of given XML or GML application schema. GO Publisher provides a dynamic method of enabling compliance to the demands of local, regional and international standards.

GO Publisher is a suite of products that enable publishing and sharing of database content using Open Standards. Like GO Loader, GO Publisher is 'schema aware', and can publish to any schema. Utilising the schema translation functionality, users are able to maintain one master database but publish to many different exchange models and enable data sharing within numerous wider communities. The GO Publisher software suite provides scalable robust data exchange performance including multiple data publication options, inbuilt schema transformation and validation and business rules.

5 Data and Product Services

Chapter 4 described the overall architecture of the CITI-SENSE Platform using a number of RM-ODP Viewpoints. Each Viewpoint provides a different angle to look at the architecture and highlights specific aspects. This chapter describes these viewpoints in more detail and how they are currently realised in the CITI-SENSE Platform. More specifically, this chapter describes the overall CITI-SENSE Platform using 3 perspectives:

1. **Usage and requirements**, which relates to the RM-ODP Enterprise Viewpoint
2. **Logical service interfaces and information model**, which relates to the RM-ODP Computational and Information Viewpoints;
3. **Implementation technologies**, which relates to the RM-ODP Engineering and Technology Viewpoints.

5.1 Usage and Requirements

Any information system is composed of a range of different, distributed systems and applications. The architecture developed within CITI-SENSE is no different. On the one hand, the project consortium consists of a large number of data service providers with various capabilities that need to be integrated to meet the project's goal to empower citizen-sensing. On the other hand, the project identifies a number of data consumers or users who are interested in using the data provided by the CITI-SENSE Platform for a variety of purposes.

5.1.1 Data Providers

On a high level two distinct groups of data providers can be identified: 1) Federated Sensor Network Providers, and 2) Individual Sensor Providers.

The majority of the project participants that provide data to the CITI-SENSE Platform already have a distributed (often commercial) network of sensors in place. These federated sensor networks consist of a number of physical sensors distributed over their network. Often these sensors are connected to a centralised data hub which periodically collects all sensor data and processes it for further dissemination within that network. Each federated network has its own characteristics and quite often contains some kind of intellectual property owned by the network provider (eg. around sensor hardware or sensor data processing algorithms).

Within the CITI-SENSE project sensor data comes in many different forms. Besides the usual air quality sensor data measurements such as CO₂, NO, PM, etc., some data providers also provide other types of data. Examples of these are, filled in questionnaires, images and audio samples. The CITI-SENSE Platform architecture should be able to process all these types of sensor data.

One of the key objectives of the CITI-SENSE project is the creation of citizens' observatories, which empower (non-expert) citizens to start measuring air quality data themselves by using, mostly, mobile and inexpensive sensors.

5.1.2 Data Consumers

The CITI-SENSE Platform would only be a centralised data storage repository would it not be without its data consumers. Data consumers are people and systems that consume data from the CITI-SENSE Platform to help them with making decisions about their environment.

As described previously in chapter 4, in CITI-SENSE the following groups of data consumers can be identified:

- Government
- Citizen
- SME/Startup business
- Academic/Research

Each of these data consumers has a very distinct set of requirements and have different expectations from the CITI-SENSE Platform. Whilst government and other large organisations would be interested in accessing the data via comprehensive open standard web service, SMEs and startup businesses may be more interested in using more light-weight novel web services to access the data. The CITI-SENSE Platform must be able to meet all these requirements.

In the CITI-SENSE project nine pilot-cities have been identified in which citizens' observatories will be developed. Each of these pilots have a well-defined number of requirements which are specified in the relevant deliverables in Work Packages 2 and 3a/3b.

5.2 Logical Service Interfaces and Information Model

The CITI-SENSE Platform is based around the concepts of a Services Orientated Architecture (SOA) in which discrete pieces of software provide application functionality as services to other applications. Some of the benefits of this decoupling of functionality is that the system is flexible and individual components of the system can be managed, improved or amended without the need to make considerable changes to other parts of the system.

The proposed technical architecture for CITI-SENSE consists of 3 distinct platforms which are developed on different technologies with similar but different end-users in mind.

1. **Sensor Application Platform (SensApp)** - allows for various sensor data provider networks to interact with the CITI-SENSE architecture.
2. **Spatial and Environmental Data Services Platform (SEDS)** - responsible for consuming, storing and publishing environmental data by adopting ISO and OGC Open Standards.
3. **Linked Data Platform** - this platform stores and manages environmental data and makes it available using Linked Data and Semantic Web principles.

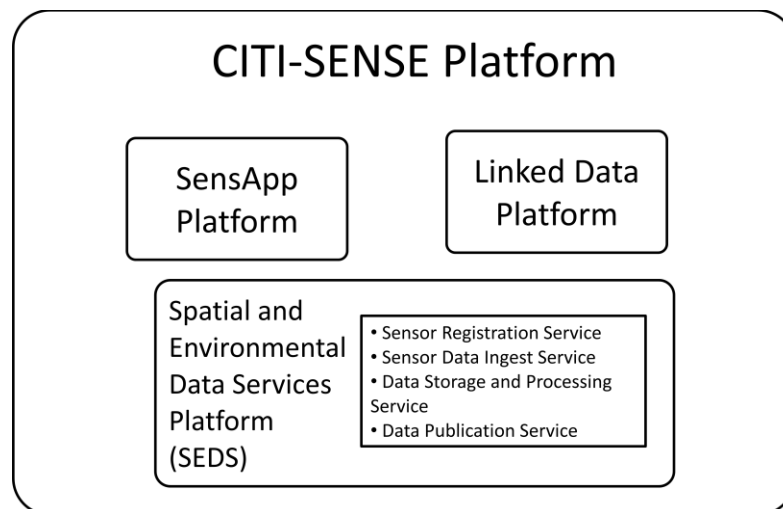


Figure 5-1 The CITI-SENSE Platform and its components

These 3 platforms are integrated to form the overarching "CITI-SENSE Platform". The CITI-SENSE Platform is designed to support the objectives of creating citizen observatories and allows citizens to collect and consume environmental data using a single, centralised technical infrastructure.

Following the Computational Viewpoint discussed in chapter 4, the following distinct functionalities in the CITI-SENSE SEDS Platform can be identified:

1. Sensor Registration Service
2. Sensor Data Ingest Service
3. Data Storage and Processing Service
4. Data Publication Service

In addition to these 4 main functions, more functions can be identified around sensor data acquisition (i.e. making observations) and data consumption, however these functions are discussed in more detail in other project deliverables and are therefore only discussed briefly in this document.

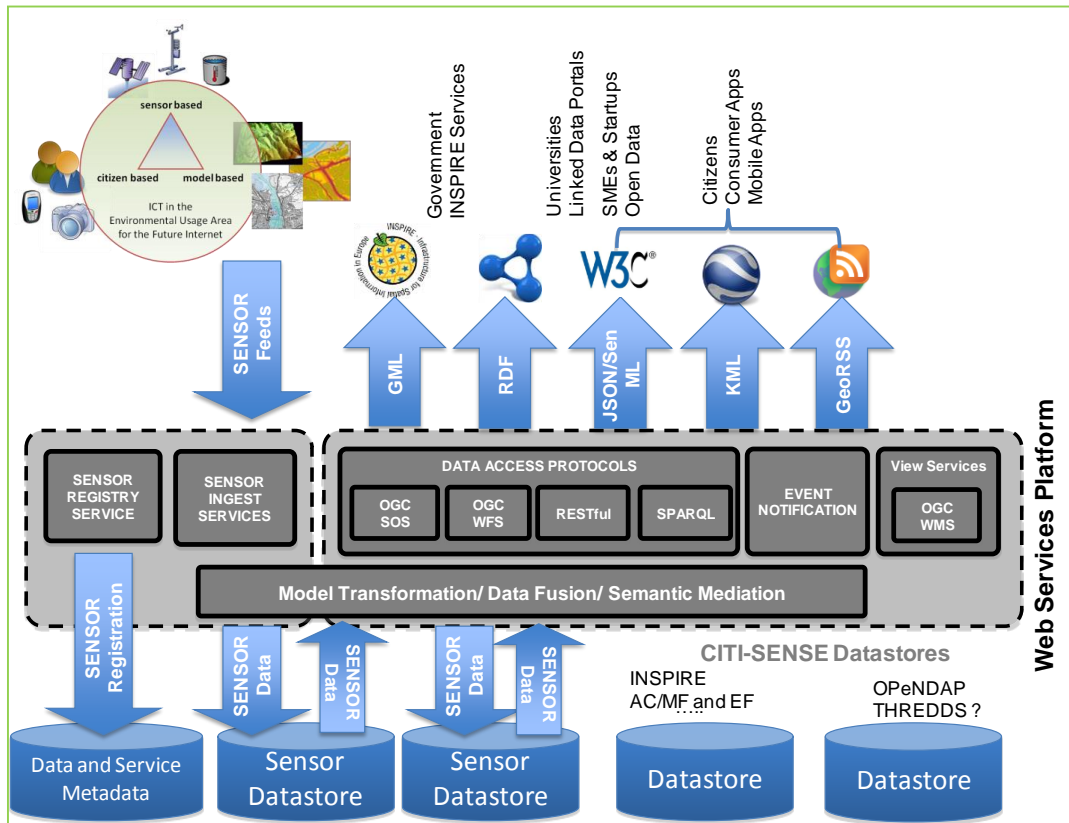


Figure 5-2 Spatial data services

5.2.1 Sensor Registration

A key requirement for sensor data networks to start adding their sensor data into the CITI-SENSE SEDS Platform, is that they register themselves first with the SEDS Platform. This allows users of the CITI-SENSE SEDS Platform to find (discover) which sensors and sensor networks are available in the CITI-SENSE SEDS Platform and which capabilities they offer.

Sensor and sensor network providers must provide all the necessary information required for discovery.

To register a sensor a "Sensor Registration Service" is provided by the CITI-SENSE SEDS Platform. This service provides a single entry point to formally register individual sensors, sensor networks or services. The data that is provided will form the Metadata for that sensor, sensor network or service and will be stored in the CITI-SENSE Platform centralised data repository.

The OGC SensorML standard has been identified for describing the sensors themselves. SensorML provides standard models and an XML encoding for describing sensors and measurement processes. SensorML can be used to describe a wide range of sensors, including both dynamic and stationary platforms and both in-situ and remote sensors.

5.2.2 Sensor Data Ingest

Once a sensor has been successfully registered in the CITI-SENSE SEDS Platform, it can start feeding sensor data into the SEDS Data Repository. The SEDS Data Repository stores and manages the ingested sensor data in a centralised and structured way.

Within the CITI-SENSE project various sensor data providers are active. Many of them already have processes and technology in place that exchange data from their sensor network, however quite often these processes and technology are optimised to work only within this bounded ecosystem. The challenge is to connect these disparate federated sensor network ecosystems to the single, centralised Data Repository.

There are two main mechanisms to ingest data from the sensor data provider networks into the SEDS Data Repository:

1. **Push** - the sensor data provider network actively pushes data to the SEDS Data Repository. This can be done periodically or in real-time.
2. **Pull** - the SEDS Data Repository actively gets data from the sensor data provider network. This can done periodically or in real-time.

Which mechanism, pull or push, to use depends very much on the capabilities that the sensor data provider network can provide. In the CITI-SENSE project both mechanisms proved to be required to deploy.

Table 5-1 Use of push and pull interfaces for observation data providers

Data Provider	Push or Pull	Description
Geotech/AirMonitors	Pull and possibly Push	Provides an external FTP server on which datasets are stored. The Sensor Data Ingest service needs to harvest the data from this FTP server.
AirBase	Pull and possibly Push	Provides a web service (API) that can be called and returns sensor data in CSV
SensApp	Push	Provides a web service that publishes data as SenML XML in a HTTP POST operation.
DNET	Push	Provides a web service (API) that publishes data as CSV or JSON.
JSI	Push	Provides a web service that publishes data.
CVUT	Push	Provides a RESTful API web service that returns data in JSON
U-Hopper	Push & Pull	Provides a web service that publishes data as XML in a HTTP POST operation

One of the key assumptions in the CITI-SENSE SEDS Platform is, that it should adopt Open Standards by default. This allows the CITI-SENSE SEDS Platform to be interoperable, flexible and future-proof. For ingesting sensor data into the SEDS Platform the ISO/OGC Web Feature Standard (WFS) is used. A WFS web service endpoint has been made available that allow sensor data providers to post their datasets against.

Data Models

Data tends to be modelled to meet a specific purpose, such as data ingestion, data storage and data publication. In the CITI-SENSE Platform data plays a vital role for a variety of purposes. Instead of developing a generic, one-size-fits-all, model that meets all the project's requirements is undesirable,

a number specific data models have been developed and data translation processes ensure that data is exchanged between these models. One of the benefits of this de-coupling of individual specific processes is that each process can be highly optimised to meets its purpose. Also, as each process may require a particular expertise and skill-set, multiple teams can work on individual processes.

Data translation processes can be represented by two key processes:

- 1) **Data format conversion** - the encoding of the dataset is translated (eg. XML to SQL, SQL to KML)
- 2) **Data model translation** - converting between conceptual models

Different end-users of the data will require different data models as well. A rich data model containing many feature types and detailed attribution would be suitable for someone wanting to carry out complex analysis, for example. The same information may be of use to a user wanting to use the data for a more general purpose, however the level of detail present may inhibit them from being able to use the data effectively. This more general use would require the data to be in a more simplified form to suit their needs.

Open Standard Interfaces

Following the assumption that the CITI-SENSE SEDS Platform architecture adopts Open Standards, to ingest data into the SEDS Data Repository the SenML standard is implemented.

SenML is a W3C standard data model for encoding measurements and simple metadata about measurements and devices (eg. sensors). The data is structured as a single object (with attributes) that contains an array of entries. Each entry is an object that has attributes, such as a unique identifier for the sensor, the time the measurement was made, and the current value. It does not contain geographic information, such as the geographic location of the device.

5.2.3 Data Storage and Processing

The data provided by the sensor data providers is stored in a central, single Data Repository. In addition to simply storing the data in a structured framework, this component also provides functionality around Semantic Annotation and Data Validation.

Semantic Annotation

The available resources in the network (sensors, data, services, users) can be registered and semantically annotated for publication and easier discovery.

Data Validation

To ensure that only valid data is ingested into the SEDS Data Repository additional data validation takes place.

5.2.4 Data Publishing Services

OGC Web Services

OGC web services will provide access to CITI-SENSE data for clients. This suite of standards is also used by GEOSS and INSPIRE and so by providing access via OGC services. CITI-SENSE data will be easy to integrate into national, European and international data sharing frameworks.

The Web Feature Service and Sensor Observation Service interfaces will be used. These interfaces allow clients to search and retrieve data. They also provide Capabilities documents which describe the service. The capabilities documents can be registered with discovery services such as the INSPIRE and GEOSS registries to make these services discoverable to the communities using those registries.

Content is returned from these services in GML. GML is not itself a format, but is a framework for defining application schemas for particular datasets. These services will therefore return data in a number of application schemas. Existing schemas will be used where possible and new schemas will be defined where necessary. The schema translation capabilities of the CITI-SENSE hub will be used to provide the same data through multiple service endpoints with different application schemas. This will allow harmonisation with other frameworks. For example, CITI-SENSE data can be translated into INSPIRE schemas to provide data which is harmonised with other INSPIRE datasets. It is likely that the CITI-SENSE data will be richer than the INSPIRE schema requires in some cases. In these cases an alternative services with the full attribution of the data can also be provided using an alternative schema.

Lightweight Web Services

The CITI-SENSE Platform will provide a number of lightweight web services suitable for SMEs or citizen users. These will be based on RESTful principles and will provide content in well supported formats including KML, XML and JSON.

REST stands for Representational State Transfer and is a style of software architecture. REST has the concept of linking to resources or data via a simple HTTP url. Services that conform to the REST constraints are referred to as being "RESTful". The simplest example of how REST works can be shown by links on a webpage. If you click on a hyperlink on a webpage it can take you to another webpage where it is essentially returning the HTML code (which can be thought of as data) of the page to be displayed. These links don't have to return just a new web page however but can also take the user to other types of data such as images, XML files or ZIP files for example.

REST can be used to provide end users with a simplified and pre-packaged way of accessing data or resources. REST is a different approach to WFS for example, as where WFS is open ended because of its query language, REST allows users to be sent down a pre-defined pattern of access.

KML is an OGC Open Standard and is well supported by consumer applications such as Google Earth and so will allow citizens to visualise CITI-SENSE data.

KML was developed by Google and submitted to the OGC to be adopted as an implementation standard. KML stands for Keyhole Markup Language and is an XML language focused on the geographic visualisation of data, including annotation of maps and images. Geographic visualisation also incorporates how the user navigates around the data.

XML and JSON are commonly used by web developers. These services can therefore be consumed by SMEs wishing to create mash-ups including CITI-SENSE data.

