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 D7.5 - Part 1: Platform Specification

D7.5: CITI-SENSE Platform and architecture Version 3 - Part 1: Specification

Work Package 7

Date: 30.11.2015

Version: 3.10

Leading Beneficiary:	SINTEF, Snowflake
Editor(s):	Arne J. Berre (SINTEF), Richard Rombouts (Snowflake)
Author(s) (alphabetically):	Arne J. Berre (SINTEF), Alberto Fernandez (Sensing&Control), Nicolas Ferry (SINTEF), Tom Forbes (Snowflake), Nicolas Ferry (SINTEF), Sungchul Hong (KICT), Seonho Kim (Saltlux), Daniele Miorandi (U-Hopper), Richard Rombouts (Snowflake), Alberto Olivares (Snowflake), Maja Pokric (Dunavnet), Dumitru Roman (SINTEF), Miha Smolnikar (JSI), Andrei Tamilin (U-Hopper), Matevž Vučnik (JSI)
Dissemination level:	Public

Versioning and contribution history

Version	Date issued	Description	Contributors
2.00	18.12.2014	Final version 2.0 – based on experience and review updates	Arne J. Berre
3.01	15.04.2015	Initial version 3.01	Arne J. Berre
3.02	15.06.2015	Update on Data and Product Services	Richard Rombouts
3.03	15.06.2015	Update on CITI-SENSE Data Model	Richard Rombouts
3.04	15.07.2015	Update on Sensor Services	Nicolas Ferry
3.05	20.10.2015	Update on Social Sensors	Seonho Kim
3.05	20.10.2015	Update on Linked Data Services	Seonho Kim
3.06	25.10.2015	Update on Visualisation and Interaction services	Andrei Tamlin
3.07	28.10.2015	Update on Ubiquitous Public Access services	Sungchul Hong
3.08	05.11.2015	Update on Security Services	Richard Rombouts
3.08	05.11.2015	Update on GEOSS Integration	Arne J. Berre
3.09	20.11.2015	Update on Data and Product Services	Richard Rombouts
3.10	30.11.2015	Integration of review comments	Arne J. Berre

Peer review summary

Internal review 1			
Reviewer	Mirjam Fredriksen (NILU)		
Received for review	November 20th, 2015	**Date of review**	November 27th, 2015

Internal review 2			
Reviewer	Leonardo Santiago (Ateknea)		
Received for review	November 20th, 2015	**Date of review**	November 28th, 2015

Executive Summary

This deliverable D7.5 provides the version 3.0 of the CITI-SENSE Platform and Architecture. This document extends on the content of deliverable D7.4.

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 updates in D7.5 compared to D7.4 are in particular an update of details of the data access in the chapters on data products and services and the CITI-SENSE data model, and also updates on the specifications of visualisation components, social sensors, UPA and Linked Data. D7.5 has been split into two parts, with part 1 focusing on the overall architecture and specifications and part 2 focusing on the operational aspects including more details of the various sensor platforms and app usages of the CITI-SENSE architecture and platform.

The D7.5 CITI-SENSE Platform and Architecture version 3.0 presents the CITI-SENSE platform with the following content: 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. Chapter 3 analyses the CITI-SENSE Platform and Architecture requirements described earlier in D7.2. The Work packages 2 and 3 based requirements from the CITI-SENSE Empowerment Initiatives which were analysed in D7.2 are here further described with respect to mappings to generic platform use cases. The baseline for the use case template is found in Annex A. 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 D7.5 part 2 for the usage of the SEDS WFS-T server. Chapter 7 on Sensor services describes the SensApp sensor storage services. These are further documented in D7.5 part 2 for the usage of SensApp. Chapter 8 describes components for the development of Visualisation and User Interaction services which is further documented in D7.5 part 2.

The previous version of the CITI-SENSE Platform and Architecture has been enhanced in a number of new areas extending the D7.5 version, such as:

- The CITI-SENSE UML data model has been extended to include new attributes such as the ParticipantID, CAQI and Global CAQI values in an updated version of the CITI-SENSE data model. Chapter 6 contains a detailed overview of the latest version.
- To protect the Platform, data and users from unwanted use, additional security layers have been added. Further details on these new security layers can be found in chapter 9.

The further relationship to the ISO 19154 Ubiquitous public access services is described in chapter 10. The strategy for GEOSS integration, including the potential use of GEOSS community portal architectures is described in chapter 11. The planned support for social sensors and Linked (Open) Data services is described in chapter 12 and 13, respectively. The existing D7.4 material has been included in this report, so it thus not necessary to read both the D7.5 and the D7.4 reports, i.e. this is version 3.0 evolving from the D7. 4 report.

Further services for consideration, such as communication and composition services and participation and empowerment services are introduced in chapter 14. A final version of the CITI-SENSE Platform and Architecture is planned for October 2016, for M48 of the project, as D7.6 CITI-SENSE Platform and Architecture 4.0.

Table of contents

Index of figures

Index of Tables

Abbreviations and Acronyms

Table 0-1 Abbreviations and Acronyms

Abbreviation / Acronym	Description
CAQI	Common Air Quality Index
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, D7.5 “Platform and Architecture – version 3”, provides the version 3 of the specification and description for the CITI-SENSE architecture and platform. D7.5 includes and extends the baseline in D7.4 with a further refinement in various areas, in particular for the realisation of the CITI-SENSE Data Model, Management and security services, Ubiquitous public access services, Social sensors and Linked Open Data services.

The updates in D7.5 compared to D7.4 are in particular an update of details of the data access in the chapters on data products and services and the CITI-SENSE data model, and also update the specifications of visualisation components, social sensors, UPA and Linked Data.

In order to support easier reading of the D7.5 document, the document has been split into two parts as follows:

- D7.5 part 1 - Platform Specification - provides an overview of the overall design and architecture;
- D7.5 part 2 - Platform Operation - contains detailed information about the operational use of the data platforms, with connected sensor platforms and apps.

The further evolution from this D7.5 will be the final D7.6 “Platform and architecture version 4” for M48.

D7.5 part 1 contains the following chapters:

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 reflect 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 earlier 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 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 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 D7.5 part 2 chapter 2 section 2.1 “Data Ingestion Services”.

The CITI-SENSE Data Model is described in more detail in chapter 6. This chapter provides more detail about the actual data model representation for the data and product services described in chapter 5, and is also elaborated in more detail in D7.5 part 2 chapter 2 “Data Publication Services “. The D7.5 document describes the new version 2.2 of the CITI-SENSE Data model.

Chapter 7 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 D7.5 part 2 on “Access to SensApp server” for the usage of SensApp.

Chapter 8 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 D7.5 parts 2 on “Visualisation Components”.

The elaboration of Management and security services with a strategy for security support is described in chapter 9. This involves in particular a relationship to the OGC standards approach for authentication and access management using SAML and other web standard protocols.

The further relationship to the ISO 19154 Ubiquitous public access services (UPA) is described in chapter 10. This is a new standard from ISO/TC211 on Geographic Information focusing on a reference architecture for the integration of geospatial systems with systems to handle distributed and mobile context management. The CITI-SENSE partner KICT from South Korea has been involved in the development and standardisation of ISO 19154 – and the CITI-SENSE project will endeavour to validate and take advantage of this standard reference architecture.

The strategy for GEOSS integration is described in chapter 11. It is an objective of the CITI-SENSE project to provide its data sets and sensor data as GEOSS resources. The CITI-SENSE project is also considering how to potentially use the GEOSS architecture to support a citizens’ observatories community portal, and also how CITI-SENSE can provide input for future GEOSS infrastructure for mobile sensing and crowd sourcing services through the GEOSS AIP activities.

The concept of social sensors is introduced in chapter 12. Social sensors provides a possibility to extract sensing information from social media, in particular related to citizens sentiment about events and happenings at certain locations related to environmental aspects. The CITI-SENSE “sister” project WeSenseIt has a separate work package activity on social sensors, on social information extraction from media, and the CITI-SENSE partner Saltlux is providing services in the area which the CITI-SENSE project will seek to take advantage of.

The support for Linked (Open) Data services is described in chapter 13. Linked Open Data provides a technology that makes it easier to integrate data from many sources – and there is currently many initiatives to do this for environmental data. The CITI-SENSE project will seek synergy with parallel projects, involving CITI-SENSE partners such as Saltlux and SINTEF, in order to make CITI-SENSE data available also as Linked Open Data – and also to relate this to GEOSS.

Chapter 14 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 “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.

Annex B "CITI-SENSE XML Application Schema" contains the XML Schema for the CITI-SENSE data model.

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)[1] 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[2].

Different Spatial Data Infrastructure standard organisations and initiatives like CEN/TC287 TR 15449 SDI[3] , 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 2-1 illustrates the main focus of each of the five RM-ODP viewpoints, and the main question that each viewpoint addresses.

[1] ISO/IEC 10746-1:1998. Information technology - Open Distributed Processing – Foundations.
[2] http://www.omg.org/mda/presentations.htm
[3] CEN/TR –15449 (2012). Geographic information — Spatial Data Infrastructures — Part 4 : Service Centric View. CEN/TC 287 WI 00287030, Secretariat: BSi.

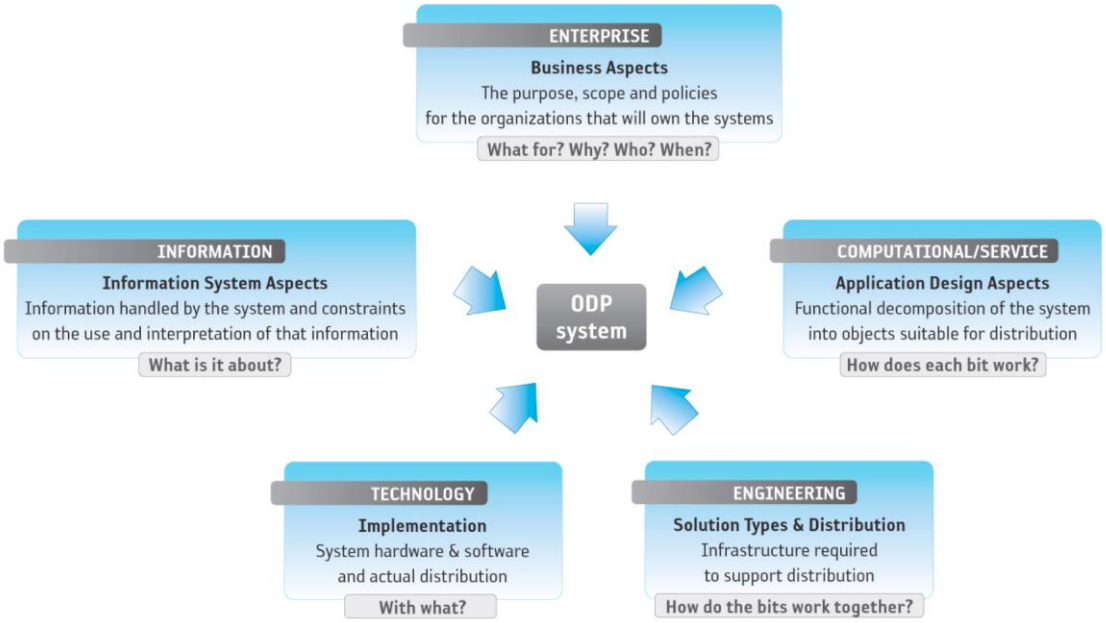


Figure 2-1 ISO RM-ODP viewpoints

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 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 are in the following described according to the various ODP viewpoints. This is aligned also with the recommendations from the CEN/TC287 TR 15449 on Spatial Data Infrastructures, following these viewpoints as illustrated in Figure 2-2, where description support for different kinds of interoperability also is illustrated.