XML stands for Extensible Markup Language and is a similar markup language to HTML. Although it is similar to HTML it is not a replacement. XML has been designed to transport and store data with a focus on what the data is. XML does not actually perform any function other than to structure, store and transport information.

With XML, tags are created and are not defined in any XML standard and XML has no predefined tags. Because of this XML is very flexible and can be thought of as a software and hardware independent tool for carrying information. XML became a W3C recommendation in 1998.

GML stands for Geography Markup Language. GML is an XML based standard for encoding geographical information developed by the Open Geospatial Consortium. The aim of GML is to allow the exchange of geographical information. It is not designed to be used as a way of storing information. Instead it is aimed at providing a way for people to share information regardless of the particular applications or technology they use. This is known as a heterogeneous environment. In the first instance GML was used to overcome the differences between different GIS applications by providing a neutral file format as an alternative to proprietary formats such as ESRI SHAPE or MapInfo TAB files etc. However, because it is independent of applications there is no reason to assume that data exchanged in GML is being exchanged between GIS - it could be being moved between databases or other types of application. GML therefore has a wider application than just GIS data transfer.

JSON stands for Java Script Object Notation and is a lightweight text-data interchange format. JSON is language independent and self describing which makes it easy to understand. Although JSON uses JavaScript syntax for describing data objects, it is still language and platform independent. JSON parsers and JSON libraries exist for many different programming languages.

GeoRSS stands for Geographic Rich Site Summary. As RSS and Atom become more prevalent as a way to publish and share information, it becomes increasingly important that location is described in an interoperable manner so that applications can request, aggregate, share and map geographically tagged feeds. GeoRSS was designed as a lightweight, community driven way to extend existing feeds with geographic information.

There are currently two encodings of GeoRSS, Simple and GML:

- GeoRSS-Simple is meant as a very lightweight format that developers and users can quickly and easily add to their existing feeds with little effort. It supports basic geometries (point, line, box, polygon) and covers the typical use cases when encoding locations.
- GeoRSS GML is a formal GML Application Profile, and supports a greater range of features, notably coordinate reference systems other than WGS-84 latitude/longitude.
-

Both formats are designed for use with Atom 1.0, RSS 2.0 and RSS 1.0, although it can be used just as easily in non-RSS XML encodings.

Semantic Services

The CITI-SENSE Platform will provide a SPARQL endpoint serving RDF. This can be used to query and link the CITI-SENSE data to other data in the semantic web. RDF stands for Resource Description Framework. RDF is a standard model for data interchange on the Web. RDF has features that facilitate data merging even if the underlying schemas differ, and it specifically supports the evolution of schemas over time without requiring all the data consumers to be changed.

RDF extends the linking structure of the Web to use URIs to name the relationship between things as well as the two ends of the link (this is usually referred to as a “triple”). Using this simple model, it allows structured and semi-structured data to be mixed, exposed, and shared across different applications. This linking structure forms a directed, labelled graph, where the edges represent the named link between two resources, represented by the graph nodes. This graph view is the easiest possible mental model for RDF and is often used in easy-to-understand visual explanations.

5.3 Implementation

5.3.1 Cloud deployment

The CITI-SENSE environment consists of a distributed network of systems and applications. To manage this network effectively and remove any dependencies on individual project participants, the CITI-SENSE Spatial and Environment Data Services (SEDS) Platform is implemented in the Cloud (i.e. the Internet).

Instead of developing a private CITI-SENSE cloud, the services from the commercial cloud service provider Amazon is used. In particular, the CITI-SENSE SEDS Platform uses Amazon Web Services (AWS) to deploy the various CITI-SENSE SEDS Platform components described in chapter 5.2. AWS provides a collection of web services that can be combined to make up a cloud computing platform.

For the deployment of the CITI-SENSE SEDS Platform the following AWS services are used:

- **Amazon EC2** (Amazon Elastic Compute Cloud), which provides virtual computers on which the CITI-SENSE software components and web services are deployed and managed.
- **Amazon S3** (Simple Storage Service), which provides data storage capacity.

5.3.2 Sensor Registration

The Sensor Registration Service implements the ISO/OGC standard SensorML. SensorML is an approved Open Geospatial Consortium (OGC) standard and provides standard models and an XML encoding for describing sensors and measurement processes.

The Sensor Registration Service can be accessed via a Web Feature Service (WFS) web service endpoint. The WFS service allows data providers to post an XML dataset which contains the sensor registration information.

WFS is an approved OGC web interface standard and provides an interface allowing requests for geographical features across the web using platform-independent calls. The decision for choosing an open standard from the geography domain (OGC) is because of the importance of the geographic location of a sensor. The WFS standard is optimised for processing geospatial data and is therefore the ideal candidate for this.

The Sensor Registration Service will be available as web service on the CITI-SENSE SEDS Platform on the Amazon cloud.

5.3.3 Sensor Data Ingest

For ingesting sensor data into the CITI-SENSE SEDS Platform a WFS web service will be made available. The WFS accepts HTTP Post messages. The WFS standard not only allows for data to be queried and returned via an open web interface, but also supports transactional operations such as inserts, updates and deletes (WFS-T).

For ingesting data from sensor data providers into the SEDS Data Repository a WFS-T web service has been made available. By using a HTTP Post operation a XML document can be posted to the WFS-T web service. The web service will validate the request and load the dataset into the Data Repository, as illustrated by Figure 5-3.

The WFS-T web service has been developed using Snowflake Software's GO Publisher software product.

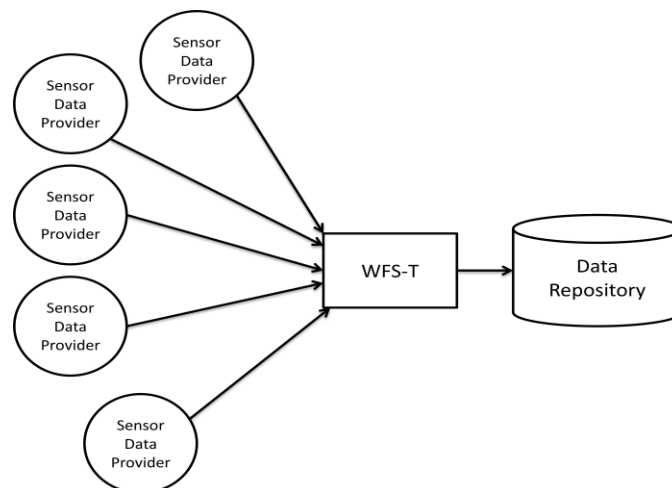


Figure 5-3 WFS Data ingestion

SenML in the CITI-SENSE Platform

The SenML model has been adopted in the SensApp Platform. SensApp is an open-source service-based application developed by SINTEF and is used to store and exploit data collected by the Internet of Things (IoT). It supports the following actions:

- Application can registers sensors, stores their data.
- SensApp notifies clients with newly arrived data.
- A third-party application is provided to visualise data.

SensApp offers web services to upload, from smart phones or other data gathering mechanisms, sensor data or extrapolated data to a repository. The data gathered by SensApp will be made available to the CITI-SENSE Platform using SenML.

Various sensor data providers in the project use the SensApp platform as a first stage data hub. In this sense the SensApp platform can be regarded as another federated sensor data provider network.

The data exchange between the SensApp Platform and the SEDS Platform adopts the SenML standard. As the SEDS Platform is spatially enabled, for the non-spatial SenML standard a GML

application schema has been developed. With this spatial GML application schema in place, spatial data can be exchanged between both the SensApp and SEDS Platforms.

Deployment in the CITI-SENSE SEDS Platform

To cope with the different capabilities of the sensor data providers (push or pull), specific web services are created and deployed on the Amazon cloud.

5.3.4 Data Storage and Processing

The SEDS Data Storage component is implemented as a relational SQL database. The SQL query interface will be a common interface at the back end of the majority of the CITI-SENSE services. The SQL interface will not be available to external applications and external applications will communicate with the CITI-SENSE platform via open standard web services.

In addition to the SQL database a Linked Data Triplestore database will be implemented which will contain a copy of the CITI-SENSE data as semantic tuples. This store will be updated from the SQL database and will be used to support the SPARQL web service.

Database technology

The free and open source database technology PostgreSQL is implemented to realise the SEDS Data Repository. PostgreSQL is cross-platform and supports many different operating systems. Currently PostgreSQL is widely used across the world and underpins some large IT infrastructures.

For storing geographical data, the PostgreSQL database is extended with the PostGIS add-on. PostGIS is an open source software program that adds support for geographic objects to the PostgreSQL database. PostGIS follows the Simple Features for SQL specification from the Open Geospatial Consortium⁷.

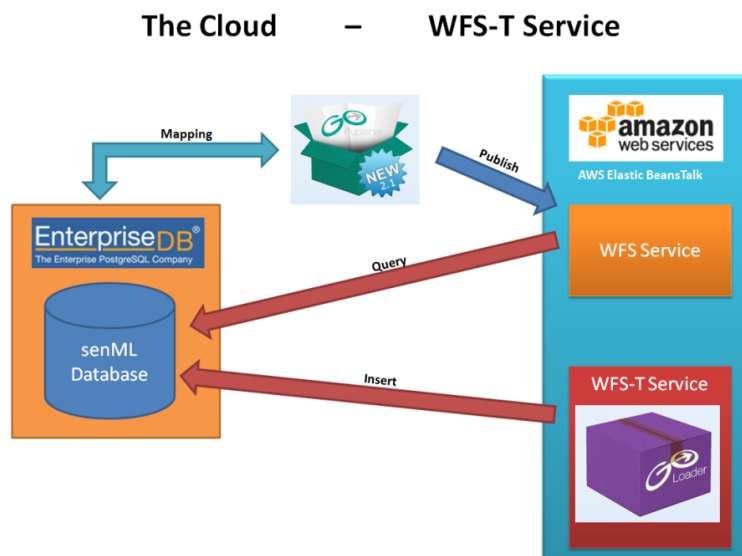


Figure 5-4 WFS Cloud services support

⁷ <http://www.opengeospatial.org/standards/sfs>

For quickly deploying the PostgreSQL database on the Amazon cloud, it has been done initial experiments with a service provided by the commercial company EnterpriseDB, "Postgres Plus Cloud Database"⁸ as illustrated in Figure 5-4. This service provisions a cloud based, elastic database instance of PostgreSQL on the Amazon cloud, using EC2 to deploy the database services and S3 for the data storage. The web-based point&click interface allows for rapid database installation and configuration.

5.3.5 Data Publication

The CITI-SENSE SEDS Platform enables access to data stored in the SEDS Data Repository via web services.

These web services include a mix of comprehensive WFS and light-weight RESTful services, which supports a variety of data encodings such as XML, GML, KML and JSON.

For creating and deploying the WFS and RESTful web services the software product GO Publisher for Snowflake Software is used.

The GO Publisher product allows for rapid configuration and deployment of WFS and REST web services via a graphical user interface, as shown in Figure 5-5. In GO Publisher a transformation is configured between the database schema in the Data Repository and any of the XML application schemas against which the sensor data needs to be published. An example of an XML application schema are the various EU INSPIRE Themes.

The web services are deployed on the Amazon Cloud and can be accessed via a WFS endpoint.

⁸ <http://www.enterprisedb.com/products-services-training/products-overview/postgres-plus-cloud-database>

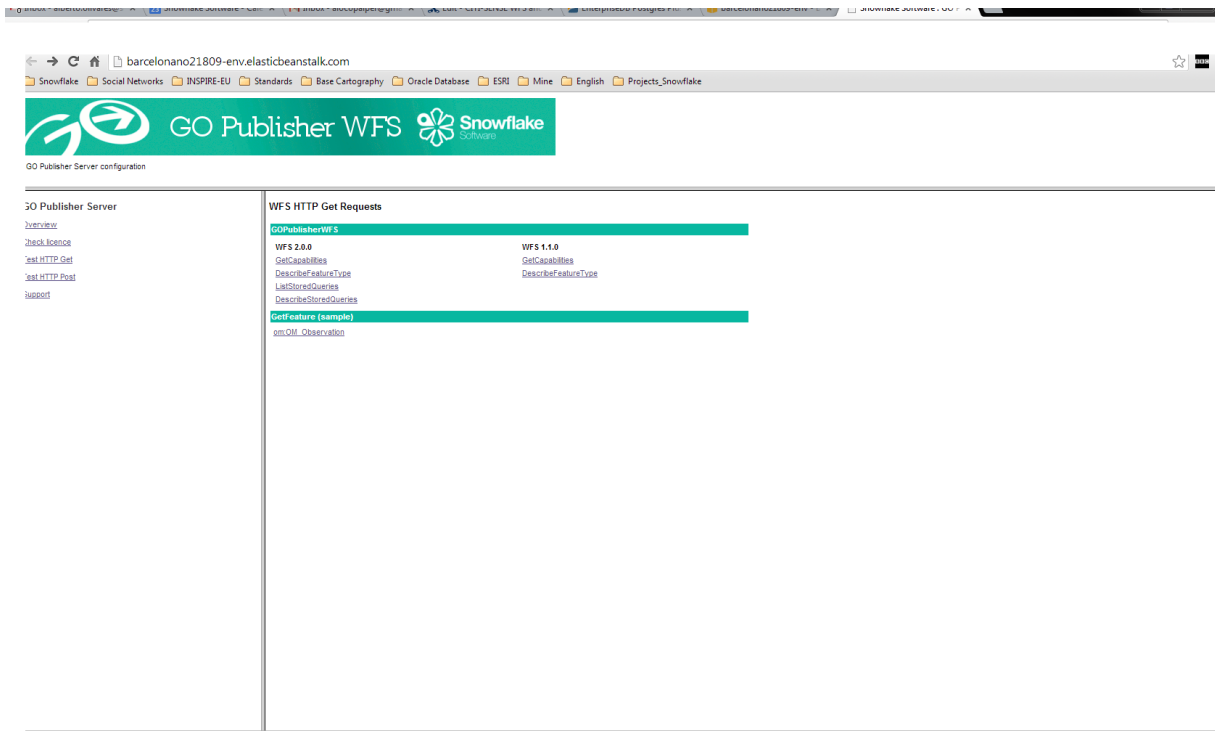


Figure 5-5 An example of the GO Publisher WFS landing page

6 Sensor services

This chapter describes the sensor services supported by the CITI-SENSE architecture and platform.

6.1 Usage and requirements

This section describes the usage context and use cases for these services. The CITI-SENSE platform aims to support the data storage and handling of sensor data for a number of independent sensor platform providers. In the CITI-SENSE project there is a number of sensor platform providers that already have their own sensor data management solutions for providing observations. The CITI-SENSE architecture is thus aiming at the support for common storage services for all of these through a possibility for both a push and a pull interface for the CITI-SENSE storage services. For sensor observation situations where no pre-existing sensor platform exist, in particular for mobile smart phone connected sensors, the CITI-SENSE project is offering the usage of the SensApp platform.

6.1.1 Sensors

Existing sensor networks already contain capabilities for storing and post-processing sensor data. These capabilities continue to operate as a federated capability within the CITI-SENSE architecture. Sensors which are not part of such a network will connect individually to CITI-SENSE.

CITI-SENSE will support two modes of data transmission from sensors.

1. Sensors can actively upload new data either periodically or as it becomes available, or
2. CITI-SENSE can periodically poll the sensors to check for new data.

Data will be validated on arrival at the CITI-SENSE hub to ensure that it forms a coherent set of information. Validation is an important aspect of the CITI-SENSE functionality since it provides a level of quality control on incoming data and thus ensures that the data held by CITI-SENSE is coherent. Without this stage in the process the CITI-SENSE data could become self-contradictory, which could render it worthless for analysis.

Following validation, valid data will be stored within the CITI-SENSE data storage services.

6.2 Logical service interfaces and information model

This section describes the service contracts and interfaces for the services offered by the SensApp platform. SensApp is a platform for sensor and sensor data handling. It is an open-source, service based application for sensor registration, data collection, storage and visualization of sensor data, developed at SINTEF⁹.

In the context of CITI-SENSE the SensApp framework can be useful in particular for the initial handling of sensor data for mobile applications for Android phones.

⁹ <http://sensapp.org/>

SensApp includes a mobile application called SensorLog which is responsible for sensor communication, data collection and passing data about the sensors and measurements to a client system, called SensApp-Android. The client is running as a separate mobile application, displaying collected raw data from the local database, communicating with the SensApp server and uploading data.

Furthermore SensApp includes a web-based administration interface for sensor manipulation and data access. It provides graphs to visualize stored sensor data. The SensApp server, the client (SensApp-Android) and the mobile application (Sensor- Log) are described below.

SensApp Server

The SensApp server is responsible for retrieving and storing sensor data, sending notifications as soon as new data is available and allowing data access to clients and third party applications, for example to visualize the sensor data.

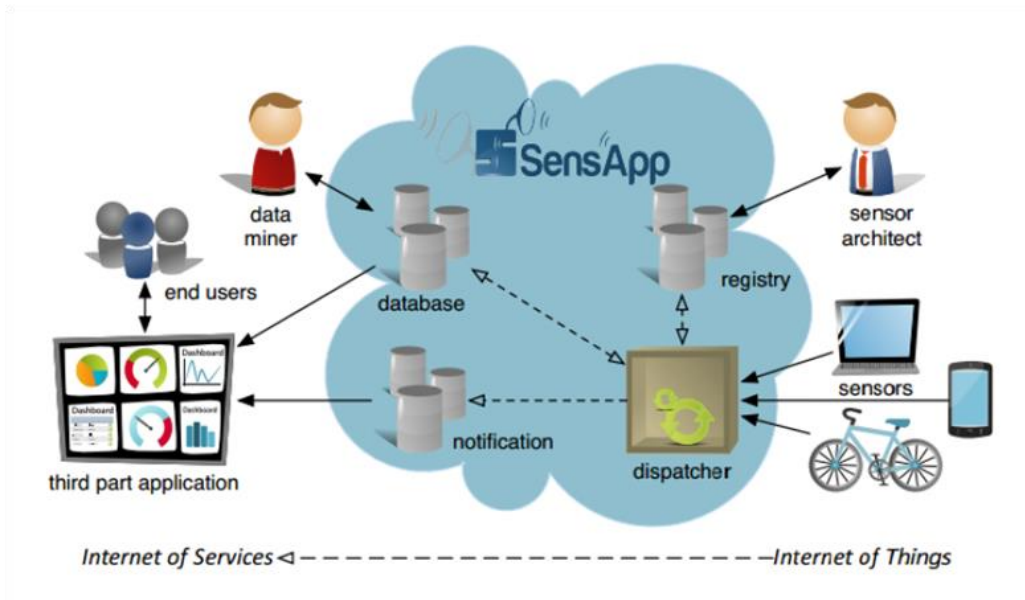


Figure 6-1 SensApp platform architecture

The Figure 6-1 shows an overview of the server architecture and its cooperation with different user roles, mobile devices and sensors as well as other applications. The user roles are a sensor architect, defining the sensors, a data miner, responsible for analysis and final conclusion about the data and the end users, accessing analyzed and visualized data and using it.

Furthermore, the figure shows how SensApp makes the Internet of Things available through the Internet of Services, meaning that all the sensors and devices are handled through services, and in particular the following four REST services:

- Registry
- Database
- Notification
- Dispatcher

These are further described here: .

- The **Registry Service** stores information of each sensor. This information contains an ID, a description and a schema for the data storage, for example the data type of the stored values. Additional information, like a location or an update frequency can be stored using user defined key-value pairs
- The **Database Service** is used to store collected sensor data in a database, (currently MongoDB is being used). Queries to transfer stored data to the data miner are provided.
- The **Notification Service** can send notifications if new or relevant data is available. This can be used, for example, by third party visualization software to show updated graphs and statistics to the user.

The **Dispatcher Service** receives data from the sensors and according to its registry entry, the data is stored in the database and a notification is triggered, if a related notification topic is registered.

Usually all SensApp services are deployed on one server, but this is not necessary, actually the architecture is distributed and therefore can be deployed on several servers. An example distributed deployment is described in the paper "SENSAPP as a Reference Platform to Support Cloud Experiments: From the Internet of Things to the Internet of Services" .

Furthermore SensApp can easily be deployed on a cloud system, because of its REST architecture.

Sensor Markup Language (SenML) was designed by the IETF so that processors with very limited capabilities could easily encode a sensor measurement into the media type, while at the same time a server parsing the data could relatively efficiently collect a large number of sensor measurements.

The markup language can be used for a variety of data flow models, most notably data feeds pushed from a sensor to a collector, and the web resource model where the sensor is requested as a resource representation (GET /sensor/temperature). SenML is defined by a data model for measurements and simple metadata about measurements and devices. The data is structured as a single object (with attributes) that contains an array of entries. Each entry is an object that has attributes such as a unique identifier for the sensor, the time the measurement was made, and the current value.

SensApp-Android

An Android application framework has been made for SensApp, as a generic application that can be further customised.

The main task fulfilled by this application is to push local data to the SensApp server. Beside that the SensApp client can display locally stored data raw or in a graphical way as shown in Figure 6-2.

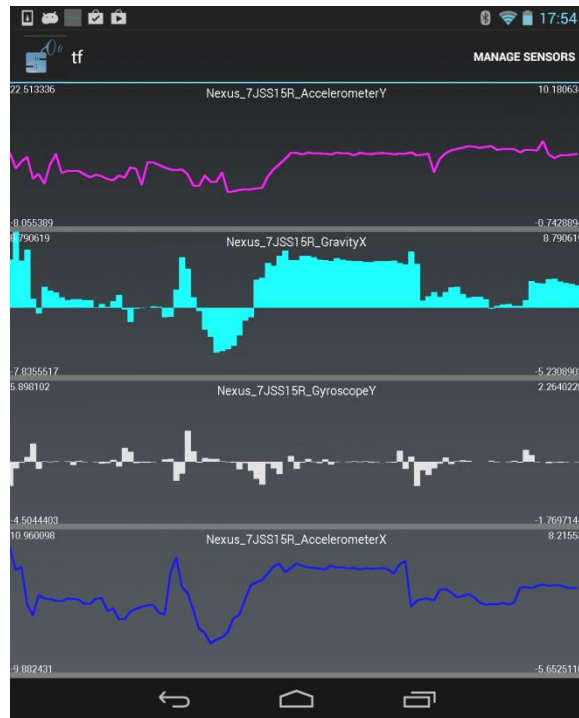


Figure 6-2 Sensor local app interface

Uploading data to the server can be configured to a periodic upload or depending on the amount of collected data.

The data sent from the client to the server is structured using a JSON implementation of SenML, and can be transmitted using HTTP or websockets.

SensorLog

SensorLog is a mobile application discovering and listing all available sensors, including build-in sensors of the device, for example an accelerometer, magnetic field or gravity sensor, and add-on sensors like a UV sensor or a weather meter.

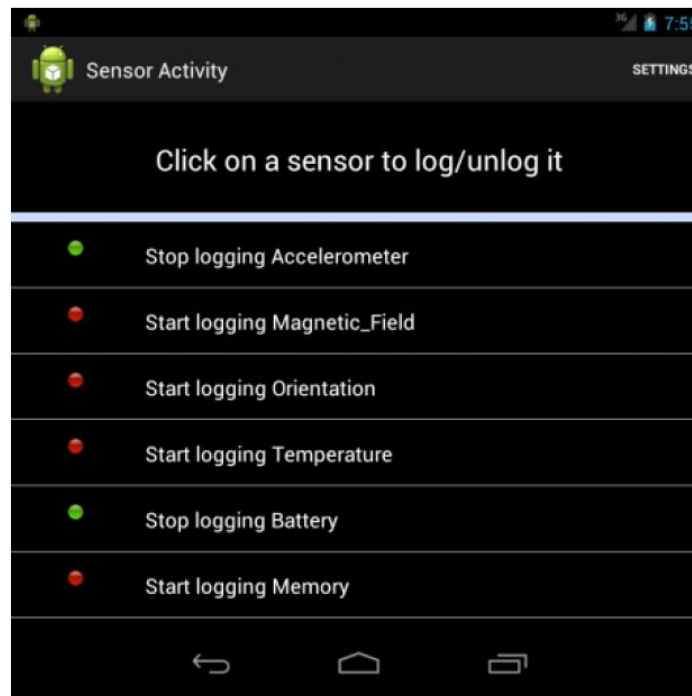


Figure 6-3 Sensor log interface

The user can select several sensors to be logged, as shown in Figure 6-3, then SensorLog gathers data from all selected sensors and pushes the data directly into the SensApp-Android application, which is responsible for server communication and data upload.

There are more and more Bluetooth sensors available, in order to extend the range of sensed data these sensors need to be supported by the sensor data framework.



Figure 6-4 Bluetooth Smart and Bluetooth Smart Ready

The latest version of the Bluetooth technology is 4.0, which introduced Bluetooth Low Energy (BLE), a protocol for better power optimization. As shown in Figure 6-4 there are two different implementations for Bluetooth devices using Bluetooth 4.0, they are either Bluetooth Smart Ready devices or Bluetooth Smart devices. A Bluetooth Smart Ready device supports a dual mode, meaning it is capable of BLE as well as classic Bluetooth functionality. On the contrary, a Bluetooth Smart device is a single mode device and therefore just BLE is supported.

As there are these two different ways for including Bluetooth in a device, the producers

of devices including Bluetooth have the capability to either use the full version which is able to connect to older Bluetooth versions as well, or to implement BLE which is cheaper, smaller and therefore leads to cheaper and smaller devices. Therefore, and as BLE is the current standard and many add-on sensors are Bluetooth Smart devices, BLE support is more important. However, in order to support older sensors as well, both, classical Bluetooth and BLE, has been implemented – choosing devices that also will be used in the CITI-SENSE Vitoria pilot case.

It is not possible to implement Bluetooth support in such a general way that the system can handle all devices in a plug-and-play manner, as they all communicate using different commands. The way the connection is set up and data is sent, is the same for all BLE or classical Bluetooth devices, so this can be generalized within SensorLog. In order to add support for a specific device, the communication needs to be implemented separately. For test reasons and to serve as an example for application developers, the support for two sensors, one classical Bluetooth sensor and one Bluetooth Smart sensor will be implemented in SensorLog. Both devices are introduced in the following.



Figure 6-5 BLE Device – UV Sensor

The BLE device, which is used as the example and, is a SunBit UV sensor, shown in Figure 6-5. It is capable of measuring the UV radiation, the acceleration and the magnetic field. Thereby it measures UVA and UVB separately.

The sensor includes local storage, enabling online logging for up to one week, depending on which values are measured. This allows the user to collect UV data continuously and, for example, upload it just once a week or once a day. Therefore the user doesn't necessarily need to use a mobile device but can upload the data at home using a Bluetooth compatible computer.

The SunBit UV sensor is a Bluetooth Smart device where the collecting device (smartphone, tablet or computer) needs to support Bluetooth 4.0.



Figure 6-6 BL Device – Weather Meter

As the test device for classical Bluetooth, a Kestrel 4000 Weather Meter (from the Vitoria pilot case) is used, as shown in Figure 6-6.

The weather meter can be used to measure ten different weather related values, such as: temperature, wind speed, relative humidity and barometric pressure. These measurements can be stored locally (online logging) and accessed via Bluetooth by a mobile device or computer. In addition to the measurements, the device provides graphing functions and minimum, maximum and average values for the measurements

6.2.1 Sensor Inputs

Sensor input will be transmitted as XML in the SenML schema, or as CSV. HTTP will be used as the transport. This provides an open and well defined interface which allows any sensor provider to connect their sensor to the CITI-SENSE hub.

Observational data (eg. filled in questionnaires) will be transmitted as XML.

6.3 Implementation technologies

This section describes the actual technology and implementation elements related to these services.

SensApp has been extended with a service called "encoder", which enables to push and to retrieve data stored into/from SensApp in CSV. In the first case, the service is then placed in front of the dispatcher in order to retrieve the CSV data before transforming it into SenML to be pushed in

SensApp. In the other case, it is placed in front of the Datastore in order to retrieve data from the database before converting it into CSV format.

6.3.1 Retrieving data in CSV format

This service enables SensApp users to retrieve data from several sensors in CSV format. Users can specify through an HTTP POST request the set of sensors they are interested in as well as the character to be used as a separator between values. Listing 1 describes the typical content of a request in the Barcelona expoapp case. The objective of this request is to retrieve the GPS latitude, longitude, and the accelerometer values from a specific participant, each column being separated by a comma.

Listing 1. Example of content for a request to the encoder

```
{
  "datasets" : [{
    "url" : http:// SENSAPP\_URL /sensapp/databases/raw/data/ACC-z-ParticipantID001
  }, {
    "url" : http:// SENSAPP\_URL /sensapp/databases/raw/data/ACC-x-ParticipantID001
  }, {
    "url" : http:// SENSAPP\_URL /sensapp/databases/raw/data/ACC-y-ParticipantID001
  }, {
    "url" : http:// SENSAPP\_URL /sensapp/databases/raw/data/GPS-latitude-ParticipantID001
  }, {
    "url" : http://SENSAPP\_URL/sensapp/databases/raw/data/GPS-longitude-ParticipantID001
  }
  ],
  "separator" : ",",
}
```

The result of such request consists in data formatted in CSV and organized as described in Table 1. Values are ordered and grouped according to their associated timestamp. Each line is composed of the value that has been collected at a specific time from each of the sensors specified in the request, each value being separated by a comma. The service represents the absence of measurement from a sensor within a line by a "-" in the CSV.

In order to generate such data, the service proceeds as follow:

1. Retrieve data from each sensors (which are already ordered on the basis of their timestamps)
2. Group all values with the same timestamp into an ordered set on the basis of the order of the sensors within the request.
3. Each group represents a line in the CSV and is added at the end of the CSV.

Table 6-1 Example of data aggregation

Time stamp	GPS lat	GPS long	ACC x	ACC y	ACC z
0:00:01	Lat	Long	x	y	z

0:00:02	Lat	Long	x	y	z
0:00:03	lat	Long	x	y	z

In order to save energy and network bandwidth of a sensor that collects measurements with a high velocity, a classical strategy consists in reducing the frequency of requests to store these measurements into SensApp. This can be done by pushing at one time several measurements. As a consequence, several values from one sensor can have the same timestamp. The converter manages such chunk of data by approximating the timestamp of each value within a chunk as follow.

First it computes the time elapsed between the storage of the chunk and of the next measurement. This duration is then divided by the number of measurements in the chunk:

$$(Timestamp\ next\ value - timestamp\ chunk) / number\ of\ values\ in\ the\ chunk$$

On the basis of the order of the values in the chunk, each of them is then associated to one of the computed timestamp.

The SensApp Administration web site has been extended in order to enable downloading the data of a composite sensor as a CSV file, as shown in Figure 6-7. This is achieved by generating and sending a request as described in Listing 1 to SensApp on the basis of the sensors that form the composite.

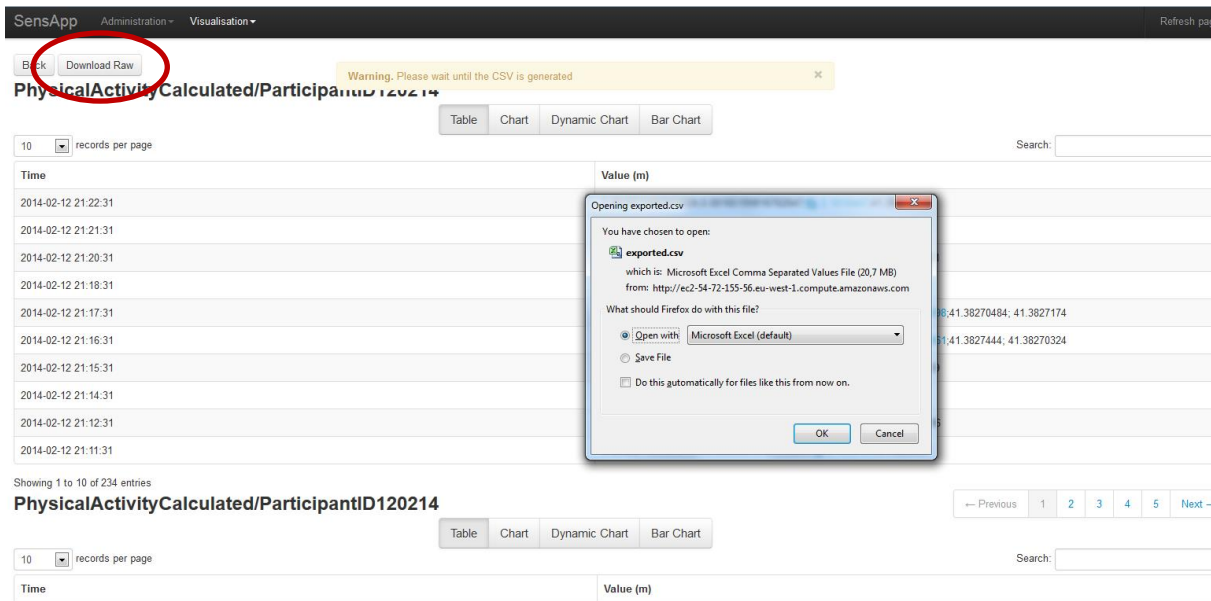


Figure 6-7 CSV export

As explained before, the encoder can also be used in order to push CSV data into SensApp through an HTTP POST request. This service operates as follow:

1. Extract from the first row of the CSV the list of sensors on which the data in the CSV should be dispatched
2. For each value from each line, associate it to the relevant sensor, generate, and send a standard SenML request to the dispatcher

In order to secure communications between sensors in SensApp an encryption mechanism has been implemented and extends this service. The data is encrypted on the client side and decrypted on the SensApp side using the Software developed by Ateknea, which takes the form of a Java jar file. This executable jar typically consumes a file whose content is encrypted and produces a decrypted file. The code below presents how the encoder service uses it.

```
var out = new java.io.PrintWriter("tmp.csv")
val splitted=data.split("::")
out.print(splitted(1))
out.close()
val cmd="java -jar /home/ubuntu/GPSDecrypter.jar tmp.csv"
Process(cmd).!
val r= new ReadData("/home/ubuntu/tmp_decrypted.csv")
val t:Array[String]=r.read2JSON(splitted(0))
```

6.3.2 On the relationship of SenML and SensorML

The SenML sensor representation format is aimed at being a light weight format, while SensorML from the OGC standard suite on Sensor Web Enablement (SWE), with services like the Sensor Observation Service (SOS), is more comprehensive. It is thus relevant to analyse the relationship between SenML and SensorML representations, as both will be relevant in the CITI-SENSE context.