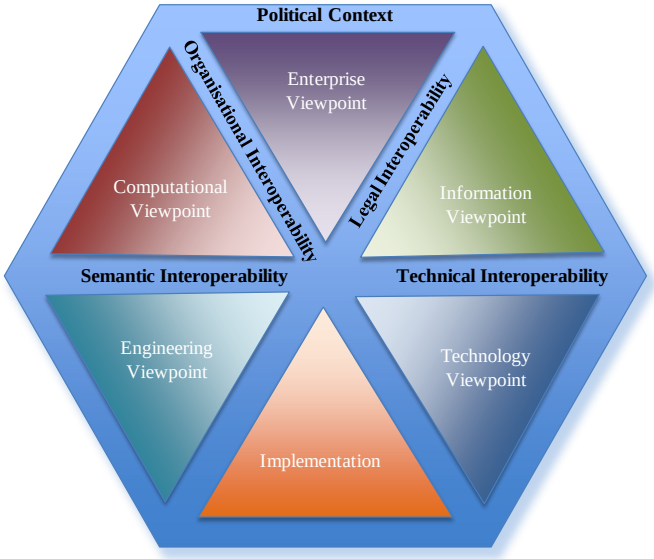


Figure 2-2 RM-ODP viewpoints and Interoperability description support

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 them 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.

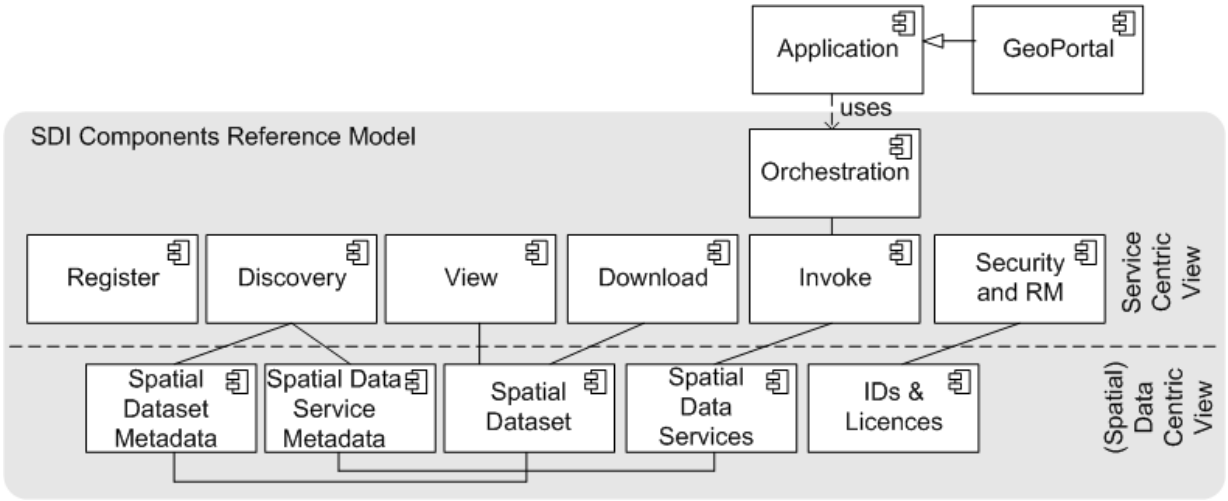


Figure 2-3 Life cycle services defined by INSPIRE

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, the following 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.

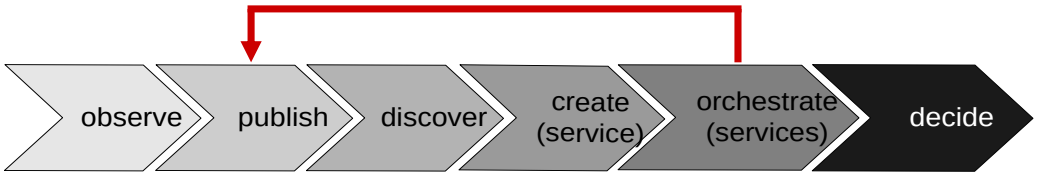


Figure 2-4 Added value chain of SDI knowledge generation

2.7.3 Overview of Stakeholders

The following roles identified are applied 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, 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.

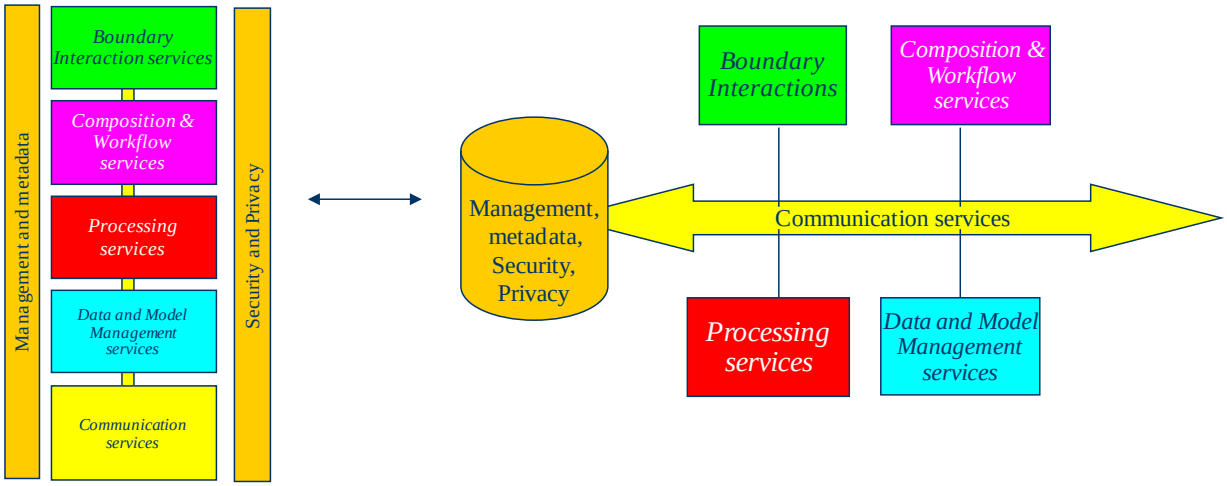


Figure 2-5 Relationships of services in both a layered and a bus architecture

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 Citizens Observatory Software Toolbox

Towards the end of the CITI-SENSE project (M48), the CITI-SENSE Platform and architecture will be offered as a Citizens Observatory Software Toolbox, with a packaging of the various services of the platform.

This will include the data server access layer, the visualisation components, the sensor data handling as well as frameworks for the app development with data access.

2.10 CITI-SENSE Generic Lifecycle Services

The initial analysis of the requirements of the various work packages 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.

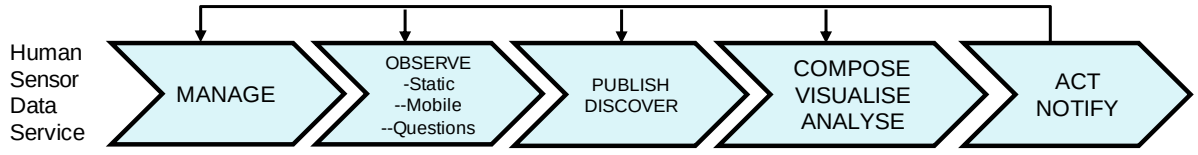


Figure 2-6 CITI-SENSE Generic life cycle services.

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 citizens' 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 citizens' 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

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
R4	Collect data from perception data: surveys with questionnaires	WP2, WP3
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 – Urban Quality

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 Spaces

WP3 will design, implement and evaluate a case study in public spaces in Vitoria (Spain). The main aim of this initiative 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)

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 – School Indoor Quality

Visualization of real time data for a number of different measurements: Temperature, Relative humidity, CO2, NO_2, 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 these case studies 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 have 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 using different kinds of web portal and social media support will be valuable to support that.

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.

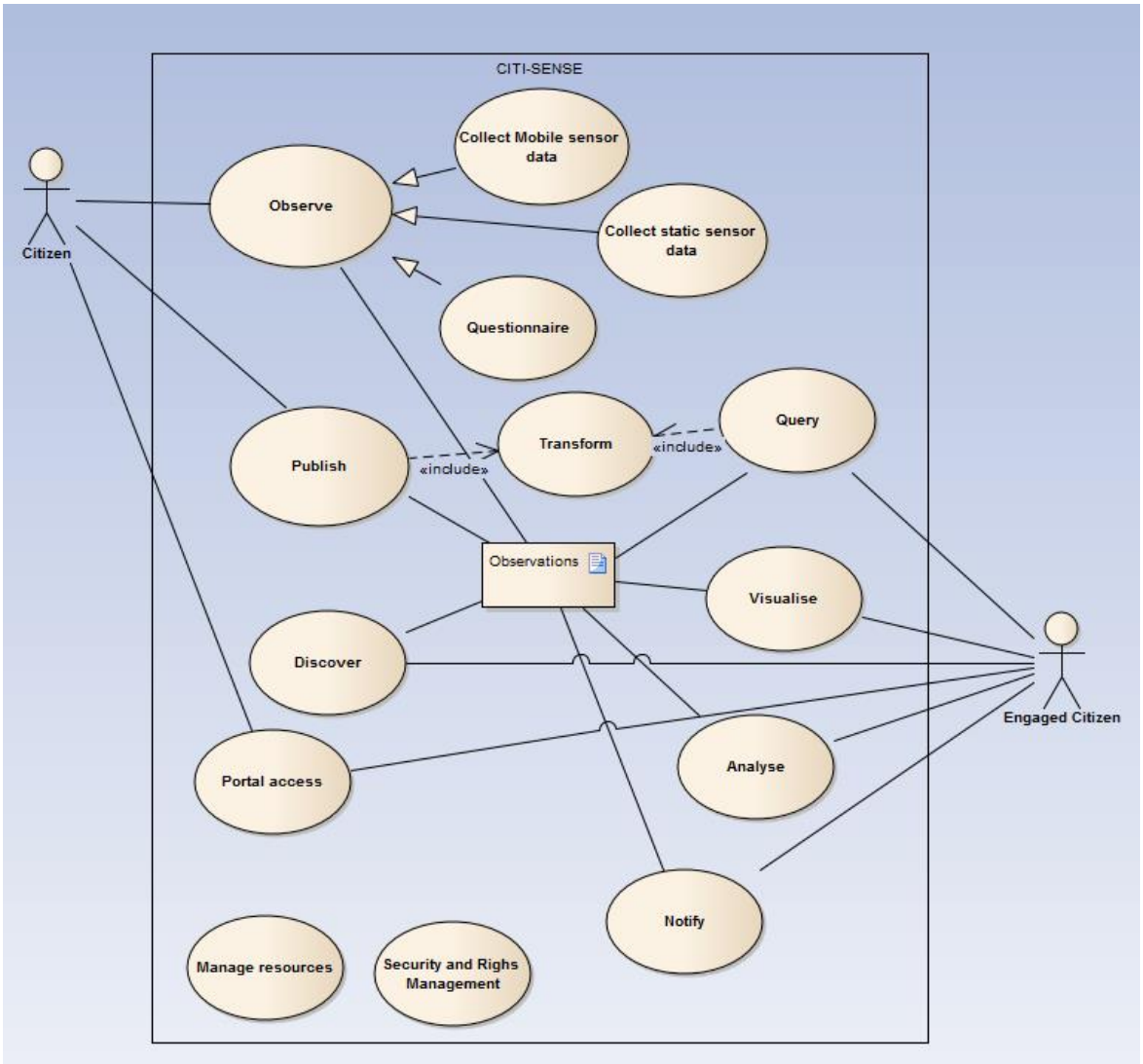


Figure 3-1 CITI-SENSE Platform use cases

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

ID	Requirement	Platform use case
R1	Collect data from static outdoor sensors: NO, NO_2, CO, O_3,PM, Noise, Temperature, Humidity, PAH	Collect static sensor 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 (Tmrt), 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): LAeq, L90 LAmax and LAmin	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 later in this report. The initial priority is on use cases to collect and visualise observation data, but some further life cycle service support is foreseen for the last phase of the project.