In order to provide access to a SenML sensor through an SOS web service we need to map SenML onto SensorML. As an example of a SOS web service we have looked at the one being provided by the open source initiative 52North¹⁰.

An SOS is characterized by:

- **Procedure:** Description of the sensor
- **Phenomenon (observable property):** The physical phenomenon that is being observed/measured (e.g. temperature). This is the "input" to the sensor.
- **Feature (of interest):** The real world object that is being measured/observed. (That produces the phenomenon.)
- **Offering:** The measurements/observations that a sensor can provide. The capabilities of the sensor (in time and space). This is the "output" from the sensor.
- **Observation:** The actual measure or observation, providing a value for a specific phenomenon, feature and offering, from a specific sensor at a specific time.

In 52North, a sensor is defined by a SensorML file while its data is stored in a PostGIS¹¹ database. The information in the SensorML definition is used to create queries to the database. But this also means that the full definition of the sensor (and its data) is not given in the SensorML and some of the information defining the sensor is stored in, and has to be queried in, the database.

¹⁰ <http://52north.org/>

¹¹ <http://www.postgis.org/>

In this PostGIS database the five characteristics listed above have the following attributes (attributes in *italic* are mandatory):

- **procedure**
 - *procedure_id* (Sensor ID given as URN)
 - **description_url** (URL to sensor definition (SensorML) relative to server URL)
 - **description_type** (reference to SensorML version)
- **phenomenon**
 - *phenomenon_id* (as URN)
 - **phenomenon_description**
 - **unit**
 - *value_type* ('numericType' or 'textType')
- **feature_of_interest**
 - *feature_of_interest_id*
 - *feature_of_interest_name*
 - **feature_of_interest_description**
 - *geom* (location)
 - *feature_type*
 - *schema_link* (not used; dummy data)
- **offering**
 - *offering_id*
 - **offering_name**
- **observation**
 - *observation_id* (automatically generated)
 - *time_stamp*
 - **text_value**
 - **numeric_value**
 - **mime_type**
 -

The SensorML file defining an SOS refer to **procedure id**, **feature of interest id**, **phenomenon id** and **offering id**, and replicates the information in **geom**, **offering name** and **unit**. In general there are many-to-many relationships between procedure, phenomenon, feature of interest and offering, but for simplicity we will in the following treat them as pairwise one-to-one.

In SenML, the meta data of a sensor is given, in JSON, as:

```
{
  "id": "<Sensor ID>",
```

```
"descr": "<Sensor description>",
"backend": {
  "kind": "<Backend: 'raw' or 'rrdb'>",
  "descriptor": "<Data description URL>",
  "dataset": "<Dataset URL>"
},
"creation_date": <Date as seconds after EPOCH as integer>,
"infos": {
  "tags": {"<Tag 1>": "<Value 1>", ..., "<Tag n>": "<Value n>"},
  "update_time": <Update time in seconds as integer>,
  "loc": { "longitude": <Longitude as float>,
           "latitude": <Latitude as float> }
}
}
```

In the following we assume "kind": "raw". The tag elements as well as update_time and loc are optional.

In addition, the data description is given by:

```
{
  "sensor": "<Sensor ID>",
  "schema": "<Data template for data base: 'Numerical' or ...>",
  "size": <Number of records as integer>,
  "data_ink": <Dataset URL>
}
```

The observation data are given as:

```
{
  "bn": "<Sensor ID>",
  "bt": <Base time in seconds after EPOCH as integer>,
  "e": [{ "u": "<Unit>", "v": <Value as float>,
           "t": <Time relative to base time in seconds as integer> }, ...]
}
```

Value "v" might instead be Boolean value "bv" or string value "sv". (A measurement can also use name "n" if the sensor with <Sensor ID> can give different kinds of measurements at the same time (e.g. temperature and air pressure). The name of each "sub-sensor" is then base name "bn" appended by name "n". But "n" is only used by composite sensors; for simplicity we disregard this option. In SenML it's also possible to specify a base unit "bu" that is applied to measurements where no unit "u" is given, but this option is not used by SensApp.)

As can be seen, SenML does not offer the distinction between phenomenon, feature and offering that SOS has. When we map the data and meta data of SenML described sensors to the data model of SensorML (i.e. as SOS/52North) it is then clear that we do not get enough information from SenML to fill all the fields of SensorML (SOS/52North). When faced with these gaps we have three alternatives: 1) provide additional information (from outside of the SensApp definitions), 2) generate the data, or 3) leave fields blank. (We can however note that the provided information can be provided within the SenML definitions by use of the "tags".)

The following mapping is proposed:

- **procedure**
 - *procedure_id* ← id converted to URN
 - *description_url* ← id converted to URL
 - *description_type* ← text/xml; subtype="SensorML/1.0.1"
- **phenomenon**
 - *phenomenon_id* ← provided or generated
 - *phenomenon_description* ← provided or blank
 - *unit* ← u
 - *value_type* ← schema converted to numericType or textType
- **feature_of_interest**
 - *feature_of_interest_id* ← generated
 - *feature_of_interest_name* ← provided or descr
 - *feature_of_interest_description* ← provided or blank
 - *geom* ← loc, provided or dummy data
 - *feature_type* ← sa:SamplingPoint
 - *schema_link* ← dummy data
- **offering**
 - *offering_id* ← provided or generated
 - *offering_name* ← descr
- **observation**
 - *observation_id* (automatically generated)
 - *time_stamp* ← bt+t converted to datetime
 - *text_value* ← sv
 - *numeric_value* ← v
 - *mime_type* ← blank

With the assumption/restriction that procedure, phenomenon, feature and offering are pairwise one-to-one, it is possible to map a SenML sensor description to the combination of SensorML SOS procedure, feature and offering. Furthermore, a measurement in SenML maps to a combination of phenomenon and observation in SOS. The conclusion is that it is possible to map representations in SenML into SensorML and thus to wrap and present this further into SOS services, if care is taken for the handling of missing information. This can enable support of interoperability between an approach using SenML and an approach using SensorML.

7 Visualisation and User Interaction Services

The visualization support of the CITI-SENSE platform aims at providing a set of reusable graphical components, or widgets, for appropriate visual depiction of heterogeneous-nature sensor data collected by the platform. By reusability in a broad sense we refer to the ability of widgets to be easily configured and deployed in various kinds of end-user applications and platforms, such as web portals and mobile devices, or smartphones. By appropriateness of visualization we refer to the ability of widgets to support multiple visual representations, such as maps, time series, charts, etc., rendering in appropriate and rich visual form raw and extracted features of collected and processed data, such as temporal, spatial, and other available dimensions.

7.1 Usage and Requirements

The main purpose of the CITI-SENSE visualization widgets, hereafter just visualization widgets or simply visualizations, consists in providing rich visual insights into environmental conditions, inferred from numerous sensors and relevant data sources feeding the CITI-SENSE platform. Rich and appropriate visualizations able to provide valuable insights into investigated environments raising environmental conditions awareness in different groups of stakeholders, be it a municipality or a single citizen.

Reusability of the visualization widgets is an important requirement in order to achieve the impact and reach target stakeholders. In order to address it, the CITI-SENSE visualization widgets will be made renderable through different end-user platforms, such as web portals and mobile applications. Besides that, to simplify the integration of the widgets by the developers, they will be made easily configurable and depending on the state of the art technologies for the target platforms.

In order to provide appropriate and rich visualizations, the visualization widgets will support main visual rendering forms relevant to the environmental data, such as maps, time series, different types of charts (such as pie, column, bar, area, etc.).

7.2 Logical Service Interfaces and Information Model

Visualization widgets are provided in order to interact with the data/product CITI-SENSE module in order to retrieve the data to be visually rendered. In order to reduce the computation overhead required for rendering visualization on the end-user side, be it web portal or mobile application, the data required for widgets has been appropriately prepared for a dedicated visualization on the side of the CITI-SENSE platform.

7.3 Implementation Technologies

Given the requirement of supporting different platforms when rendering visualizations, the most flexible solution is to use the same technology both for desktop-based web platforms and mobile platforms. Taking this into consideration, the best option consists in using HTML5/JavaScript technology, which is natively supported by most of modern browsers rendering web portals, and

supported as well through the webview component in modern mobile platforms, such as iPhone and Android.

Given the requirement of configurability and ease of deployment, the best option consists in implementing widgets as jquery plugins, due to the fact that most of modern web portals with dynamic and interactive content are already based on jquery JavaScript extension.

Regarding the specific visualization technologies to be used for implementing CITI-SENSE widgets we have surveyed and selected the following JavaScript-enabled libraries, considering the maturity of provided technology and richness of the functionality:

- Maps
 - Google Maps + Google maps JavaScript API
 - OpenStreetMap + OpenLayers JavaScript API
 - Kartograph
 - Heatmaps

- Timeseries and other charts
 - Google chart tools
 - Highcharts
 - Envision
 - Crossfilter
 - Raphael

A short description of these are provided in the following.

7.4 Widgets for Maps

7.4.1 Google Maps + Google Maps JavaScript API



Figure 7-1 Google Map widgets

The [Google Maps Javascript API](https://developers.google.com/maps/documentation/javascript/)¹² lets you embed [Google Maps](https://www.google.com/maps/) in your own web pages. API provides a number of utilities for customizing and manipulating maps in order to create interactive user experience.

¹² <https://developers.google.com/maps/documentation/javascript/>

7.4.2 OpenStreetMap + OpenLayers JavaScript API

[OpenLayers](#) makes it easy to put a dynamic map in any web page. It can display map tiles and markers loaded from any source. OpenLayers has been developed to further the use of geographic information of all kinds.

7.4.3 Kartograph

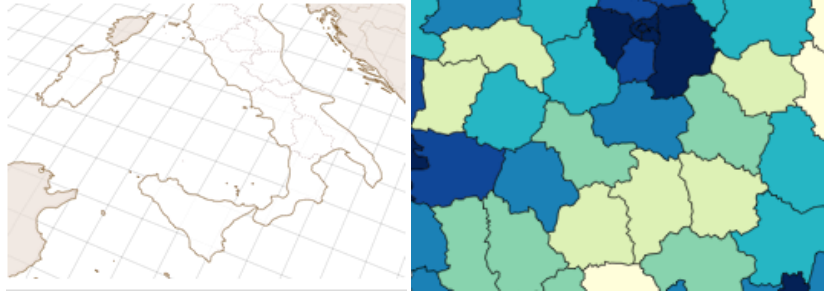


Figure 7-2 Kartograph widgets

[Kartograph](#) is a simple and lightweight framework for building interactive SVG map applications without Google Maps or any other mapping service.

7.5 Widgets for Charts

7.5.1 Google chart tools - interactive JavaScript library

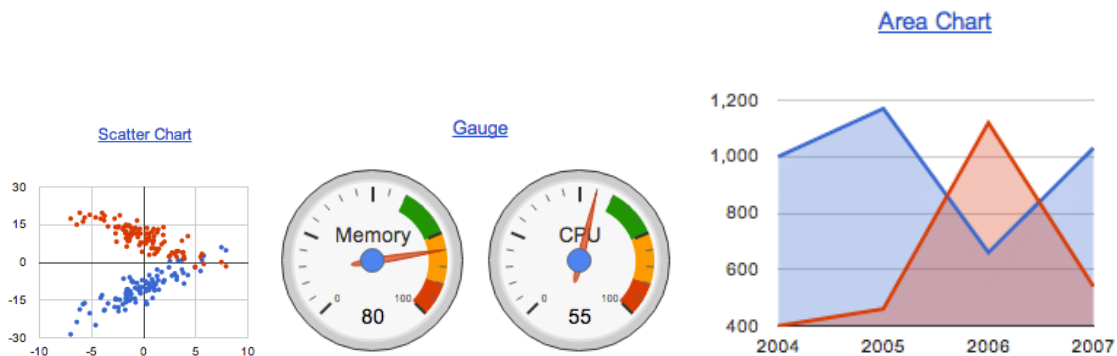


Figure 7-3 Google chart widgets

[Google chart tools](#) is a rich set of visualization components, including pie, scatter, gauge, bar, line, column charts and many others. The charts are based on pure HTML5/SVG technology (adopting VML for old IE versions)

7.5.2 highcharts.js - interactive JavaScript charts



Figure 7-4 Highchart widgets

[Highcharts](#) is a charting library written in pure HTML5/JavaScript, offering intuitive, interactive charts to your web site or web application. Highcharts currently supports line, spline, area, areaspline, column, bar, pie, scatter, angular gauges, arearange, areasplinerange, columnrange, bubble, box plot, error bars, funnel, waterfall and polar chart types.

7.5.3 envision.js - A HTML5 time series charts

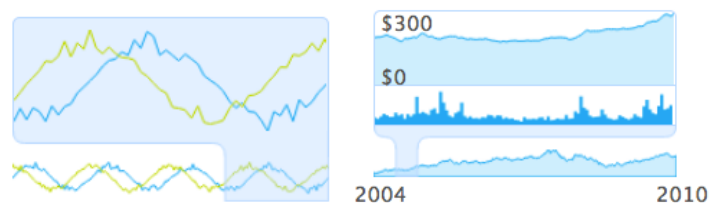


Figure 7-5 Envision time series widgets

[envision.js](#) is a javascript library for creating fast, dynamic and interactive HTML5 visualizations of real-time data.

7.5.4 Crossfilter - Fast Multidimensional Filtering library for Coordinated Views

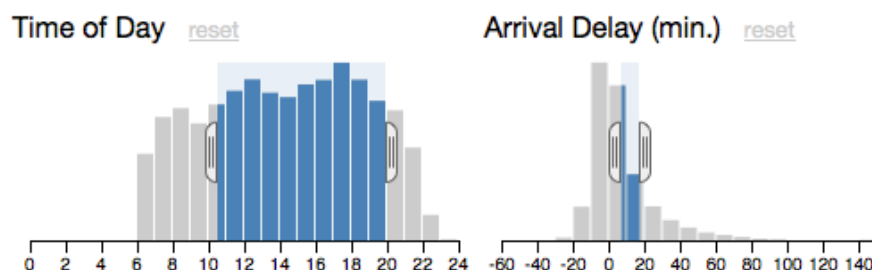


Figure 7-6 Crossfilter widgets

[Crossfilter](#) is a [JavaScript library](#) for exploring large multivariate datasets in the browser. Crossfilter supports extremely fast (<30ms) interaction with coordinated views, even with datasets containing a million or more records.

7.5.5 Raphael.js - JavaScript library

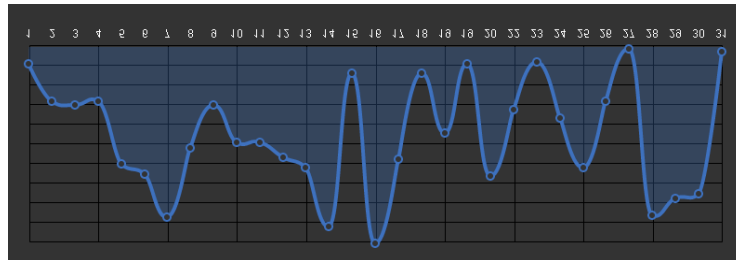


Figure 7-7 Raphael vector graphics widget

[Raphael](#) is a small JavaScript library that should simplify your work with vector graphics on the web. If you want to create your own specific chart or image crop and rotate widget, for example, you can achieve it simply and easily with this library.

Raphael uses the SVG W3C Recommendation and VML as a base for creating graphics. This means every graphical object you create is also a DOM object, so you can attach JavaScript event handlers or modify them later. Raphael's goal is to provide an adapter that will make drawing vector art compatible cross-browser and easy.

Raphael currently supports Firefox 3.0+, Safari 3.0+, Chrome 5.0+, Opera 9.5+ and Internet Explorer 6.0+.