4 CITI-SENSE Architecture

The CITI-SENSE project aims to develop “citizens' observatories” to empower citizens to contribute and participate in environmental governance, to enable, 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 viewpoint[4].

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.

[4] http://rm-odp.wikispaces.com/ODP+Viewpoints

4.2 Enterprise Viewpoint

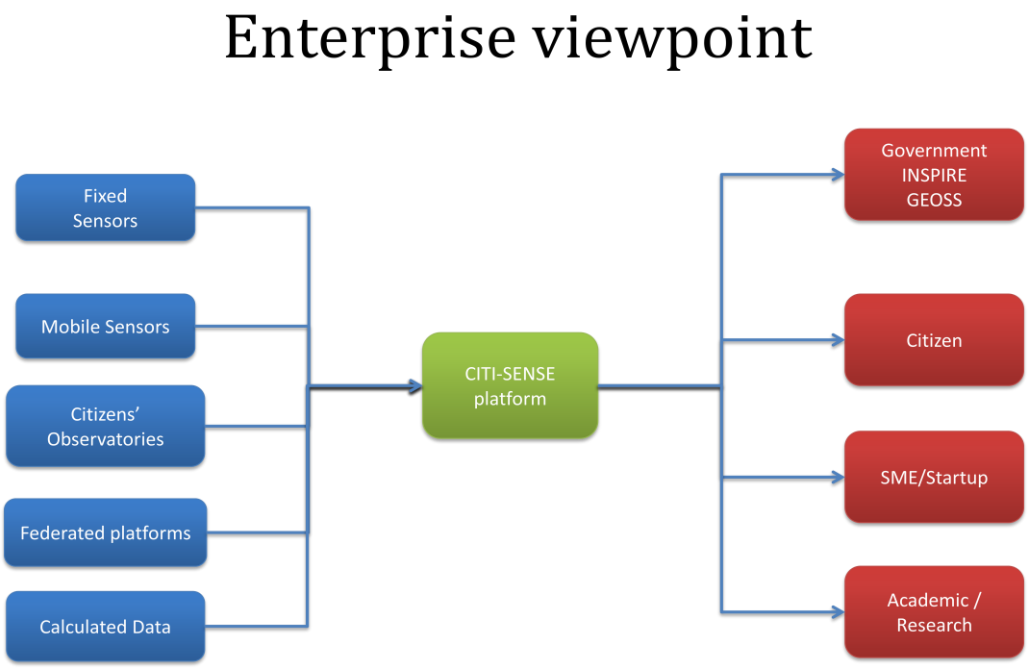


Figure 4-1 Enterprise viewpoint

4.2.1 Sensing Systems

At the data collection end of the system (the blue boxes in Figure 4-1), there are a 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.

Type of Organisation	Use of CITI-SENSE	Description
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.

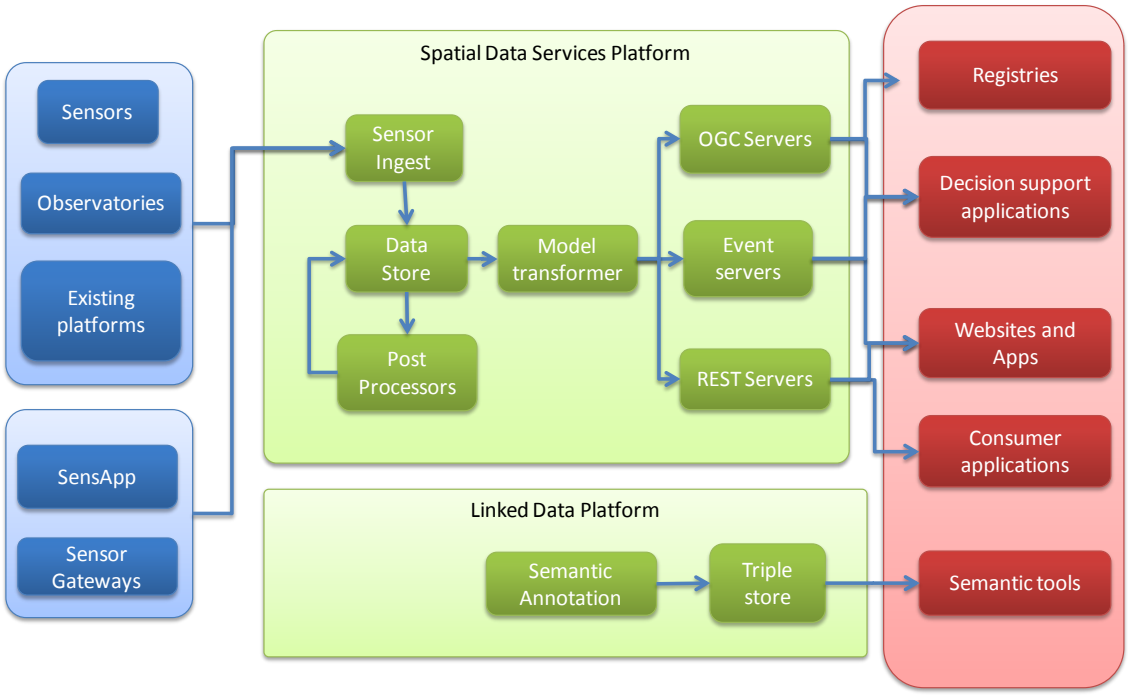


Figure 4-2 Computational Viewpoint

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

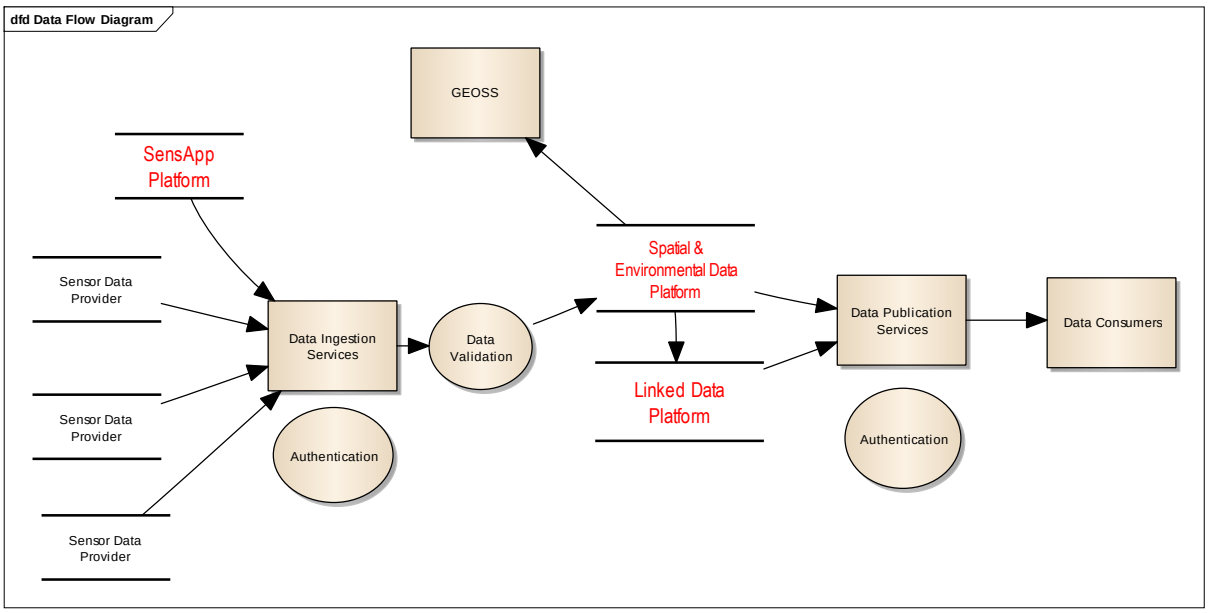


Figure 4-3 CITI-SENSE Platform data flow

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.

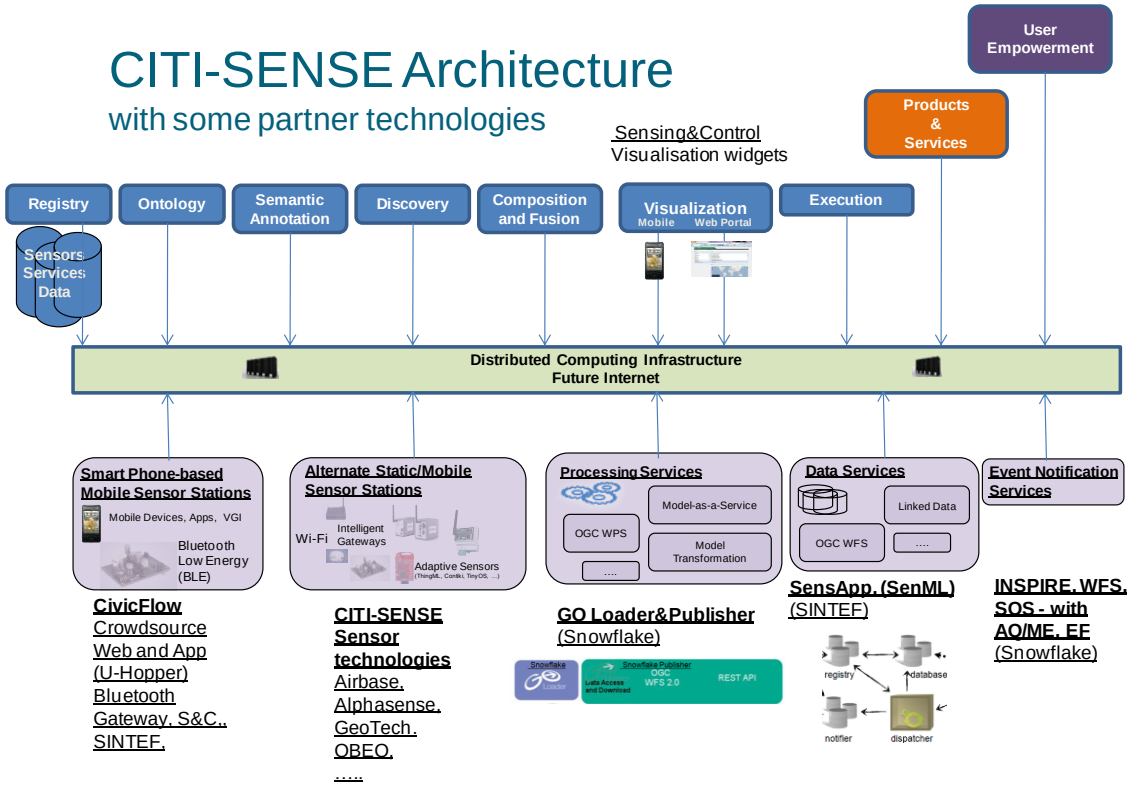


Figure 4-4 CITI-SENSE Architecture as a service oriented architecture

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.

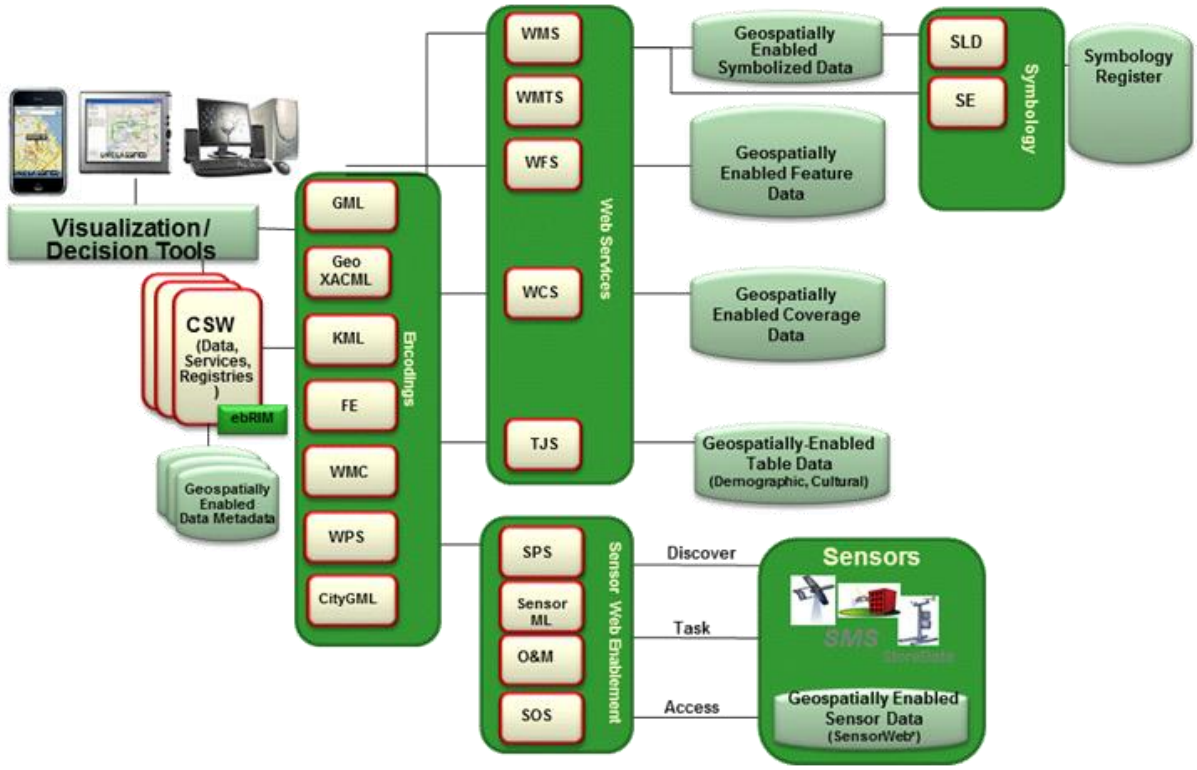


Figure 4-5 OGC Web Services Architecture

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 website[5] and the OGC Reference Model[6] 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 **Catalogue 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**
    - **Web Map Service WMS**: also published as ISO 19128, provides three operations (GetCapabilities, GetMap, and GetFeatureInfo) in support of the creation and display of registered and superimposed map-like views of information that come

[5] www.opengeospatial.org and http://www.opengeospatial.org/standards/is
[6] http://www.opengeospatial.org/standards/orm

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 geo data 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 catalogue 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 Catalogue standard defines common interfaces to discover, browse, and query meta data 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 the 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.
    - The support of both Request/Reply and Publish/Subscribe patterns is needed to fulfil the requirements of many services that have a pubsub element.
- 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.

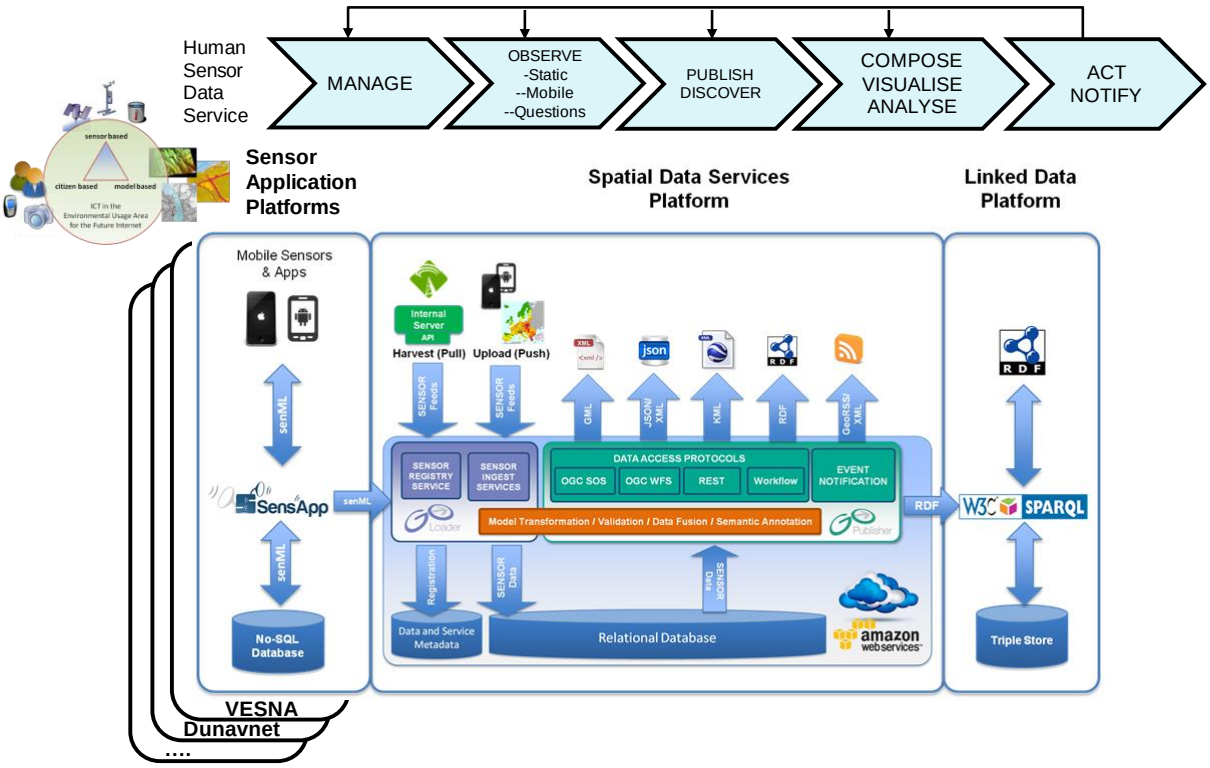


Figure 4-6 CITI-SENSE Platform

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.

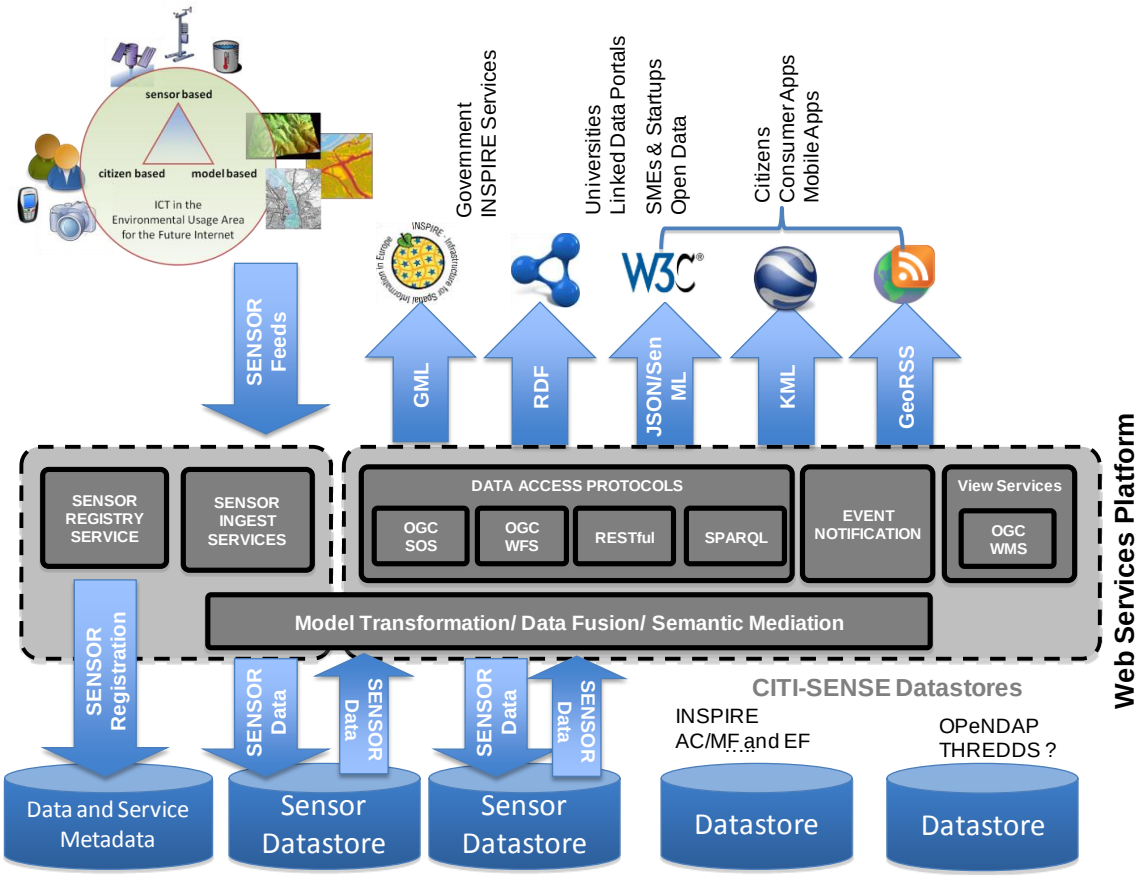


Figure 4-7 Architecture for sensor and data management

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.

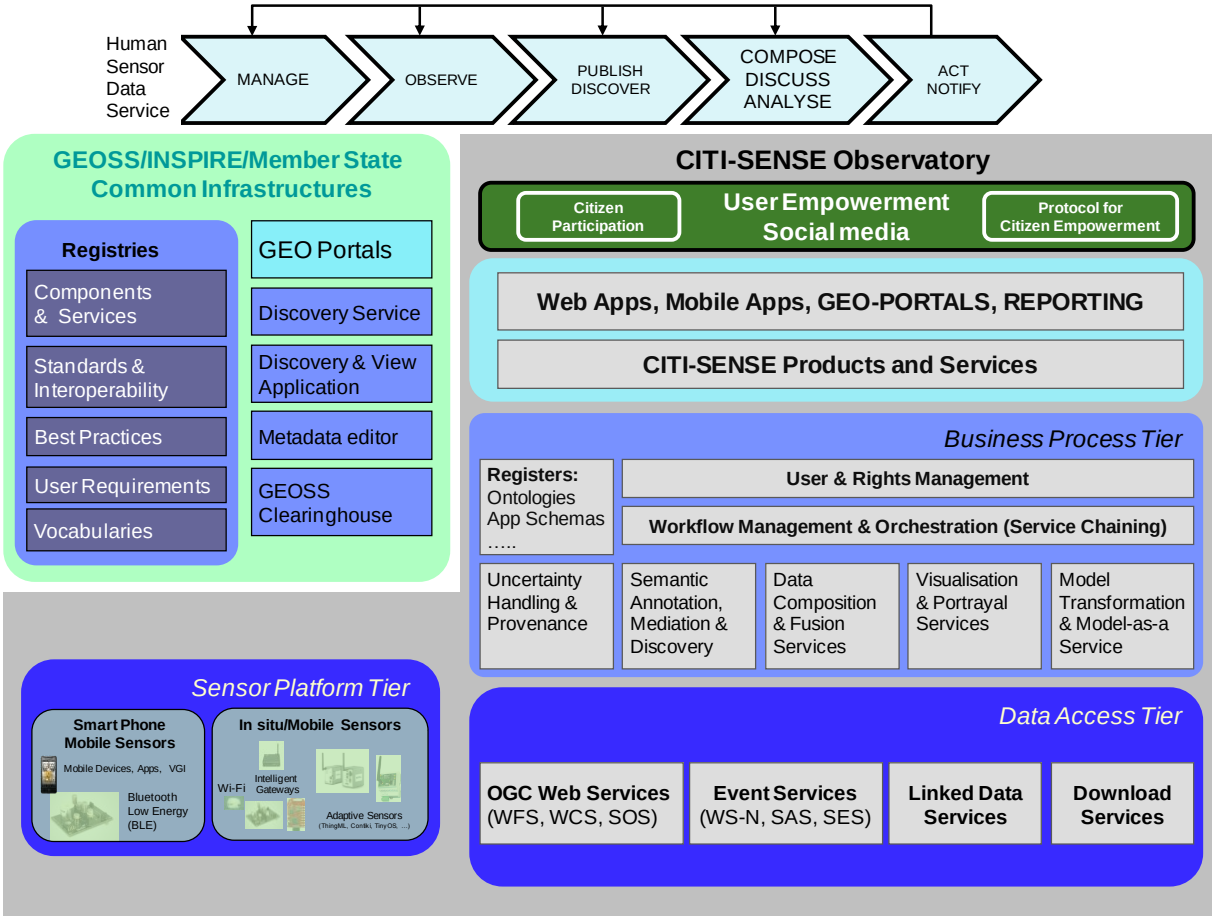


Figure 4-8 CITI-SENSE adaptation of GEOSS architecture

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 SensApp 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.