7.6 Heatmaps

7.6.1 heatmap.js – A HTML5 canvas heatmap library

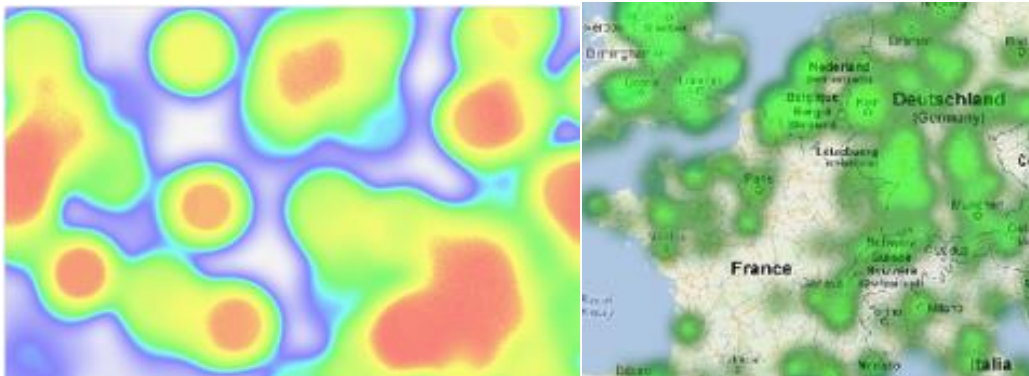


Figure 7-8 Heatmap widget

[heatmap.js](#) is a JavaScript library that can be used to generate web heatmaps with the html5 canvas element based on your data. Heatmap instances contain a store in order to colorize the heatmap based on relative data, which means if you're adding only a single datapoint to the store it will be displayed as the hottest (red) spot, then adding another point with a higher count, it will dynamically recalculate. The heatmaps are fully customizable – you can choose your own color gradient, change its opacity, datapoint radius and much more.

Heatmap can be added as a layer to maps: google maps v3, openlayers, ArcGIS, Leaflet

7.7 Mobile widgets

7.7.1 JavaScript (web-based) solutions

In terms of the mobile platforms (i.e. Android, iOS and Windows), the most elegant solution is to use same technology as chosen for the desktop platforms. Taking this into consideration, the best option would be to utilise JavaScript client/server technology for the visualisation tools such as Highcharts or something similar. In this case, the visualisation would be done within the native (objective C, Java etc) mobile applications utilising webview component and appropriate JavaScript and HTML code.

Example of such an usage is given in <http://stackoverflow.com/questions/15113692/showing-highcharts-in-a-jquery-mobile-application> using the JavaScript based activity with an example shown in <http://stackoverflow.com/questions/10472839/using-javascript-in-android-webview>. This approach would require some programming and application design.

An alternative way would be to use a framework such as PowerGadgets Mobile (<http://www.softwarefx.com/mobile/pgm/>). This framework includes creator that enables application development without programming required but subsequently reduces the flexibility in the overall application functionality, branding etc. This approach would also enable other application frameworks such as DHTMLX Touch (<http://dhtmlx.com/touch/>) to be used.

An additional list of widget toolkits that could be used for the mobile developments are listed here <http://www.webdesignerdepot.com/2012/11/10-javascript-frameworks-to-improve-your-mobile-development/>.

7.7.2 Native solutions

In order to build a purely native mobile application not relying on the webview (browser) component and thus simplifying the architecture (not requiring a charting server component) it would be necessary to use a native charting libraries with all the drawing functionality handled within the mobile device. There are different possibilities available for different platforms and therefore most likely not leading to the same visual appearance.

For example, http://www.keepeedge.com/products/iphone_charting/ offers the native library for the iOS and also for Android OS (http://www.keepeedge.com/products/android_charting/). Other solutions such as core-plot (<https://code.google.com/p/core-plot/>), Shinobi Controls (<http://www.shinobicontrols.com>) or Three D Graphics (<http://www.threedgraphics.com/tdg/products/tools/ioschart/>) offer iOS-only native visualisation.

Android only solutions include AndroCharts (<http://codecanyon.net/item/androcharts-android-chart-library/4620242>), Tee Charts (<http://www.steema.com/teechart/mobile>), Android Charts (<http://www.artfulbits.com/products/android/aicharts.aspx>) and QC Chard 2D (<http://www.quinn-curtis.com/QCChart2DAndroidProdPage.htm>). There are also other solutions similar to these.

7.8 Cross platform issues

There is a number of issues related to cross platform development that we need to relate to. HTML5 applications are created using common web technology, they are easy to develop and support and they can be deployed to any platform. Is it still sometimes necessary to develop an application native, or has this approach become obsolete? It has been discussed if HTML5

applications would take over the market, but even if many applications are developed using HTML5, often in combination with different technologies, many applications are also implemented native. This is mainly because some applications need native features in the mobile device that HTML5 applications can not use. Some issues are:

- Most hardware in the mobile device can only be accessed in a native application.
- It is harder to make an application look and feel native with a HTML5 development approach. Native applications can use customized components that a user expects to use when operating the application, like scroll bars and gesture recognizers. Implementing a native looking layout using HTML5, JavaScript and CSS is harder.
- Native applications have a better performance than applications developed with web technologies. For some applications this difference is not noticeable, but for some games and other high output applications, the HTML5 implementation would be significantly slower than the native. Native platforms have the fastest graphics and most fluid animations.
- Storing content offline can not be done in a secure manner with a HTML5 application. This is not a problem with native applications.
- Web applications are not accepted in native app stores.

Hybrid approaches, combining both HTML5 and use of platform specific features is a possibility for addressing some of these issues.

The conclusion for CITI-SENSE is that we should be able to support the functionality we need with HTML5 based applications, and multi-platform toolkits like PhoneGap. The CITI-SENSE architecture will, however, also support native-based apps for those who would have a preference for that in some situations.

7.9 Visualisation technologies from OGC, and related GEOSS projects etc.

The OGC consortium and the GEOSS community is using a number of standardised visualisation related technologies like WMS etc. The CITI-SENSE project will explore the further usage of these as much as possible.



8 Further Services for Consideration

8.1 CITI-SENSE Data Model

The CITI-SENSE Data Model will be further detailed and implemented in an extended version for the D7.3 version of the CITI-SENSE Platform and Architecture report.

8.2 Management and Security services

Management and Security services become important for the deployment of the CITI-SENSE platform. D7.3 version of the CITI-SENSE Platform and Architecture report will specify further a strategy for how this will be supported.

8.3 Ubiquitous Public Access Services

ISO/TC211 (Geographic Information and Geomatics) is an international standards organization and is trying to standardize “Geographic information – Ubiquitous public access – reference model (UPA)” for linking and interoperating spatial information. D7.3 version of the CITI-SENSE Platform and Architecture report will specify further a strategy for further exploration of this.

8.4 GEOSS Integration

It is an objective of CITI-SENSE to provide results and resources in the the GEOSS infrastructure. The Group on Earth Observations (GEO)¹³ is coordinating efforts to build a Global Earth Observation System of Systems, or GEOSS. The further strategy for this will be more explored in the D7.3 version of this report.

8.5 Social Sensors

A Social Sensor is a software agent that provides observations about an environment via communicating to other agents. The role is to understand citizen’s experience about their environment. This approach to sensing will be described more in D7.3 CITI-SENSE Platform and Architecture interim version.

8.6 Linked (Open) Data Services

Linked Open Data [2] is emerging as a source of higher visibility for environmental data that will enable the generation of new businesses as well as a significant advance for research in the environmental area. The opportunity to make CITI-SENSE data available as Linked Open Data will be described more in the D7.3 version of this report.

¹³ <https://www.earthobservations.org>

8.7 Communication and Composition services

Further need for support for communication and composition services will be analysed for the next phases of the CITI-SENSE project, as outlined for task 7.4 until M36 of the project. This task will investigate the reuse, integration, and enhancements of components for discovery, composition, mediation, and execution of environmental data and services.

8.7.1 Event Services

The platform will consider to support event services which allow clients to receive notifications about relevant events and data values within the CITI-SENSE platform. This will be explored further for later versions of this report.

8.8 Participation and Empowerment services

Further services for integration with social media and social innovation platforms will also be explored for further versions of the CITI-SENSE platform.

9 Conclusion and Further Platform Evolution

Further versions of the CITI-SENSE Platform and Architecture will be updated based on experiences with the existing services, as well as based on the further evolution on some of the services selected for further consideration.

This D7.1 CITI-SENSE Platform and architecture 1.0 provides the foundation for the services provided by the CITI-SENSE platform.

The continued D7.3 CITI-SENSE Platform and architecture document will extend from this baseline Platform and architecture document and provides the further foundation for the services provided by the CITI-SENSE platform.

The plan for D7.4 due for M24 is a further focus on the consolidation of the data flow support for a number of sensor providers, and also support for the initial level of security mechanisms. Further requirements will be related to identified needs by the user-driven/participatory design approach suggested by WP4 in D4.2, and with the ontologies proposed in D7.2.

The plan for D7.5 due for M36 is on enhancements to the platform technologies based on the experiences from the various empowerment initiatives, and a further focus on the support for Linked Open Data utilising linked data technology results from other projects currently in progress¹⁴.

The plan for D7.6 due for M48 is for a final packaging of the platform for further use and deployment after the end of the project.

¹⁴ www.dapaas.eu and www.smartopendata.eu

10 Annex A: Access to CITI-SENSE WFS-T Server

10.1 Setup

This section describes how to access the CITI-SENSE WFS-T server provided by Snowflake Software using their GO Publisher and GO Loader software products.

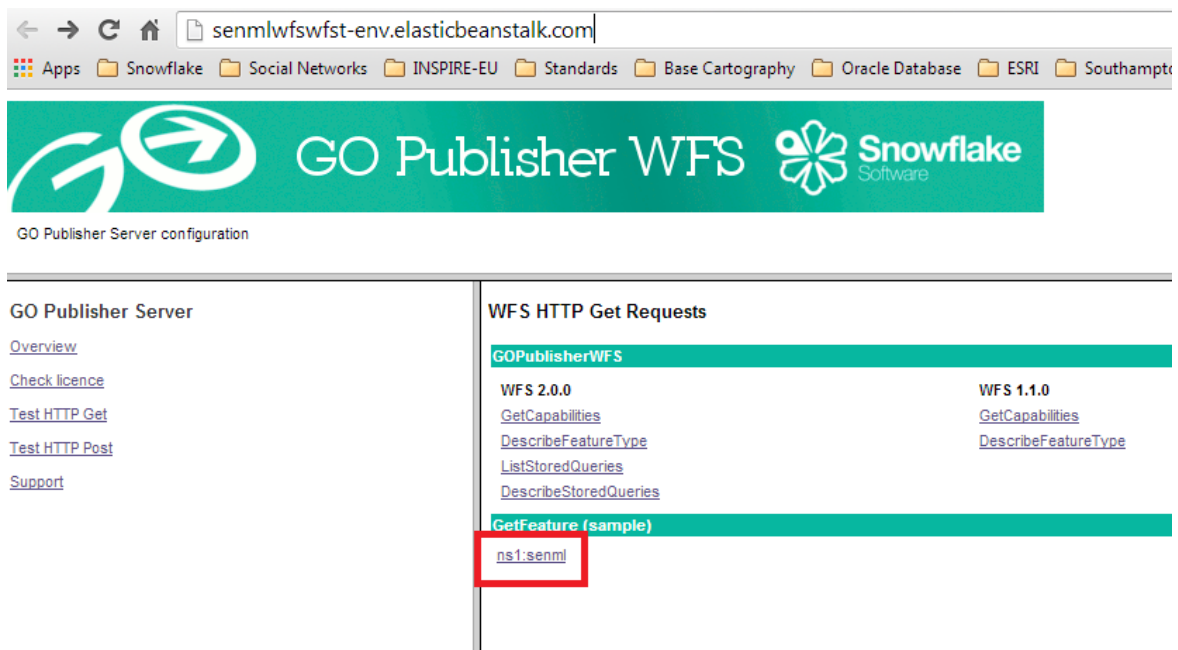
In CITI-SENSE a WFS-T web service is deployed to insert SenML data from SensApp Platform into the PostGIS database that is hosted on the Amazon Cloud. Configuring and deploying the WFS-T for SenML data involves the following steps:

- Set up a database in the Amazon Cloud and create a database schema that stores the data;
- Use GO Publisher to create a Web Archive file (.war) file which encapsulates all required components to deploy the web services, such as application schemas and the transformation configuration;
- Use GO Loader to create a project file (.glp), that contains the transformation configuration between the SenML application schema (source) and the database schema (target);
- Add the GO Loader project file in the WAR file that was created in the previous step;
- Set up and configure an Amazon application on which the WFS-T web service will be deployed;
- To test the web service, use a REST console to insert SenML data onto the WFS-T.

10.2 Samples

WFS landing page

When a WFS web service is successfully deployed on the application server, GO Publisher provides a useful landing page that allow users to run quick tests and confirm correct working of the web service.



senmlwfsfst-env.elasticbeanstalk.com

Apps Snowflake Social Networks INSPIRE-EU Standards Base Cartography Oracle Database ESRI Southampt

GO Publisher WFS Snowflake Software

GO Publisher Server configuration

GO Publisher Server

- [Overview](#)
- [Check licence](#)
- [Test HTTP Get](#)
- [Test HTTP Post](#)
- [Support](#)

WFS HTTP Get Requests

GOPublisherWFS

WFS 2.0.0	WFS 1.1.0
GetCapabilities	GetCapabilities
DescribeFeatureType	DescribeFeatureType
ListStoredQueries	
DescribeStoredQueries	

GetFeature (sample)

[ns1.senml](#)

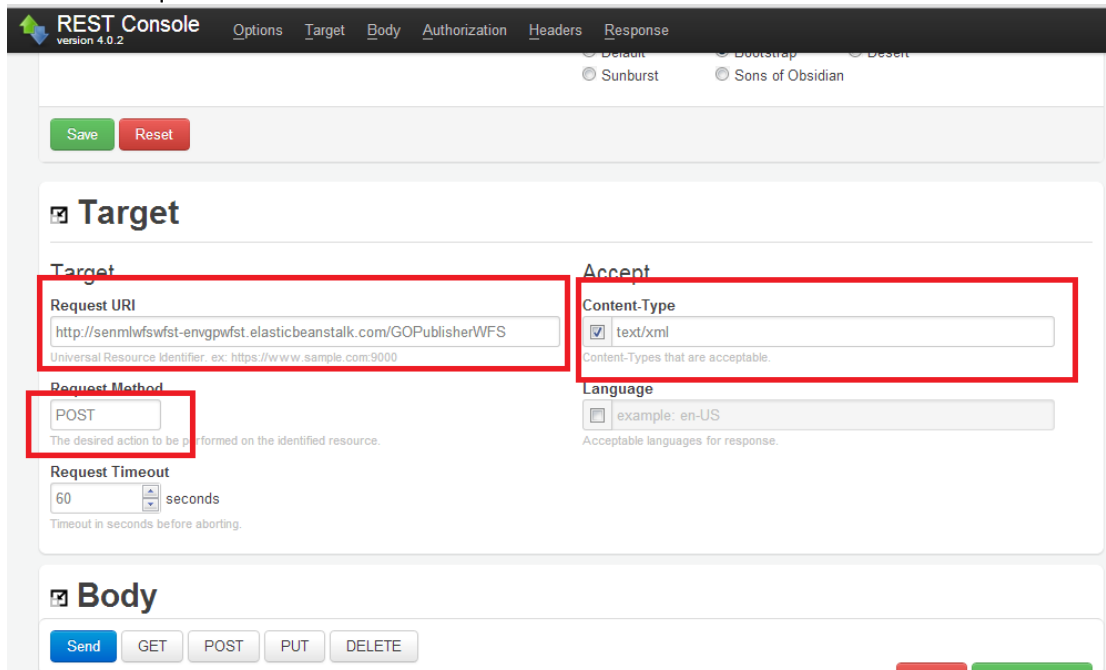
Example of a WFS Insert transaction in XML

The XML sample below shows a typical WFS insert operation encoded in XML. In this sample a SenML encoded observation is inserted. This observation has four temperature measurements at four time intervals.

```
<?xml version="1.0"?>
  <wfs:Transaction version="2.0.0" service="WFS" xmlns:ns1="urn:ietf:params:xml:ns:senml" xmlns:gml="http://www.opengis.net/gml" xmlns:wfs="http://www.opengis.net/wfs/2.0" xmlns:s xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="urn:ietf:params:xml:ns:senml ./senml.xsd http://www.opengis.net/wfs/2.0 http://schemas.opengis.net/wfs/2.0/wfs.xsd http://www.opengis.net/gml http://schemas.opengis.net/gml/3.1.1/base/gml.xsd">
    <wfs:Insert>
      <senml bt="1363175059" bn="NILU/UVSensor69_19/alt9_alb9_oct9_AO" xmlns="urn:ietf:params:xml:ns:senml">
        <e v="1.6" t="99999" u="W/m2"></e>
        <e v="1.6" t="345652" u="W/m2"></e>
        <e v="1.6" t="1030259" u="W/m2"></e>
        <e v="1.6" t="1022359" u="W/m2"></e>
      </senml>
    </wfs:Insert>
  </wfs:Transaction>
```

REST Console

To test the correct working of the WFS-T the Chrome web application REST Console is used to post the insert request as detailed above.