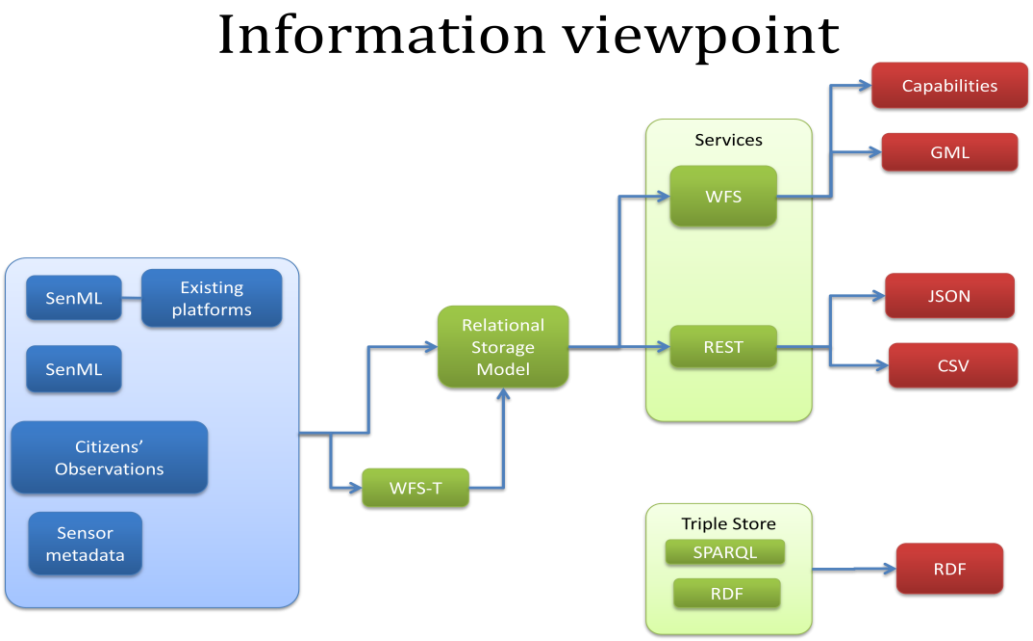


Figure 4-9 Information viewpoint

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)[7], OASIS[8] and the Open Geospatial Consortium (OGC)[9] 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:

[7] www.w3c.org
[8] www.oasis-open.org
[9] www.opengeospatial.org

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 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
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.
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.

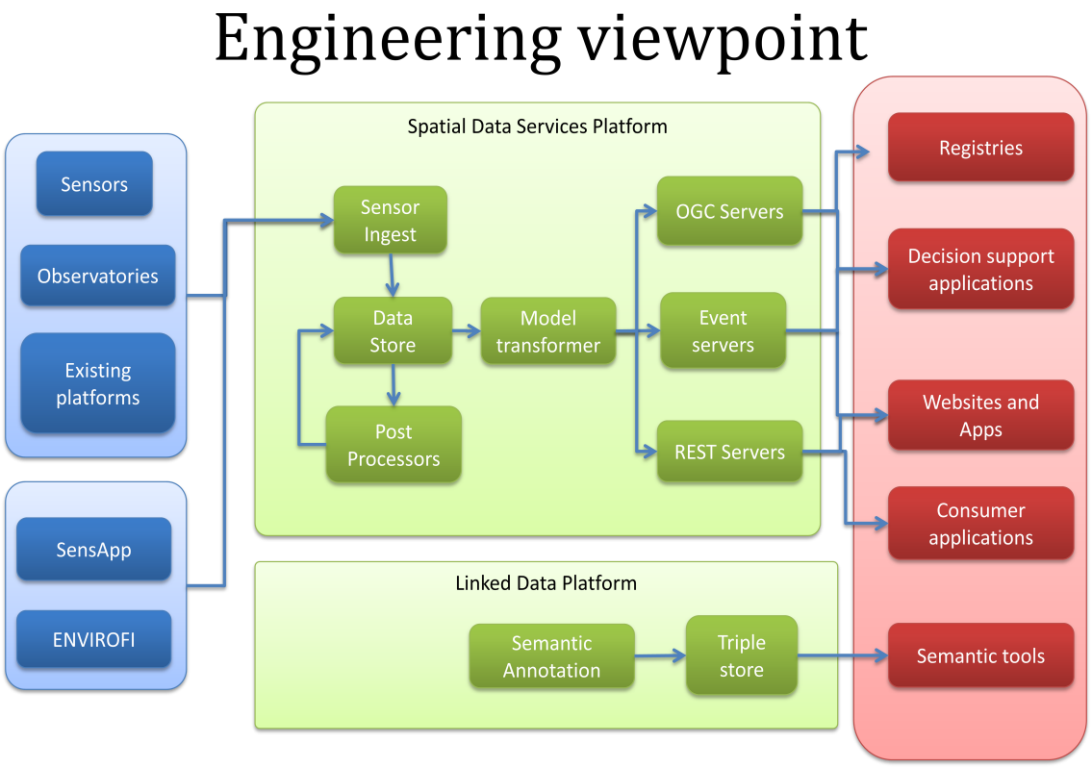


Figure 4-10 Engineering viewpoint

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, 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 and CSV 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.

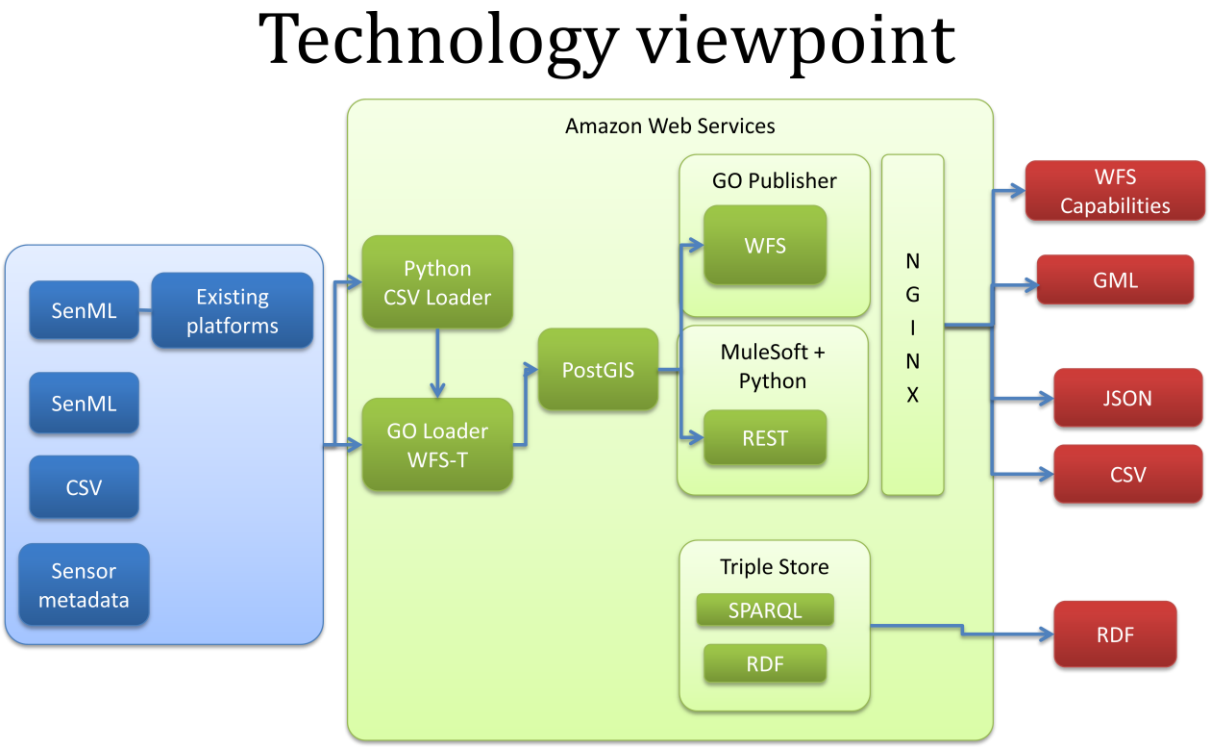


Figure 4-11 Technology viewpoint

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. 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 is 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.

MuleSoft Anypoint

MuleSoft is a US-based company that provides integration software for connecting applications, data sources and APIs, in the cloud or on-premises. Within the CITI-SENSE SEDS Platform, their free version of their MuleSoft Anypoint product is used to implement the REST web services that serve sensor data in the CSV and JSON formats.

Python

Where no off-the-shelf software solution could be found to implement a requirement, bespoke code was written. Python is a widely used, general purpose, programming language and allows for rapid coding and deployment in small and large technical architectures.

In the SEDS Platform Python has been used to develop a component that periodically pulls sensor data (in CSV) from two separate sensor data providers and transforms the CSV dataset to a WFS Post request, which is posted to the Ingestion Service (WFS-T).

Additionally, bespoke Python code was developed to support one specific data publishing scenario, which required a large amount of data to be published.

NGINX

NGINX, pronounced as “engine-x”, is a free, open-source, high-performance HTTP server and reverse proxy. It has a global user base and is being implemented in numerous applications.

In the SEDS Platform NGINX is used to fulfil the role as reverse proxy server. In computer networks a reverse proxy retrieves resources on behalf of a client from one or more servers. The resources are then returned to the client as though they originated from the proxy server itself.

The main benefit of implementing a reverse proxy is that it hides the existence and characteristics of the back-end servers, thus improving ease-of-access for client application s and adds additional security to the SEDS Platform.

The NGINX product comes with additional functionality for load balancing and data caching, both of which have been used in the implementation of the SEDS Platform.

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 (e.g., 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 CO_2, NO, PM, etc., some data providers also provide other types of data. Examples of these are 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
- Citizens
- 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 Service Oriented 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.

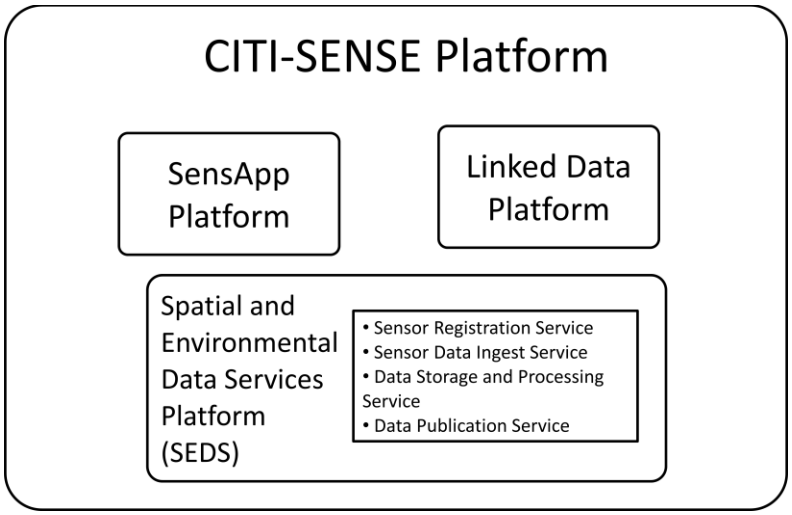


Figure 5-1 The CITI-SENSE Platform and its components

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.