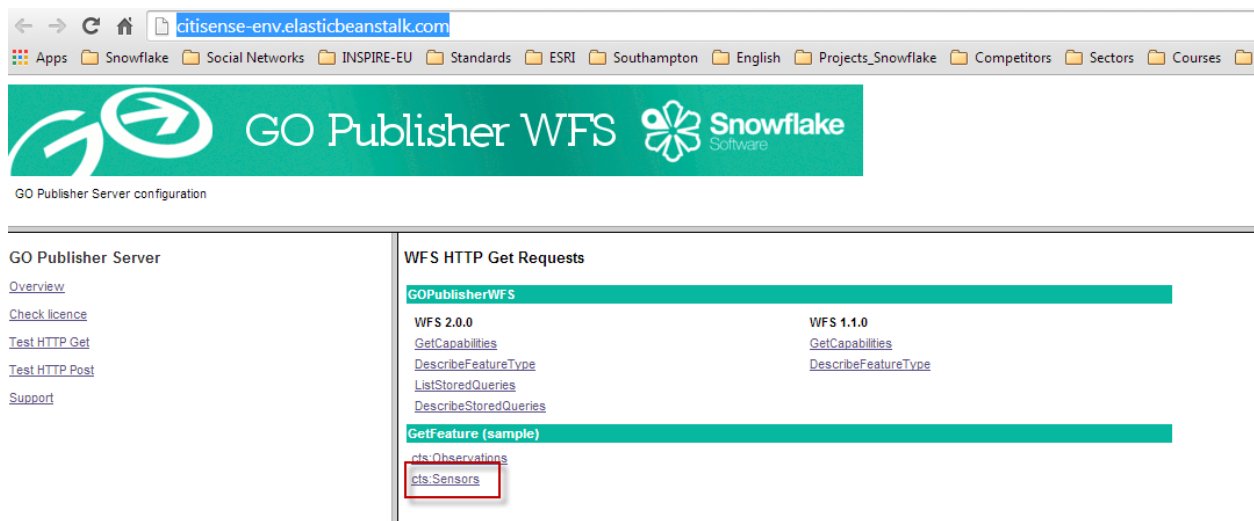
10.3 Sensors Data Access via WFS

The data stored in the centralised database can be access through open standard web services such as WFS. Snowflake set up a number of sample WFS endpoints that you can connect to to query the data. The WFS returns the result-set of the query in a GML encoding (i.e. a GML file).

The WFS endpoint to consult the sensors include into the system is: <http://citisense-env.elasticbeanstalk.com/>

This will open a landing page from which you can test various standard WFS calls, such as GetCapabilities, DescribeFeatureType and GetFeature.

- **GetCapabilities** returns an XML document that describes some basic information about the WFS web service and the type of capabilities it has.
- **DescribeFeatureType** returns and XML document, that list which feature types this web services supports.
- **GetFeature** returns a GML document, that contains actual data based on your query criteria (eg. all sensor data in a certain time interval, all data from a specific geographic region, etc.).



Example

By clicking the [cts:Sensors](#) link on the landing page the WFS returns the first 10 features/results for this feature type from the database. The resulting GML document looks something like this (showing the first 3 sensors):

```
<!--Created by GO Publisher WFS 3.0.0 Build 2f41ba3 from 2013-11-04 18:50-->
<!--Snowflake Software Ltd. (http://www.snowflakesoftware.com)-->
<wfs:FeatureCollection xsi:schemaLocation="http://www.citi-sense.eu/citisense
cts.xsd http://www.opengis.net/wfs/2.0
http://schemas.opengis.net/wfs/2.0/wfs.xsd" numberMatched="12"
numberReturned="10" timeStamp="2014-05-
13T15:43:22" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
```

```
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:cts="http://www.citi-  
sense.eu/citisense" xmlns:gml="http://www.opengis.net/gml/3.2"  
xmlns:xlink="http://www.w3.org/1999/xlink"  
xmlns:gmd="http://www.isotc211.org/2005/gmd"  
xmlns:gco="http://www.isotc211.org/2005/gco"  
xmlns:gss="http://www.isotc211.org/2005/gss"  
xmlns:gts="http://www.isotc211.org/2005/gts"  
xmlns:gsr="http://www.isotc211.org/2005/gsr"  
xmlns:wfs="http://www.opengis.net/wfs/2.0"  
xmlns:ows="http://www.opengis.net/ows/1.1"  
xmlns:fes="http://www.opengis.net/fes/2.0"<wfs:member><cts:Sensors  
gml:id="SEN_1"><cts:sensorProviderId xlink:href="#1"/>
```

```
<cts:idsensor>1  
</cts:idsensor>  
<cts:name>SQWE  
</cts:name>  
<cts:description>Interior  
</cts:description>  
<cts:dateregistration>2014-03-01T10:23:54.000  
</cts:dateregistration>  
<cts:descriptiveword1>NO  
</cts:descriptiveword1>  
<cts:descriptiveword2>O2  
</cts:descriptiveword2>  
<cts:language>English  
</cts:language>  
</cts:Sensors>  
</wfs:member>  
<wfs:member><cts:Sensors gml:id="SEN_2"><cts:sensorProviderId  
xlink:href="#2"/>
```

```
<cts:idsensor>2  
</cts:idsensor>  
<cts:name>SGWJK  
</cts:name>  
<cts:description>Exterior  
</cts:description>  
<cts:dateregistration>2014-03-01T10:23:54.000  
</cts:dateregistration>  
<cts:descriptiveword1>NO  
</cts:descriptiveword1>  
<cts:descriptiveword2>O2  
</cts:descriptiveword2>  
<cts:language>English  
</cts:language>  
</cts:Sensors>  
</wfs:member>  
<wfs:member><cts:Sensors gml:id="SEN_3"><cts:sensorProviderId  
xlink:href="#3"/>
```

```
<cts:idsensor>3  
</cts:idsensor>  
<cts:name>23ERD  
</cts:name>  
<cts:description>Exterior  
</cts:description>
```

```

<cts:dateregistration>2014-03-02T10:23:54.000
</cts:dateregistration>
<cts:descriptiveword1>NO
</cts:descriptiveword1>
<cts:descriptiveword2>O2
</cts:descriptiveword2>
<cts:language>English
</cts:language>
</cts:Sensors>
</wfs:member>
<wfs:member><cts:Sensors gml:id="SEN_4"><cts:sensorProviderId
xlink:href="#3"/>

```

10.3.1 Inserting data using the WFS-T service

To ingest a sensor into the database, the data needs to be posted using a HTTP Post operation to the WFS-T web service. An example of the contents of this HTTP Post operation can be found below.

```

<?xml version="1.0"?>
<wfs:Transaction version="2.0.0" service="WFS" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:cts="http://www.citi-sense.eu/citizensense"
xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gco="http://www.isotc211.org/2005/gco"
xmlns:gss="http://www.isotc211.org/2005/gss" xmlns:gts="http://www.isotc211.org/2005/gts"
xmlns:gsr="http://www.isotc211.org/2005/gsr" xmlns:wfs="http://www.opengis.net/wfs/2.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:fes="http://www.opengis.net/fes/2.0">
  <wfs:Insert>
    <cts:Sensors gml:id="SEN_11">
      <cts:sensorProviderId xlink:href="#1"/>
      <cts:idsensor>12</cts:idsensor>
      <cts:name>RRRRRR</cts:name>
      <cts:description>Interior</cts:description>
      <cts:dateregistration>2014-05-06T16:00:00.000</cts:dateregistration>
      <cts:descriptiveword1>N02</cts:descriptiveword1>
      <cts:descriptiveword2>O3</cts:descriptiveword2>
      <cts:language>Catalan</cts:language>
    </cts:Sensors>
  </wfs:Insert>
</wfs:Transaction>

```

- `<cts:Sensors gml:id="SEN_11">` is inserted into the database following a sequence, so you do not have to worry about these values.

- For inserting the `<cts:sensorProviderId xlink:href="#1"/>` it is needed to consult the static table for sensorproviders and the primary_key automatically assigned to the sensor:

	primary_key [PK] numeric	idprovider character var	name character var
1	1	1	QU
2	2	2	Alphasense
3	3	3	Obeo
4	4	4	CRIC
5	5	5	Geotech
6	6	6	Airbase
7	7	7	CVIT
8	8	8	U-Hopper
9	9	9	DNET
10	10	10	JSI
*			

Running the Request in REST Console

For submitting a WFS-Insert we use a tool like the REST Console. The [REST Console](#) can be used as an extension to the Chrome web browser and allows you to run various operations.

The steps below detail how to run the INSERT request in the REST Console.

A. Go to the Target element.

The parameters are:

- Request URL: <http://citisense-wfst.elasticbeanstalk.com/GOPublisherWFS>
- Request Method: POST
- Content Type: text/xml

Target

Target

Request URI

Universal Resource Identifier, ex: https://www.sample.com:9000

Request Method

The desired action to be performed on the identified resource.

Request Timeout
 seconds
Timeout in seconds before aborting.

Accept

Content-Type
 text/xml
Content-Types that are acceptable.

Language
 example: en-US
Acceptable languages for response.

B. In the Body element copy your WFS Insert data

Request Payload > Raw Body

```
<?xml version="1.0"?>
<wfs:Transaction version="2.0.0" service="WFS" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:cts="http://www.citi-sense.eu/citisense"
xmlns:gml="http://www.opengis.net/gml/3.2" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gmd="http://www.isotc211.org/2005/gmd" xmlns:gco="http://www.isotc211.org/2005/gco"
xmlns:gss="http://www.isotc211.org/2005/gss" xmlns:gts="http://www.isotc211.org/2005/gts"
xmlns:gsr="http://www.isotc211.org/2005/gsr" xmlns:wfs="http://www.opengis.net/wfs/2.0"
xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:fes="http://www.opengis.net/fes/2.0">
  <wfs:Insert>
    <cts:Sensors gml:id="SEN_11">
      <cts:sensorProviderId xlink:href="#1"/>
      <cts:idsensor>12</cts:idsensor>
      <cts:name>RRRRRR</cts:name>
      <cts:description>Interior</cts:description>
      <cts:dateregistration>2014-05-06T16:00:00.000</cts:dateregistration>
      <cts:descriptiveword1>N02</cts:descriptiveword1>
      <cts:descriptiveword2>03</cts:descriptiveword2>
      <cts:language>Catalan</cts:language>
    </cts:Sensors>
  </wfs:Insert>
</wfs:Transaction>
```

Body

Content Headers

Content-Type
 example: application/x-www-form-urlencoded
The mime type of the body of the request (used with POST and PUT requests)

Encoding
 example: utf-8
Acceptable encodings. [See HTTP compression.](#)

Content-MD5
 example: Q2hiY2sgSW50ZWdyaXR5IQ==
A Base64-encoded binary MD5 sum of the content of the request body.

Request Payload

RAW Body

```

<cts.name>RRRRRR</cts.name>
<cts.description>Interior</cts.description>
<cts.dateregistration>2014-05-
06T16:00:00.000</cts.dateregistration>

<cts.descriptiveword1>NO2</cts.descriptiveword1>

<cts.descriptiveword2>O3</cts.descriptiveword2>
<cts.language>Catalan</cts.language>

</cts.Sensors>

</wfs:Insert>
</wfs:Transaction>

```

This will create a RAW body and treat the params below as query string params only.

Leave the other options unticked (is default).

C. Click the SEND button to commit the Insert operation.



D. Check the result

When the Insert operation has been successful, the following message appears in the Response window.

Response

Response Body
RAW Body
Response Headers
Response Preview
Request Body
Request Headers

Color Theme Bootstrap **Force Syntax Highlighting** Auto JSON XML HTML CSS

```

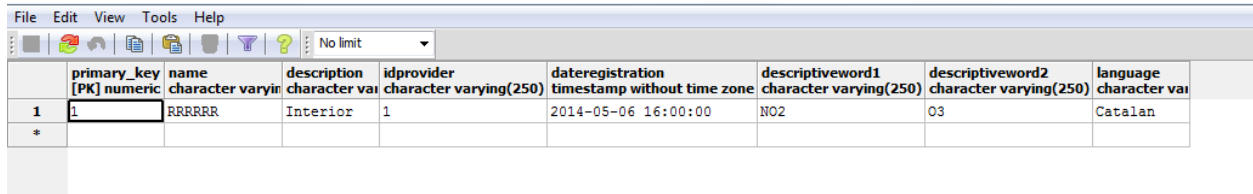
1. <?xml version="1.0" encoding="UTF-8" standalone="yes"?>
2. <wfs:TransactionResponse version="2.0.0" xsi:schemaLocation="http://www.opengis.net/wfs/2.0 http://schemas.opengis.net/wfs/2
3.   <wfs:TransactionSummary>
4.     <wfs:totalInserted>1</wfs:totalInserted>
5.   </wfs:TransactionSummary>
6.   <wfs:InsertResults>
7.     <wfs:Feature>
8.       <fes:ResourceId rid="ns1:senml.null"/>
9.     </wfs:Feature>
10.  </wfs:InsertResults>
11. </wfs:TransactionResponse>

```

Note the element `<wfs:totalInserted>1</wfs:totalInserted>` , this indicates that one new feature has been inserted successfully.

10.3.2 How does this data look like in the database?

The screenshot below show the result of the HTTP Post operation to the WFS-T web service. The Snowflake Software technology that sits in the WFS-T web service transforms the incoming data into standard SQL statements and loads the dataset into the database.



	primary_key [PK] numeric	name character varying	description character varying	idprovider character varying(250)	dateregistration timestamp without time zone	descriptiveword1 character varying(250)	descriptiveword2 character varying(250)	language character varying
1	1	RRRRRR	Interior	1	2014-05-06 16:00:00	N02	03	Catalan
*								

11 Annex B: Access to SensApp Server

11.1 Setup

The following describes how to access the CITI-SENSE SensApp server provided by SINTEF

11.2 Installing SensApp dependencies

SensApp is designed to run in Jetty, a lightweight application server that can easily be deployed on constrained servers running Unix-based systems (32Mb of RAM are sufficient). Here is installation instructions for setup of a SensApp cloud service. One SensApp cloud service is initially being provided by SINTEF, but more services can be set up if necessary.

1. MongoDB:
 - a. <http://www.mongodb.org/downloads>
 - b. Create C:\data and C:\data\db directories
 - c. Start the MongoDB server (run `mongod.exe`)
2. Git:
 - a. <http://code.google.com/p/msysgit/downloads/list>
3. Maven:
 - a. <http://maven.apache.org/download.html>
4. Optional: Google Chrome
 - a. <https://www.google.com/intl/en/chrome/browser/?hl=en>
5. Optional: Google Chrome REST Console
 - a. https://chrome.google.com/webstore/detail/cokgbflfommojglbmbpenpphppikmonn?hl=fr&utm_source=chrome-ntp-launcher

11.3 Deploying SensApp in the development environment

1. Clone the SensApp code repository:
 - a. Run the git terminal
 - b. Execute `git clone git://github.com/SINTEF-9012/sensapp.git`
2. Go to the root project:
 - a. `cd SensApp`
 - b. `cd net.modelbased.sensapp`
 - c. `mvn clean install15`
3. Run the current development system:
 - a. `cd ..`
 - b. `cd net.modelbased.sensapp.system.sprints.first`
 - c. `mvn jetty:run`
4. To kill the SensApp server, just hit Ctrl-C in the terminal.

¹⁵ The first time, maven will download a lot of dependencies ... the other builds will be more quick.

12 Annex C: Visualisation components

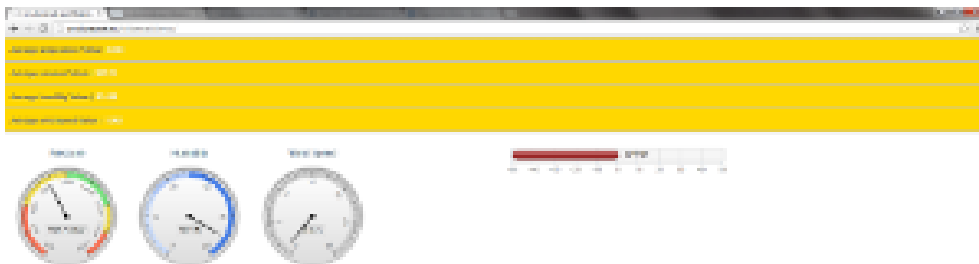
12.1 Widget framework for Measurements (DNET)

The following Widget framework for measurements has been designed and provided by the CITI-SENSE partner Dunavnet.

The documentation and demo source code has been provided so that , mobile and web applications can easily be developed.

The framework has the following features:

- Based on Highcharts widgets (<http://www.highcharts.com/demo/>)
- Integrated with PHP in order to allow simple customization
- Demos with source code provided for iOS, Android and web platforms
- Supports mysql, mssql, postgresql and mongoDB
- Currently hosted on DNET server, but can be integrated on CITI-SENSE platform
- Please have a look at the [documentation](#) page for more details on how to use the framework
- Demo page is at this [link](#) (shown in below image)



12.1.1 Installation

Download and unzip citisense to a project folder. Open blank index.php and simple 'include' or 'require' Citisense.php from /libs/

```
1 <?php
2 include_once("libs/Citisense.php");
```

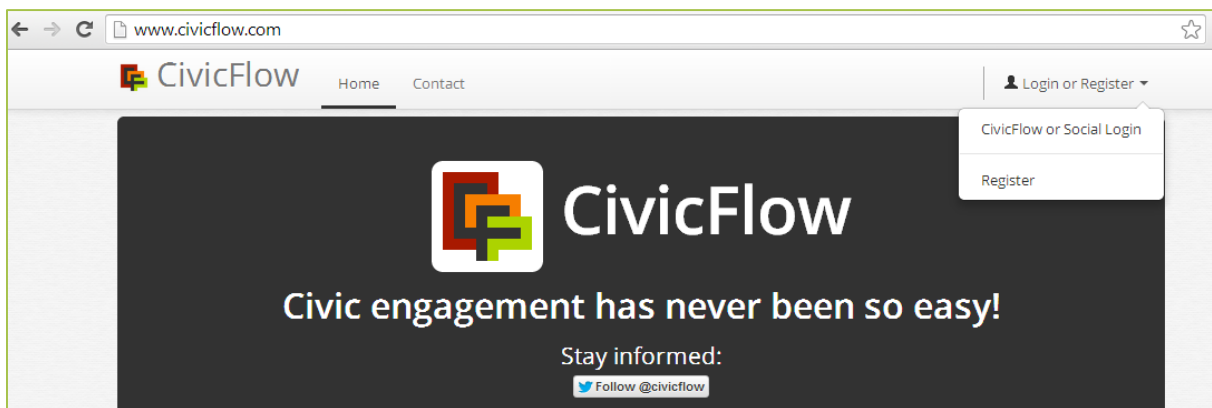
12.2 Widget framework for Questionnaires (U-Hopper)

A widget framework for multi-channel questionnaires has been developed by U-Hopper.

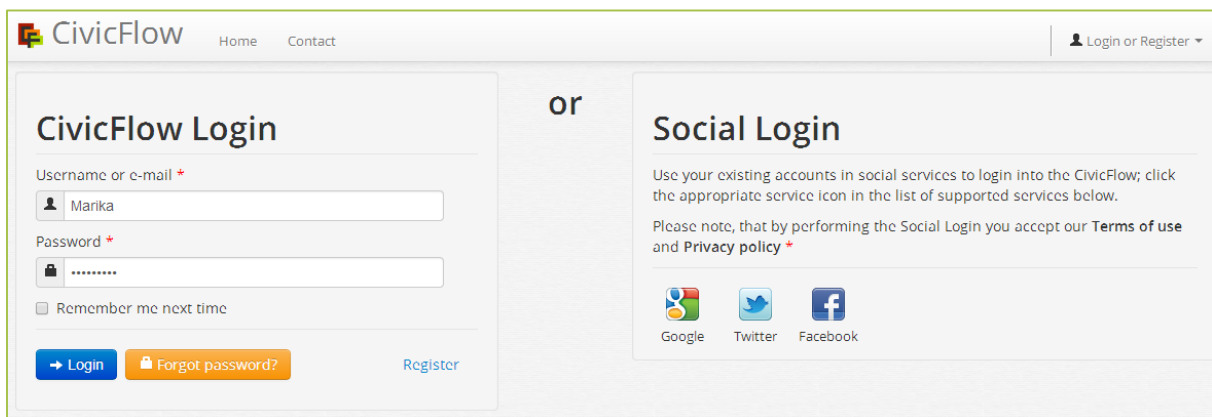
Go to www.civicflow.com and choose "CivicFlow or Social Login" or choose "Register" if you want or need to create a new account.

Registering as a user is uncomplicated and takes a minute or two.

If you choose "Social Login", you can log in with your Google, Twitter or Facebook account. This means CivicFlow will access your social profile information.

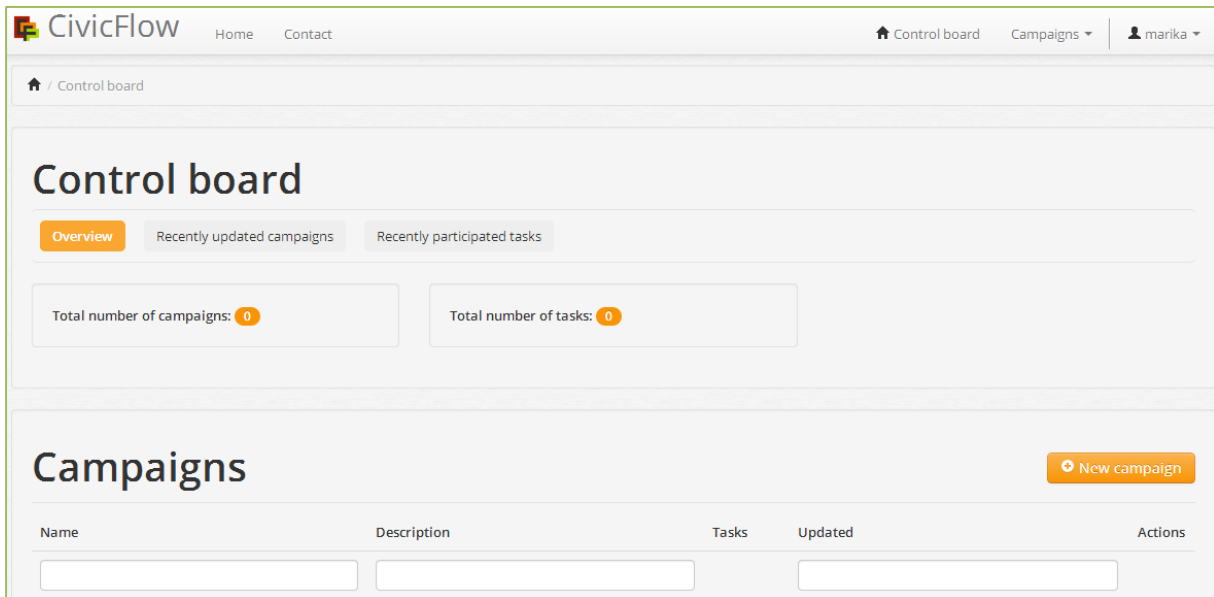


In the screenshot below, I choose to log in as a registered user.



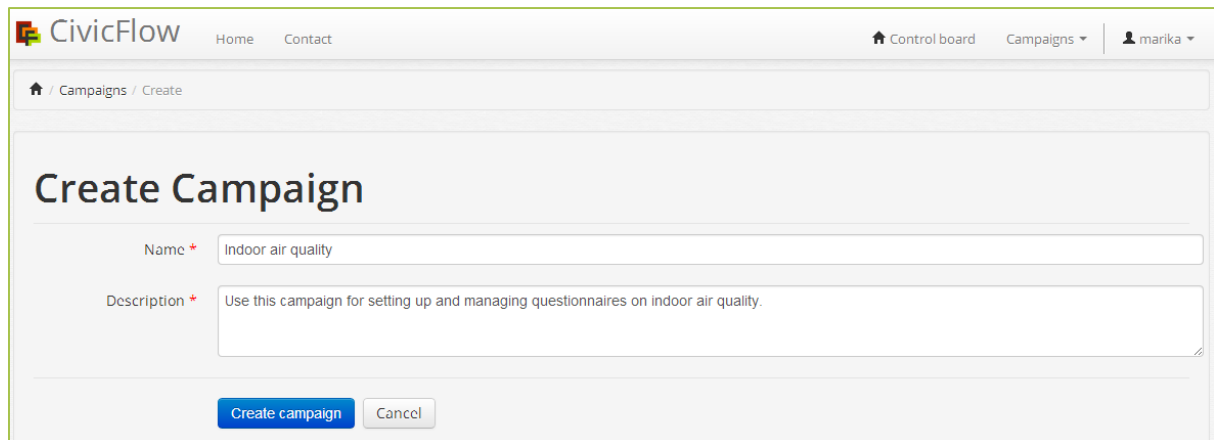
You need to **create a campaign** before you can create a questionnaire. You can add several questionnaires to one campaign.

To create a new campaign, click "New campaign"-button.



The screenshot shows the CivicFlow Control board interface. At the top, there is a navigation bar with 'Home' and 'Contact' links, and a user profile for 'marika'. The main heading is 'Control board', with tabs for 'Overview', 'Recently updated campaigns', and 'Recently participated tasks'. Below the tabs, there are two summary cards: 'Total number of campaigns: 0' and 'Total number of tasks: 0'. The 'Campaigns' section features a 'New campaign' button and a table with columns for Name, Description, Tasks, Updated, and Actions. The table currently contains no data rows.

In this example, I create a campaign entitled "Indoor air quality" and add the description "Use this campaign for setting up and managing questionnaires on indoor air quality". Click "Create campaign" when you have filled in name of campaign and description.



The screenshot shows the 'Create Campaign' form in the CivicFlow interface. The form has two input fields: 'Name' with the value 'Indoor air quality' and 'Description' with the value 'Use this campaign for setting up and managing questionnaires on indoor air quality'. At the bottom of the form, there are two buttons: 'Create campaign' (highlighted in blue) and 'Cancel'.

13 Annex I: Use case template and use cases

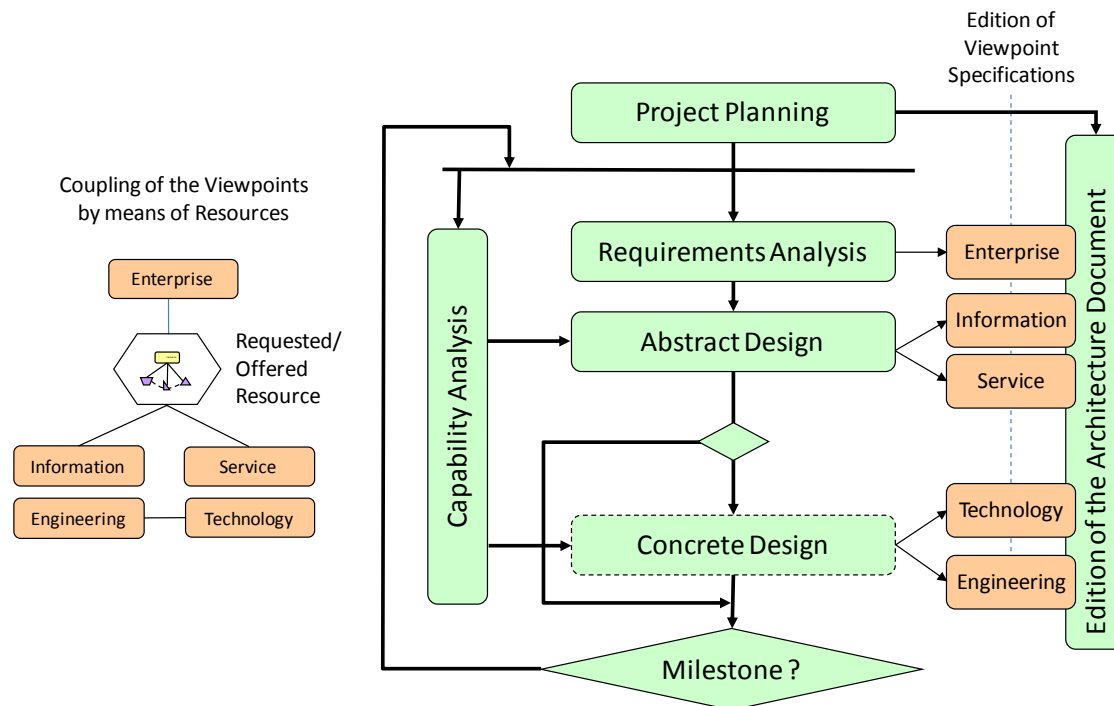
13.1 Use case methodology

This annex provides an overview about an SDI oriented Use Case Analysis Methodology. This methodology is derived from the SERVUS methodology that aims at a Design Methodology for Information Systems based upon Geospatial Service-oriented Architectures and the Modelling of Use Cases and Capabilities as Resources¹⁶. This methodology approach which has been successfully practiced in earlier environmental and geospatial project has been adapted for CITI-SENSE.

The SERVUS methodology relies upon a resource model as a common modelling language which is derived from the Representational State Transfer (REST) architectural style for distributed hypermedia systems. Hereby, a resource is considered to be an information object that is uniquely identified, may be represented in one or more representational forms (e.g. as a diagram, XML document or a map layer) and support resource methods that are taken from a limited set of operations whose semantics are well-known (uniform interface). A resource has own characteristics (attributes) and is linked to other resources forming a resource network. Furthermore, resource descriptions may refer to concepts of the domain model (design ontology) using the principle of semantic annotation, yielding so-called semantic resources.

Figure 14 shows the focus of the RM-ODP viewpoints typically during the steps of service analysis, design and implementation. The main viewpoint during initial analysis is the enterprise viewpoint. This also serves as the foundation for describing the resources (in terms of data, services and/or sensor information) which is required. An initial step will then be to compare the requested resources with potentially offered resources through a discovery and search process, in order to identify if the request for resources can be met by resources that already are available.

¹⁶ www.envirofi.eu



Relationship of RM-ODP viewpoints and analysis and design

Use case modelling has been shown to be an efficient and powerful approach to reach a common understanding of the system itself and its behaviour. In interdisciplinary projects, involving thematic experts from different domains (e.g. geospatial, environmental) as well as IT experts, it is as challenging to reach consensus on a common terminology. Otherwise, the consequences would include different interpretations and assumptions about the systems to be developed. Thus to avoid misunderstandings, use case descriptions should be based on a common vocabulary, stemming from a glossary and a thesaurus whenever possible.

The description of use cases is necessary to capture all functional and non-functional requirements of the system. The use cases also describe the interaction between the users and the system. Use cases are the most common practices for capturing and deriving requirements. The requirements of the system are described in a narrative way with minimal technical jargon.

In the geospatial context use cases are typically described in a semi-formal way, based on a structured textual description in tabular form derived from a template. Recent European research projects (such as SANY¹⁷, ENVIROFI¹⁸, ENVISION¹⁹, EO2HEAVEN²⁰ and TRIDEC²¹) based the description of their use cases on a similar template.

Based upon this approach, additional information about the requested information resources (e.g. type and format of needed data) is necessary to completely describe a use case from both a user's

¹⁷ EU FP7 project no. 033564 Sensors Anywhere (SANY) - <http://www.sany-ip.eu>

¹⁸ EU FP7 FI PPP project, ENVIROFI, <http://www.envirofi.eu>

¹⁹ EU FP7 project no. 1234, ENVIRONMENTAL Services Infrastructure with ONtologies, www.envision-project.eu

²⁰ EU FP7 project no. 244100 Earth Observation and ENVIRONMENTAL modeling for the mitigation of HEALTH risks (EO2HEAVEN) - <http://www.eo2heaven.org/>

²¹ EU FP7 project no. 258723 Collaborative, Complex and Critical Decision-Support in Evolving Crisis (TRIDEC) - <http://www.tridec-online.eu>

and system's point of view. The requirements should be derivable from the use cases. Three types of requirements can be identified:

- Functional requirements,
- Informational requirements,
- Non-functional requirements.

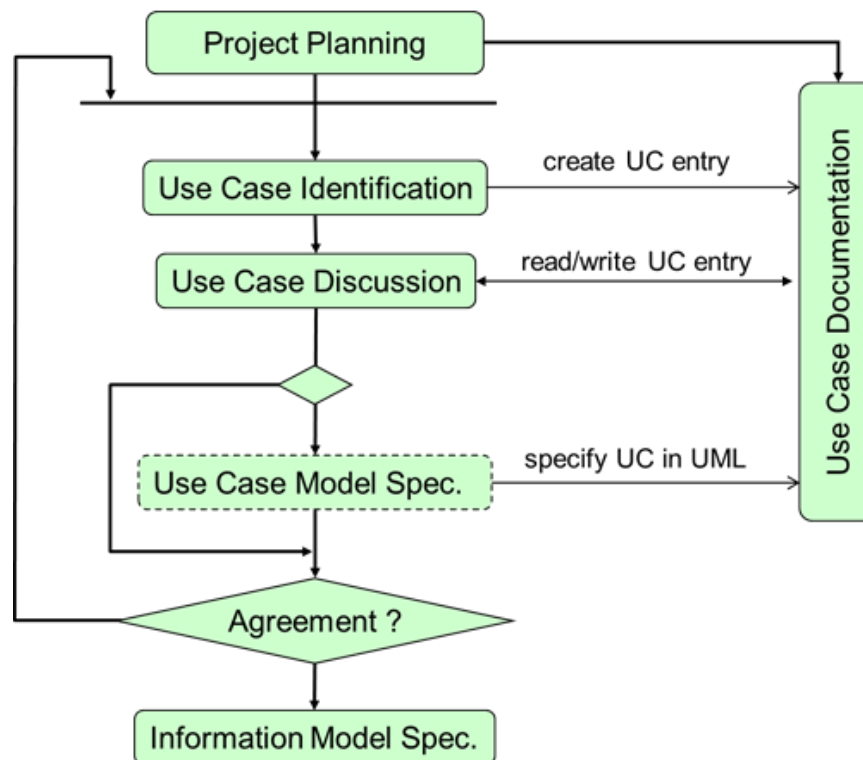
Functional requirements can be derived from the sequence of actions (main success scenario, extensions and alternative paths). The informational requirements address data that is exchanged between two communication partners, i.e. between users and the system or between system components. The non-functional requirements cover all requirements that do not alter the foreseen functionality of the system, e.g. the quality of data and results.