The Sensor Application Platform and Linked Data Platform will be discussed in more detail in chapters 7 and 13, respectively.

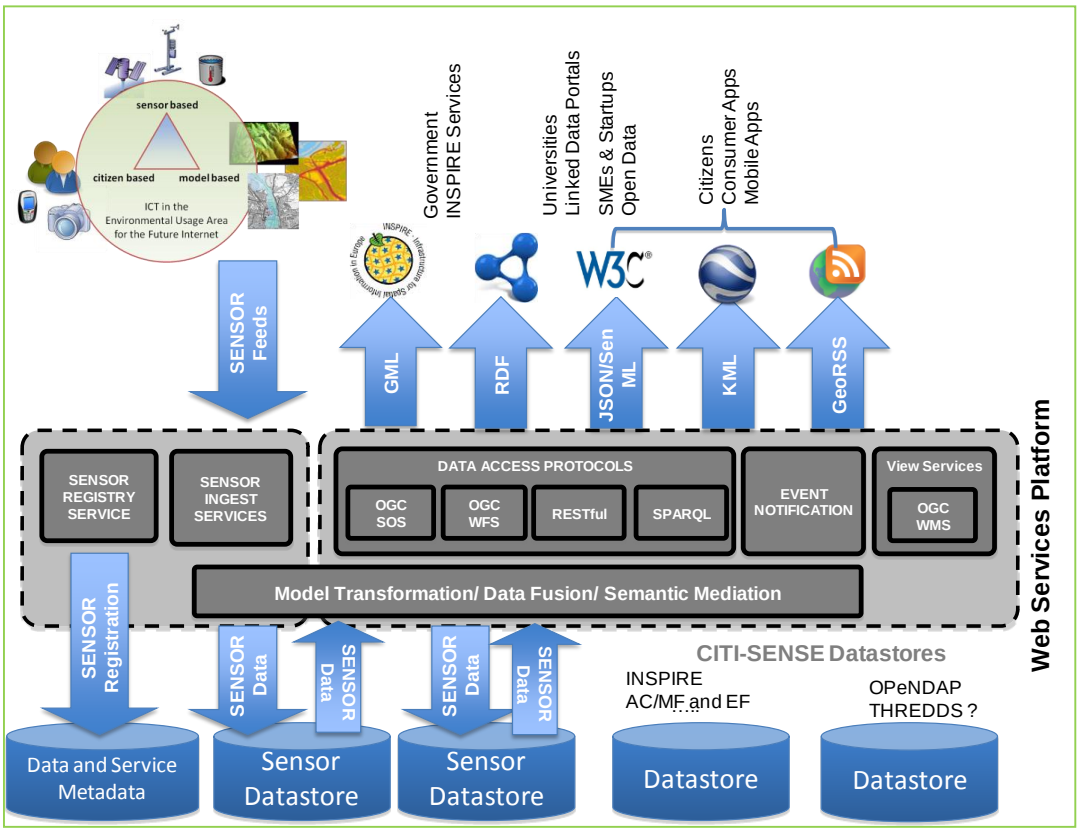


Figure 5-2 Spatial data services

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 Platform. The SEDS Platform 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 (near) real-time.
2. **Pull** - the SEDS Data Repository actively pulls data from the sensor data provider network. This can done periodically or in (near) real-time.

Which mechanism, push or pull, 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/AQMesh	Pull	Data provider provides an external FTP server on which datasets are stored. The Sensor Data Ingest service periodically harvests the data from this FTP server.
AirMonitors	Pull	Data Provider provides an external FTP server on which datasets are stored. The Sensor Data Ingest service periodically harvests the data from this FTP server.
SensApp	Push	Data Provider provides a web service that pushes data as SenML in a HTTP POST operation to the SEDS Platform.
City Air	Push	Data Provider provides a web service that pushes data as XML in a HTTP POST operation to the SEDS Platform

DNET	Push	Provides a web service (API) that publishes data as CSV or JSON.
JSI	Push	Data Provider provides a web service that pushes data as XML in a HTTP POST operation to the SEDS Platform.
CVUT	Push	Data Provider provides a web service that pushes data as JSON in a HTTP POST operation to the SEDS Platform.
U-Hopper	Push	Data Provider provides a web service that pushes data as XML in a HTTP POST operation to the SEDS Platform.
LEO/Atekne	Push	Data Provider provides a web service that pushes data as XML in a HTTP POST operation to the SEDS Platform.

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 (e.g. XML to SQL, SQL to XML)
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 (e.g., 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 (WFS) standard web interface (ISO-19142) is implemented. This interface allows clients to query and request data. It also provides so-called “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 returned through the WFS service is in GML. GML is a standard data encoding for exchanging geographical data. 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 provides a number of lightweight web services suitable for SMEs or citizen users. These are based on RESTful principles and will provide content in well supported formats including CSV 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.

The JSON data format is commonly used by web developers and is often used 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 aims 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.

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:

- **AWS EC2** (Amazon Elastic Compute Cloud), which provides virtual computers on which the CITI-SENSE software components and web services are deployed and managed.
- **AWS RDS** (Relational Database Service), which provides a pre-installed and configured relational database.
- **AWS S3** (Simple Storage Service), which provides data storage capacity.

Two identical systems have been deployed with different purposes:

- **TEST Environment**: This system was released on 24th August 2015. This environment has been used for development purposes and for testing future improvements of the SEDS Platform.
- **PRODUCTION Environment**: This system was released on 1st September 2015. This environment has been used to load, store and publish real data from sensor providers.

The following AWS instances have been created on EC2 and RDS within the TEST and PROD environments:

Table 5-2 WS instances on EC-2 and RDS

Server	Purpose	Instance Type	Specification
EC2			
WFS-1	Primary WFS service - provides OGC WFS and REST service endpoint to publish data in the XML, CSV and JSON encodings.	T2.medium	2 vCPU, 4 GB RAM
WFS-2	Secondary WFS service - provides OGC WFS and REST service endpoint to publish data in the XML, CSV and JSON encodings. Resides in a separate subnet from WFS-1.	T2.medium	2 vCPU, 4 GB RAM
WFST-1	Primary WFS-T service - provides data ingestion services for data providers.	T2.micro	1 vCPU, 1 GB RAM
WFST-2	Secondary WFS-T service - provides data ingestion services for data providers. Resides in a separate subnet from WFST-1.	T2.micro	1 vCPU, 1 GB RAM
NGINX-1	Primary NGINX service - acts as a reverse proxy server and provides load-balancing functionality	T2.micro	1 vCPU, 1 GB RAM
NGINX-2	Secondary NGINX service - acts as a reverse proxy server and provides load-balancing functionality. Resided in a separate subnet from NGINX-1.	T2.micro	1 vCPU, 1 GB RAM
NAT1	Allows internet access to the other EC2 instances (for added security reasons)	T2.micro	1 vCPU, 1 GB RAM
RDS			
CITISENSE-VERSION2-2	Hosts PostgreSQL/PostGIS database for data storage, maintenance and retrieval.	Db.m3.large	2 vCPU, 7.5 GB RAM

The image below shows the location and relationships of each of the instances in the overall SEDS Platform architecture.

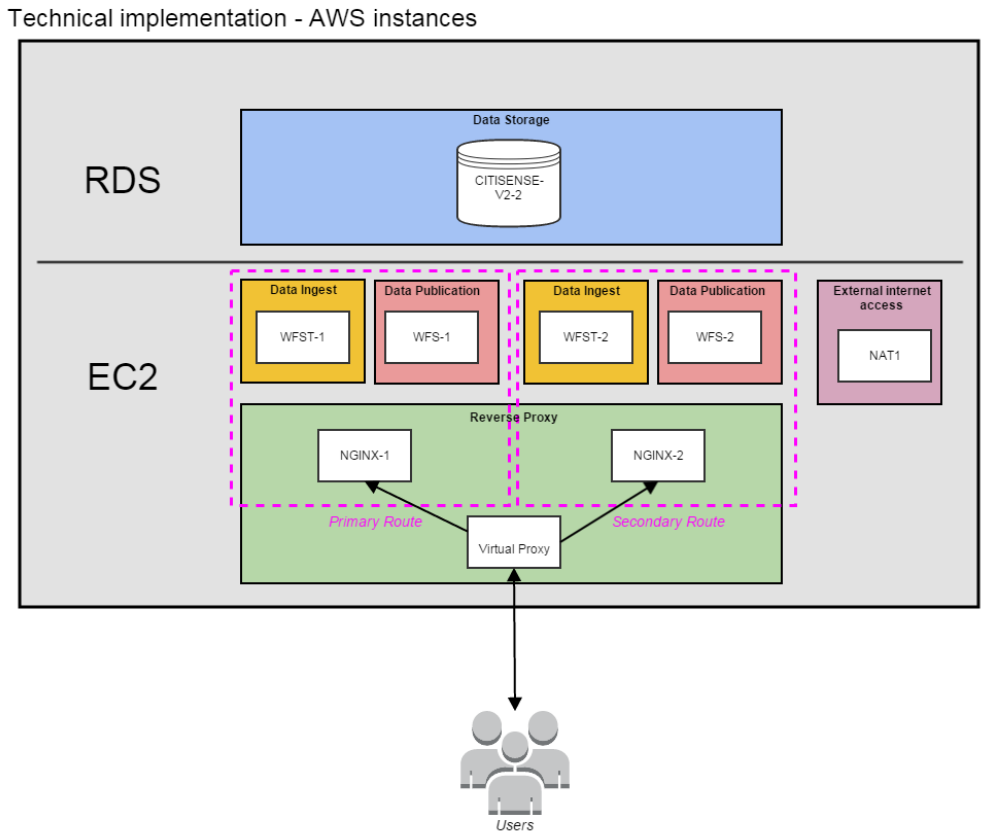


Figure 5-3 Instances in the overall SEDS Architecture

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 end-point. 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 Identifiers

In CITI-SENSE a large number of sensing devices are deployed, ranging from static stations, mobile devices and wearable sensors, delivered by a variety of different suppliers. Each supplier applies its unique and often optimised methodology for uniquely identifying individual devices within their sensor network. To avoid duplication of identifiers from the various sensor data providers, and therefore avoid the risk of misidentification, it is advisable to agree on a common specification to uniquely identify any sensing device.

A unique identifier can be defined as a numeric or alphanumeric string that is associated with a single entity within a given system. In CITI-SENSE, we define a single entity as a sensing device within the CITI-SENSE context, such as a static monitoring stations or a mobile device.

Benefits

There are a number of benefits to be gained by adopting a common understanding in the definition of unique identifiers for individual sensors:

1. Standardisation in the definition of the ID sensor within the CITISENSE project prohibits any misidentification or duplication of sensors;
2. Easy to understand (human-readable) by the sensor providers. The sensor data providers can easily identify their sensors.
3. Easy to understand (human-readable) by the end-users. The sensor identification provides end-users with a basic and quick indication ("metadata") of the provider and location of the individual sensor.

5.3.3.1Unique Identifiers in the EU INSPIRE Directive

The EU INSPIRE Directive[10] aims to establish an infrastructure for spatial information in Europe to support Community environmental policies and policies or activities which may have an impact on the environment. The INSPIRE Directive has been adopted as part of national legislation by all EU member states, therefore making it a legal obligation to adhere to.
The INSPIRE Directive does specifically apply to datasets that have been produced or received by a Public Authority, or are managed or updated by a Public Authority where it falls within the scope of its public task.