This approach provides a basis for use case development. However, the SERVUS methodology proposes that beside the functional and non-functional requirements the informational requirements are very important to complete the use case description. For a more detailed analysis and as a first step towards information modelling it is necessary to consider input data, data format, data type, data encoding, and the desired format of the output data, too. Thus the template contains additional issues like 'Requested Information Resources'.

The common form of a use case description is to describe it from the user's point of view where only the external perceivable behaviour is reflected. The described system is a black box for the user. This template should be used by both sides, The users and the system developers and operators. Both sides and all involved experts have to understand the use cases in the same way. Especially the IT experts should understand the user's requirements because they have to develop the IT components on the basis of the descriptions.

It is expected that each use case will be described in a semi-formal way. A form was created to structure the textual description. The table represents the use case template and is shown in Annex C. The methodology describes the use case template items, explains what each item mean, instructs how to fill them out and includes additional examples and tips. Use Case Analysis Process

The figure illustrates the analysis phase as a prelude of the SERVUS Design Methodology . As a first step of an analysis iteration loop is a set of preliminary use cases (UC) is identified, mostly by those thematic experts who drive the study.



Procedure of Use Case Analysis

The methodology proposes that use cases are initially described in structured natural language but already contain the list of requested resources. This description is the language, which is used in the use case discussion that takes place in workshops that are facilitated by the system analyst. Depending on the level of agreement that can be reached the iteration loop is entered again in order to refine or add new use cases.

In order to identify inconsistencies and check the completeness of the use case model, the system analyst may transform the semi-structural use case description into formal UML specifications. However, these UML diagrams should still be on a high abstraction level such that a discussion with the end-user is possible. It is the advantage of this formal transition step already in an early analysis phase to detect inconsistencies and missing information as quickly as possible. The UML specification helps to discuss and check the use cases together with the thematic experts.

However, in addition to the usual UML use cases they already comprise the links to the set of requested (information) resources, their representation forms and the requirements to create, read, write or delete them. Once an agreement is reached about the set of use case descriptions and related UML specifications it is then up to the system analyst to specify the resulting information model taking the resource model as a first guidance.

13.2 Use case template

This template is an extended and modified version of the use case template suggested in ISO 19119 [3], which in particular has been extended with a possibility to describe Requested Information Resources found suitable in an SDI setting.

Use Case Template	Description	Examples
Use Case Name	Name of the use case	Collect static sensor data, collect mobile sensor data
Use Case ID	Unique identifier of a use case	
Goal	Short description (max. 100 characters) of the goal to be achieved by a realization of the use case.	System generates alerts based on user observations exceeding threshold values
Summary	Textual description of the use case.	The user opens the browser which shows map-window with the water height after the tsunami event in the affected area
Actors	Primary Actor that initiates the use case execution, secondary actors and possibly other stakeholders.	Examples may be citizen, administrator or employee of a SDI agency
Requested Information Resources	Information category or object that is required to execute the use case or is being generated during the course of the use case execution. The requested information resource shall be listed together with its requested access mode (create, read, update or delete) or "manage" which encompasses all access modes.	<ul style="list-style-type: none"> • user observation (read) • user-specific effect (read, update) • alert (manage)
Preconditions	Description of the system/user status (statement) that is required to start the execution of the use case. Note that use cases can be linked to each other via "preconditions". This means, a precondition for a use case can be either an external event or another use case. In this case the use case ID should be provided in the field „preconditions“.	The user has opened the portal successfully.
Post conditions	Description of the system/user status (statement) that holds true after the successful execution of the use case.	Report is displayed on the screen.
Main success scenario	Numbered sequence of actions (use case workflow) to be carried out during the execution of the use case.	<ol style="list-style-type: none"> 1. User chooses assessment report. 2. He specifies one or more components (default should be all). 3. He sets a time-frame (last 24 hours,

		last week, last month) 4. The system shows a report as graphical visualisation.
Extensions or alternative parts (optional)	Extension of an action of the main success scenario. The action to be extended shall be referred to by its number (e.g. 1) appended by a letter (e.g. 1a). Alternate path through the main success scenario w.r.t. an identified action.	1a. The user defines the temporal extent b. The user defines an unavailable temporal extent. A new dialogue window opens and requires a new temporal extent. 4a. User can select to view report in different formats, e.g. tabular or graphical map
Priority	The priority of the use case to be considered when assessing its importance for a development cycle.	One of the following: <ul style="list-style-type: none"> • Must have: The system must implement this goal/ assumption to be accepted. • Should have: The system should implement this goal/ assumption: some deviation from the goal/assumption as stated may be acceptable. • Could have: The system should implement this goal/assumption, but may be accepted without it.

Table 2. Description of the Use Case Template

13.3 Use cases for the CITI-SENSE Platform

Figure 3-1 has presented a UML use case diagram for the CITI-SENSE Platform.

The use cases of the CITI-SENSE Platform are further elaborated in the sections below.

1. Platform-WP7-Observe-static: Observe - Collect static sensor data
2. Platform-WP7-Observe-mobile: Observe - Collect mobile sensor data
3. Platform-WP7-Observe-question: Observe – Questionnaire
4. Platform-WP7-Publish: Publish (observations/resources)
5. Platform-WP7-Discover: Discover (observations/resources)
6. Platform-WP7-Access: Query-Access
7. Platform-WP7-Transform: Transform
8. Platform-WP7-Visualise: Visualise
9. Platform-WP7-Analyse: Analyse (Compose, Process, Fusion (Product/App)...))
10. Platform-WP7-Notify: Notify (Ev.mgmt, alarm)
11. Platform-WP7-Security: Security and Rights Management
12. Platform-WP7-Manage-humans: Manage resources- humans
13. Platform-WP7-Manage-sensors: Manage resources- sensors
14. Platform-WP7-Manage-data: Manage resources- data
15. Platform-WP7-Manage-service: Manage resources- services

Use-case-name (table template)

Use Case Name:	Name
Use Case ID:	Area-WPYY-XXX
Goal:	<use case goal>
Summary:	<use case summary>
Actors:	All actors
Requested Information Resources:	Information model references
Preconditions:	Before the use case
Postconditions:	After the use case
Main success scenario:	Steps for primary scenario
Extensions:	Any extensions or variations
Priority:	Must, Should, Could

Observe - Collect static sensor data

Use Case Name:	Observe - Collect static sensor data
Use Case ID:	Platform-WP7-Observe-static
Goal:	Be able to collect and store all relevant static sensor data
Summary:	This use case represents the collection of all kinds of static sensor data
Actors:	Citizen and sensor-owner
Requested Information Resources:	A number of air quality and environmental related sensor data can be collected
Preconditions:	Sensor is registered in the system -through the manage-resource use case

Postconditions:	The collected sensor data is stored in the system
Main success scenario:	<ol style="list-style-type: none"> 1. Sensor platform collects relevant sensor data 2. Sensor data is uploaded to CITI-SENSE server in push or pull mode 3. Sensor data is available through the server
Extensions or alternative parts (optional):	Variations will exist depending on the interfaces with different sensor platforms
Priority:	Must

Observe - Collect mobile sensor data

Use Case Name:	Observe - Collect static sensor data
Use Case ID:	Platform-WP7-Observe-mobile
Goal:	Be able to collect and store all relevant mobile sensor and observation data
Summary:	This use case represents the collection of all kinds of mobile sensor data
Actors:	Citizen and sensor-provider
Requested Information Resources:	A number of air quality and environmental related sensor data can be collected with associated mobile locations.
Preconditions:	Sensor is registered in the system -through the manage-resource use case
Postconditions:	The collected mobile sensor data is stored in the system
Main success scenario:	<ol style="list-style-type: none"> 1. Mobile Sensor platform collects relevant sensor data 2. Mobile Sensor data is uploaded to CITI-SENSE server in push or pull mode 3. Sensor data is available through the server
Extensions or alternative parts (optional):	Variations will exist depending on the interfaces with different sensor platforms
Priority:	Must

Observe – Questionnaire

Use Case Name:	Observe – Questionnaire
Use Case ID:	Platform-WP7- Observe – Questionnaire
Goal:	Be able to collect questionnaire data from citizens and store the questions/answers in these as observations
Summary:	Provide a facility for storage of questionnaire data as observations from citizens
Actors:	Citizen
Requested Information Resources:	Information elements required are related to the generic model for a questionnaire
Preconditions:	An interaction for the creation of questionnaire answers has started
Postconditions:	Questionnaire answers have been stored
Extensions:	None
Priority:	Must

Publish (observations/resources)

Use Case Name:	Publish (observations/resources)
Use Case ID:	Platform-WP7-Publish
Goal:	Be able to make observations/resources permanent available, with different access/authorisation schemes
Summary:	Identify and ensure that stored observations are accessible for various user groups
Actors:	Citizen
Requested Information Resources:	Information elements required are the information elements that will be published emerging from static/mobil sensors and questionnaires. The information representations should in particular be related to OGC standards and later also as Linked Data/RDF.
Preconditions:	The observations have been made.
Postconditions:	The data has been stored by the relevant services.

Main success scenario:	1. Data is made available in suitable storage form 2. Data is made available through suitable access mechanisms, also suitable for later discovery.
Extensions:	None
Priority:	Should have – will be relevant in the coming phases of the project, and also include the relationship to GEOSS publications

Discover (observations/resources)

Use Case Name:	Discover (observations/resources)
Use Case ID:	Platform-WP7-Discover
Goal:	Be able to search and find relevant observations/resources
Summary:	Possibility to search and retrieve available observations and resources, based on suitable query criteria.
Actors:	Citizen
Requested Information Resources:	Information elements required are related to what is needed to find/discover already stored observations/resources.
Preconditions:	Available data is approved
Postconditions:	Relevant observations/resources have been discovered
Main success scenario:	1. Data is made available 2. Data is being processed
Extensions:	None
Priority:	Should have – will be relevant in the coming phases of the project, and also include the relationship to GEOSS publications

Query-Access

Use Case Name:	Query-Access
Use Case ID:	Platform-WP7-Access
Goal:	Be able to access the observations/resources in the system, possibly

	through a suitable query mechanism. The architecture will allow for different representation formats to be used, in particular related to OGC standards and later also as Linked Data/RDF.
Summary:	Provide access to the available observations/resources through suitable API and query interface
Actors:	Citizen – through appropriate data access interface, Programmers – for accessing the data through an appropriate API (WFS-T)
Requested Information Resources:	Information elements required are determined by the data elements of relevant observations/resources
Preconditions:	Available data is approved
Postconditions:	The goal has been reached.
Main success scenario:	1. Data is made available 2. Data is being provided to be visualised, analysed and/or processed.
Extensions:	None
Priority:	Must

Transform

Use Case Name:	Transform
Use Case ID:	Platform-WP7-Transform
Goal:	Be able to transform between different representation formats for observations/resources – in order to provide support for multiple input and output formats – (xml, JSON, RDF, ... etc)
Summary:	This use case represents
Actors:	Citizens – need – through programmers support
Requested Information Resources:	Information elements required are the foundation elements needed for a source to target mapping and conversion.
Preconditions:	Available data is approved
Postconditions:	The data has been transformed into suitable target representations
Main success scenario:	1. Data is made available 2. Data is being processed to be converted from source to target
Extensions:	None

Priority:	Must for XML/JSON – should/could have for RDF/Linked Data
------------------	---

Visualise

Use Case Name:	Visualise
Use Case ID:	Platform-WP7-Visualise
Goal:	Be able to visualise and present the observations/resources from the CITI-SENSE storage services in various suitable forms.
Summary:	This use case represents the ability to visualise the relevant observation data in various ways.
Actors:	Citizen
Requested Information Resources:	Information elements required are gathered from the various types of sensor information,
Preconditions:	Observation data is available in the server.
Postconditions:	Data is being visualised in appropriate forms
Extensions:	None
Priority:	Must

Analyze

Use Case Name:	Analyse (Compose, Process, Fusion (Product/App...)A
Use Case ID:	Platform-WP7-Analyze
Goal:	Be able to provide relevant analysis/processing capabilities
Summary:	This use case represents the possibility to analyse the observation data in different ways
Actors:	Citizen and analysts
Requested Information	Information elements required are related to both existing observation data and data from other sources to be related to.

Resources:	
Preconditions:	Available data is approved
Postconditions:	Appropriate analysis has been completed
Extensions:	None
Priority:	Should have

Notify

Use Case Name:	Notify ((Ev.management, alarm, with publish/subscribe possibilities)
Use Case ID:	Platform-WP7-Notify
Goal:	Be able to notify about situations of interests
Summary:	This use case represents an ability to notify interested users/services about events and situations – based on a publish/subscribe schemes. Conditions might be handled through complex event processing combining various criteria.
Actors:	Citizens and supporting services
Requested Information Resources:	Information elements required will depend on the particular situations -
Preconditions:	Available data is approved
Postconditions:	The goal has been reached.
Main success scenario:	1. Data is made available 2. Data is being processed
Extensions:	None
Priority:	Required

Security and Rights Management

Use Case Name:	Security and Rights Management
Use Case ID:	Platform-WP7-Security-DRM
Goal:	Be able to support suitable security measures (authentication/authorisation) and Digital Rights Management (DRM)
Summary:	This use case is transversal across some existing use cases, providing adequate support for need security/safety and right management handling.
Actors:	Citizen
Requested Information Resources:	Information elements required are related to user profile and availability of the actual observation data to be used.
Preconditions:	Relevant observation data is available
Postconditions:	Relevant security schemes has been defined
Main success scenario:	1. Data is made available – only if security access allows for it 2. Data is being processed
Extensions:	None
Priority:	Should have

Manage sensors

Use Case Name:	Manage sensors
Use Case ID:	Platform-WP7-Manage-sensors
Actors:	Sensor provider
Goal:	Register sensors in the system as preparation for further collection of sensor data.
Summary description:	The sensor is registered in the system. The registering of sensors also includes registering of sensor metadata, i.e., the capabilities of the sensor according to an agreed upon format to be defined.
Actors:	Sensor platform manager
Preconditions:	External sensor available.
Postconditions:	External sensor registered.
Flow of events:	1. Sensor platform manager: Invokes the register sensor 2. System: Sensor is registered.
Exceptions:	None
Priority:	Must

Manage resources: humans/sensors/data/services

Use Case Name:	Manage
Use Case ID:	Platform-WP7-Manage Platform-WP7-Manage-humans Platform-WP7-Manage-sensors Platform-WP7-Manage-data Platform-WP7-Manage-services
Goal:	Be able to manage the resources through their lifecycle, including different measures for humans, sensors, data and services.
Summary:	This use case represents all different resources that needs to be managed by the platform
Actors:	Citizen and API users
Requested Information Resources:	Information elements required are the metadata needed to manage the different resource types, i.e. humans, sensors, data and services
Preconditions:	Needed data and other resources have been made available
Postconditions:	Management information about relevant resources has been created/updated/deleted.
Extensions:	None
Priority:	Must

14 References

Table 14-1 References

01	Asadi, M. and Ramsin, R. (2008). MDA-based Methodologies: An Analytical Survey. In: Schieferdecker, I. and Hartman, A. (eds.): European Conference on Model-driven Architecture – Foundations and Applications (ECMDA-FA) 2008, LNCS 5095, pp. 419-431, Springer-Verlag Berlin Heidelberg.
02	Berners-Lee, T., Hendler, J. and Lassila, O. (2001). The Semantic Web. Scientific American Magazine, 2001.
03	Berre, A. (2014). ISO 19119:2 revision , www.isotc211.org
04	Bizer, C., Heath, T. and Berners-Lee, T. (2009). Linked Data - The Story So Far, International Journal on Semantic Web and Information Systems, Vol 5, Issue 3.
05	CEN/TR –15449 (2012). Geographic information — Spatial Data Infrastructures — Part 4 : Service Centric View. CEN/TC 287 WI 00287030, Secretariat: BSi.
06	Simonis, I. (ed.) (2008). OGC® Sensor Web Enablement Architecture”. OGC Best Practices Document 06-021r4, Version 0.4.0. Available at: http://www.opengeospatial.org/standards/bp
07	ISO/IEC 10746-1:1998. Information technology - Open Distributed Processing – Foundations.