Under the INSPIRE Directive 34 thematic themes are defined. Datasets that fall within any of these 34 themes must be made available using the INSPIRE implementation rules. One of these themes cover Environmental Monitoring Facilities (EMF), which are defined as: "*Location and operation of environmental monitoring facilities includes observation and measurement of emissions, of the state of environmental media and of other ecosystem parameters (biodiversity, ecological conditions of vegetation, etc.) by or on behalf of public authorities.*" [11]

The definition of the Identifier in the INSPIRE initiative (InspireID) is an external unique object identifier published by the responsible body, which may be used by external applications to reference the spatial object.

[10] http://inspire.ec.europa.eu/
[11] Deliverable 7.2 - Core ontology network for city observatories applications. Point 4.1.3 Data Sensors - INSPIRE data specification on Environmental Monitoring Facilities

The InspireID consists of three elements, which together form a universally unique identifier:

Attribute	Definition	Type	Voidability
localID	A local identifier, assigned by the data provider. The local identifier is unique within the name space that is no other spatial object carries the same unique identifier.	CharacterString	
namespace	Namespace uniquely identifying the data source of the spatial object.	CharacterString	
versionID	The identifier of the particular version of the spatial object with a maximum length of 256 characters. If the specification of a spatial object type with an external object identifier includes life-cycle information, the version identifier is used to distinguish between the different versions of a spatial object. Within the set of all versions of a spatial object, the version identifier is unique.	CharacterString	voidable

5.3.3.2 Proposed sensor identification methodology in CITI-SENSE

Following the methodology from the INSPIRE Directive, the structure of the CITI-SENSE sensor unique identifier is constructed by combining three components which substitute the localID, namespace and versionId. The components are separated with a hyphen (-). All components are in uppercase.

The following pattern is a representation of the proposal for the CITISENSE sensor unique identifier:

{PREFIX} - {DOMAIN} - {LOCALID}

Prefix

The prefix identification refers to the name of the CITI-SENSE project. This parameter is fixed.

{CITISENSE}-{DOMAIN}-{LOCALID}

Prefix

Domain

This parameter is variable. In our case, the domain refers to the name of the supplier of the sensor device (i.e. the sensor provider).

{PREFIX}-{DOMAIN}-{LOCALID}

Name of the
sensor provider

LocalID

This parameter is also variable. It is a local identifier assigned by each sensor provider.

{PREFIX}-{DOMAIN}-{LOCALID}

A local identifier assigned by the Sensor Provider

Examples:

CITISENSE-GEOTECH-0001
CITISENSE-JSI-0001
CITISENSE-AIRBASE-2

5.3.4 Sensor Data Ingest

Once a sensor has been successfully registered in the CITI-SENSE SEDS Platform, it can start ingesting sensor data. 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 their individual ecosystems. The challenge is to connect these disparate federated sensor networks to the single, centralised Data Repository.

One of the cornerstones of the CITI-SENSE SEDS Platform is the fact that it adopts 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 compliant, Web Feature Standard (WFS) is used. A **WFS-T**, transactional web service endpoint is available and allows sensor data providers to post their datasets against. This can be done periodically or in real time.

Two data ingestion web service patterns are identified:

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 Platform fetches ("*pulls*") data from the sensor data provider network and stores it in the Data Repository. This can be done periodically or in real-time.

Which pattern to use by the sensor providers depends on the technological capabilities that the sensor data provider network can provide.

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 play 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, a number specific data models have been developed. Data translation processes ensure that data is exchanged between these models. One of the benefits of this de-coupling of individual processes is that each

process can be highly optimised to meet its purpose. Also, as each process may require a particular expertise and skill-set, multiple teams can work on individual processes.

1. Data translation can be represented by two key processes:
2. **Data format conversion** - the encoding of the dataset is translated (e.g., XML to SQL, SQL to KML)
3. **Data model translation** - converting between conceptual models

Different end-users of the data will require different data models. A rich data model containing many feature types and detailed attribution would be suitable for someone wanting to carry out complex analysis. 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 general use would require the data to be in a more simplified model to suit the users' 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 adopted.

SenML is a W3C standard data model for encoding measurements and simple metadata about measurements and devices (e.g., 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 location of the device.

5.3.5 Data Storage and Processing

The Data Storage component in the SEDS Platform 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 a free and open source software application which adds support for geographic objects to the PostgreSQL database. PostGIS follows the Simple Features for SQL specification from the Open Geospatial Consortium[12].

[12] http://www.opengeospatial.org/standards/sfs

The solution adopted to create the PostgreSQL/PostGIS Relational Database has been the one provided by the Amazon Web Service. The following image shows how is connected to the different WFS services created for the communication with the actors involved in the CITISENSE project:

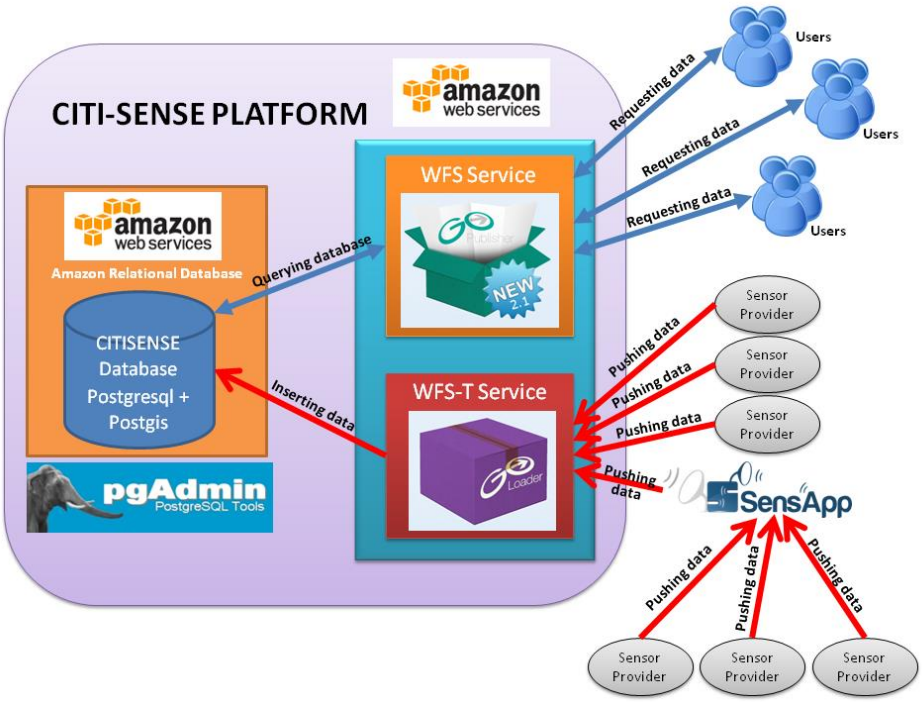


Figure 5-4 PostgreSQL/Postgis RDS connected to the WFS and WFS-T

5.3.6 Data Publication

The CITI-SENSE SEDS Platform enables access to data stored in the SEDS Data Repository via open 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, CSV and JSON.

For creating and deploying the WFS web services the software product GO Publisher for Snowflake Software is used and for creating and deploying the REST services Mulesoft Anypoint has been used.

WFS

The GO Publisher product allows for rapid configuration and deployment of WFS web services via a graphical user interface. 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.

REST

Mulesoft Anypoint Studio is used to create and deploy the REST services. Its graphical user interface allows for rapid configuration.

6 CITI-SENSE Data Model

6.1 Usage and Requirements

At the heart of the CITI-SENSE SEDS Platform sits the Data Repository component. In its basic function, the Data Repository forms the link between data providers and data consumers and stores the data from the data providers in a single, centralised environment.

More specifically the data model must be able to perform a number of functionalities, such as:

- Store millions of sensor data measurements;
- Store data for measurements over a long time period (years).
- Handle large amounts of small datasets being inserted in short intervals (eg. every 5 mins);
- Allow the data to be queried and retrieved in within reasonable response times.

In the CITI-SENSE project data can be stored using two distinct technologies: (i) a relational SQL database and (ii) a triplestore for the storage and retrieval of Linked Data. This chapter describes the data model developed for the relational SQL database.

6.2 Information model

6.2.1 Data Model Design

For designing the data model the Model Driven Architecture approach (MDA) has been followed. MDA defines system functionality using a platform-independent model using an appropriate domain-specific language. MDA is a recommend best practice and many well-known standards and models have been developed using the MDA approach.

For CITI-SENSE a high-level model was developed first using the Unified Modelling Language (UML). UML is a general purpose modelling language and is an ISO standard. The software tool Enterprise Architect by Sparx Systems was used to create the UML model.
Following agreement on the UML model, it was then exported to an XML Schema document (XSD) using the open source software tool ShapeChange[13] from interactive instruments GmbH.

6.3 Implementation

In Deliverable D7.2 the Semantic Sensor Network (SNN) ontology is proposed to be a foundation for the data model. The SSN ontology was initiated by the World Wide Consortium (W3C) in 2009 and can be used to model sensor device, systems, processes and observations. Furthermore SSN enables expressive representation of sensor, sensor observations and knowledge of the environment.

For the first initial draft design of the data model a more pragmatic and simpler approach is taken. The draft design has been developed in the spirit of the SSN ontology, but only adopts a small number of basic elements for now.

For creating the UML model a 3-step approach has been followed:

[13] http://shapechange.net/

1. A simple diagram was initially created that described the various actors and defined the relationships between them.
2. From this simple diagram a more comprehensive use case diagram was developed in Enterprise Architect using UML.
3. In the final third step a class diagram was developed, in Enterprise Architect, which identifies the various classes and their relationships. This class diagram forms the basis for the subsequent XML Schema.

6.3.1 Classes

In the CITI-SENSE UML model the following classes have been defined –with the red circled classes for the Common Air Quality Index (CAQI) being new additions to this version 2.2 of the model:

Class name	Description
Cities	Pilot cities participating in CITI-SENSE (e.g., Barcelona)
SensorProvider	Details about the sensor data providers (e.g., AirBase)
Sensor	Information about the sensor that is used to make the observation (e.g., Nexus4).
Observation	Information about the observation, such as time, location and type.
Measurement	The result(s) of an observation. E.g., temperature, pollution level and location.
Questionnaire	A specific type of observation used by the CivicFlow and CityAir applications which capture subjective data.
Question	Each Questionnaire contains one or many questions.
Response	The Response to a Question
Answer	The Answer to a Question
CAQI Value	Common Air Quality Index; provides a general overview of the air quality
Global CAQI	Global Common Air Quality Index

The full UML class diagram is pictured below:

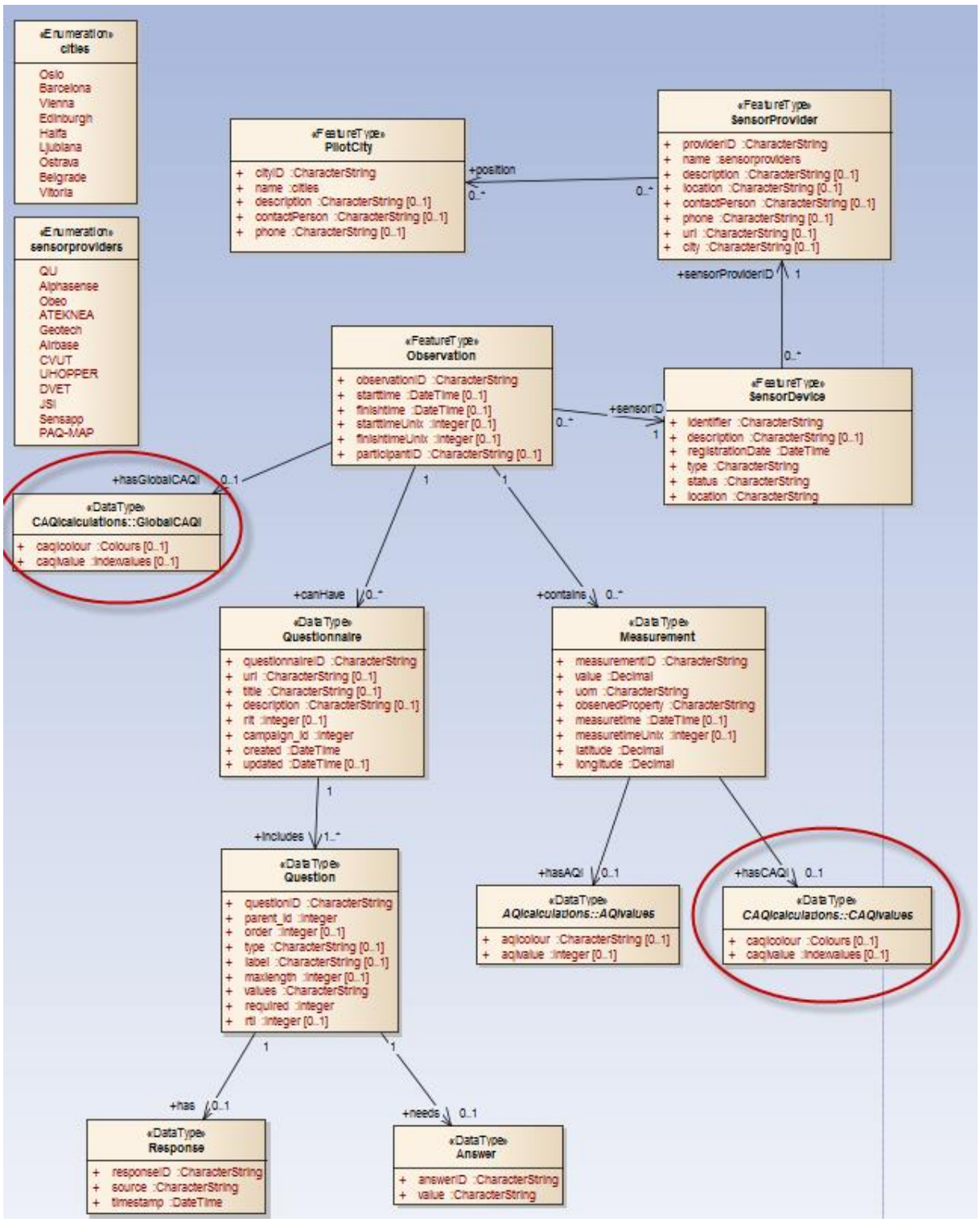


Figure 6-1 Data model UML class diagram

Adopting Open Standards

A key requirement of the CITI-SENSE SEDS Platform architecture is that sensor data will be exchanged using Open Standards. As geography plays a vital role in the overall Platform architecture, the ISO19136/OGC GML standard is adopted to encode sensor data. The adoption of the GML standard is

mirrored in the UML model by embedding specific GML elements in the class diagram. This is most obvious in the choice of using the <<FeatureType>> tag that defines the class' stereotype. Also, for defining geographical elements, the attribute types are defined using GML elements such as "GM_Point".

Export UML model to XML Schema
This CITI-SENSE UML model is transformed to an XML Schema document using the free application ShapeChange. The ShapeChange tool enables the conversion between an Enterprise Architect export file (.xmi) to an XML application schema (.xsd) by using a user-defined configuration. In this transformation, the UML class are converted into GML Feature Classes and UML attributes become GML properties.

From XML Schema to SQL database schema
The off-the-shelf software product GO Loader from Snowflake Software has been used to create a database model based on the XML Application Schema created from the UML model. GO Loader parses any given XML application schema and based on the user's configuration settings, generates the SQL code which creates the database schema, including tables, joins and indexes. This allowed for a rapid deployment of a simple database schema that can be used for prototyping activities.

6.3.2 Further developments

It is important to note, that the data model will be developed further during the next phases of the CITI-SENSE project and continue evolving. By adopting the MDA approach and using the UML standard, it allows the model to grow and adept without the need of creating a new model from scratch.

On 1 September 2015 an improved SEDS Platform was released to consortium partners. It implements version 2.2 of the data model, which incorporates many recommendations from the previous trials. In the next version of the data model further comments and experiences from the preliminary tests will be absorbed. Some of the proposed improvements include:

1. Distinguish clearly between the name of the pollutant and the units of measure.
2. Standardisation and harmonisation in the naming in the definition of tables and attributes.
3. Further integration of the OGC Observations& Measurements standard.
4. As mentioned earlier, the W3C Semantic Sensor Network ontology has been proposed as the foundation for the development of the CITI-SENSE Platform and its components. In the next iteration of the data model design, the SSN ontology will be further adopted and integrated.

7 Sensor services

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

7.1 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 are 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 without any pre-existing sensor platform, in particular for mobile smart phone connected sensors, the CITI-SENSE project is offering the usage of the SensApp platform.

7.1.1 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.

7.2 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 SINTEF[14].

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. As depicted in Figure 7-1, the SensApp framework includes a mobile client service with a particular emphasis on support for Android Smartphones with Bluetooth connected sensors. The SensApp client side is composed of two applications: SensApp Android and SensorLog.

[14] http://sensapp.org/

Figure 7-1. Overall architecture

SensorLog 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.

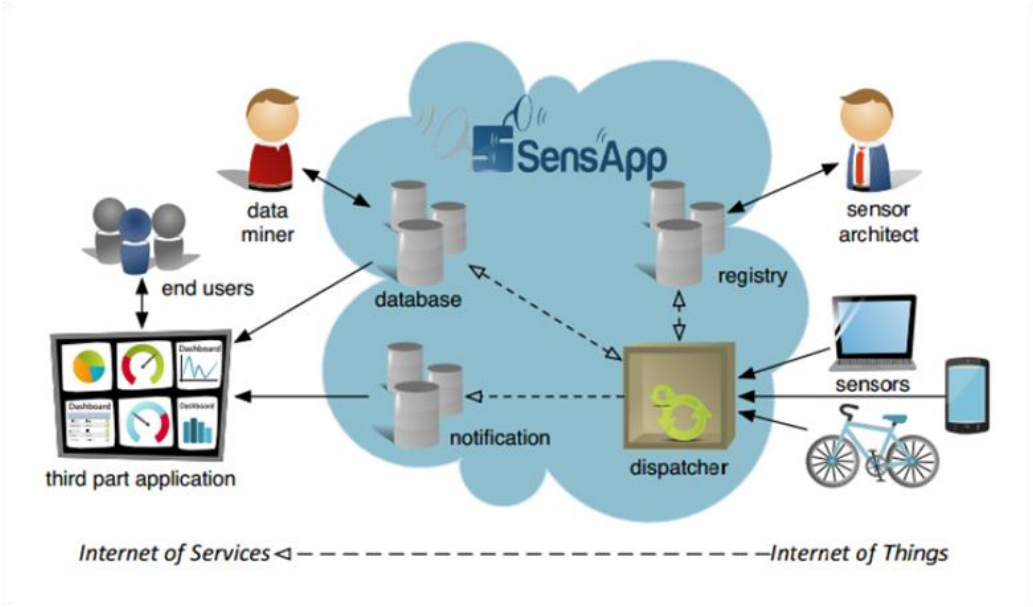


Figure 7-2 SensApp platform architecture

The Figure 7-2 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.
Indeed, the services are implemented as REST services exposed on top of the HTTP protocol which makes them easy to deploy on a cloud infrastructure. Sensor data are represented with the SenML standard, used as an intermediary representation of sensor data in the platform. SensApp also provides a web-based administration interface built on top of the offered REST interface. This interface supports the manipulation of sensors, and provides graphical widgets to visualize data collected in SensApp.

The data exchanged between the various SensApp services themselves and between these services and the devices are represented in SenML standard using JSON syntax. The choice for this syntax has been made to minimise the amount (size in term of byte) of data to be exchanged between the services and the devices.

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 7-3.

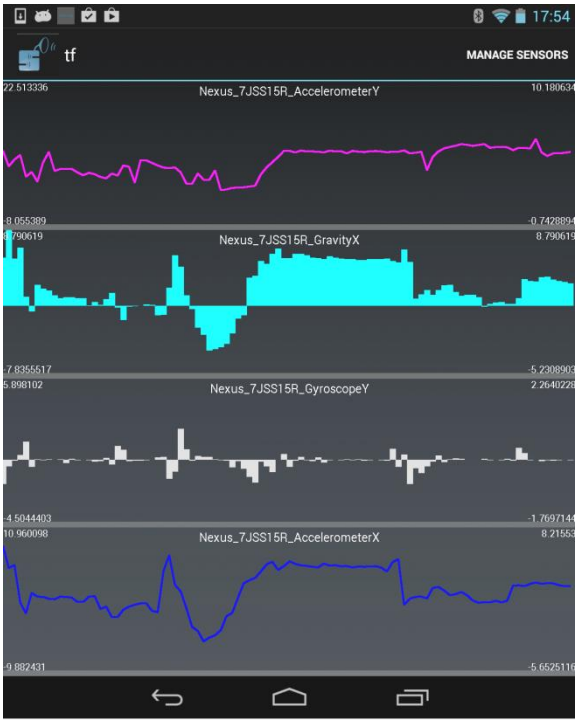


Figure 7-3 Sensor local app interface

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 that communicates via Bluetooth. The set of Bluetooth sensors can easily be extended by exploiting the template provided in the application code repository[15].

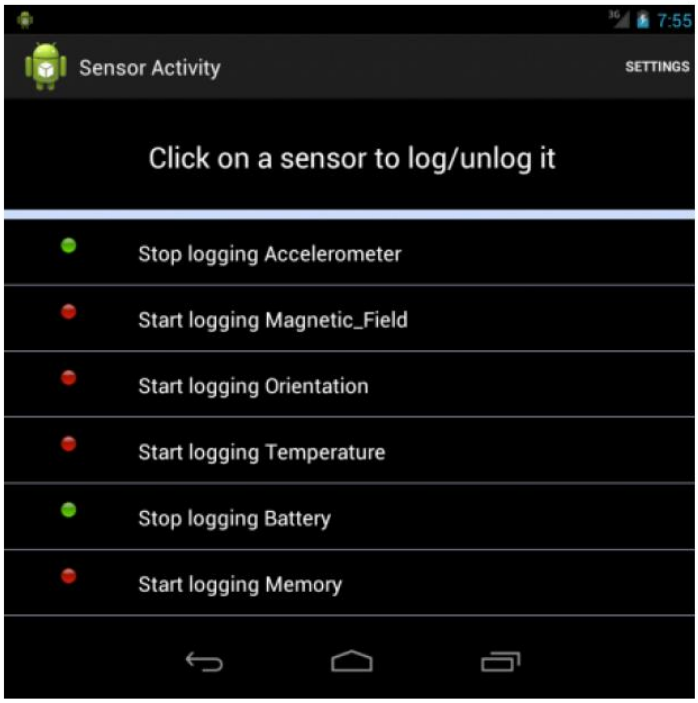


Figure 7-4 Sensor log interface

The user can select several sensors to be logged, as shown in Figure 7-4, 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 7-5 Bluetooth Smart and Bluetooth Smart Ready

[15] https://github.com/SINTEF-9012/sensapp-android-client

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 7-5 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 case study.

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 send, 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 7-6 BLE Device – UV Sensor

The BLE device, which is used as the example and, is a SunBit UV sensor, shown in Figure 7-66. 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. Therefor 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 7-7 BL Device – Weather Meter

As the test device for classical Bluetooth, a Kestrel 4000 Weather Meter (from the Vitoria case study) is used, as shown in Figure 7-7.

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

7.2.1 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) is transmitted as XML.

7.3 Implementation technologies

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

Technically, the SensApp server is written in Scala and uses Spray[16]. In order to run properly, SensApp requires a servlet container (Jetty) and a NoSQL database (MongoDB[17]). The SensorLog and SensApp Android applications are java-based Android applications.

In particular, during the project, 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.

7.3.1 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:

[16] http://spray.io/
[17] http://www.mongodb.org/

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 7-1 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 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.

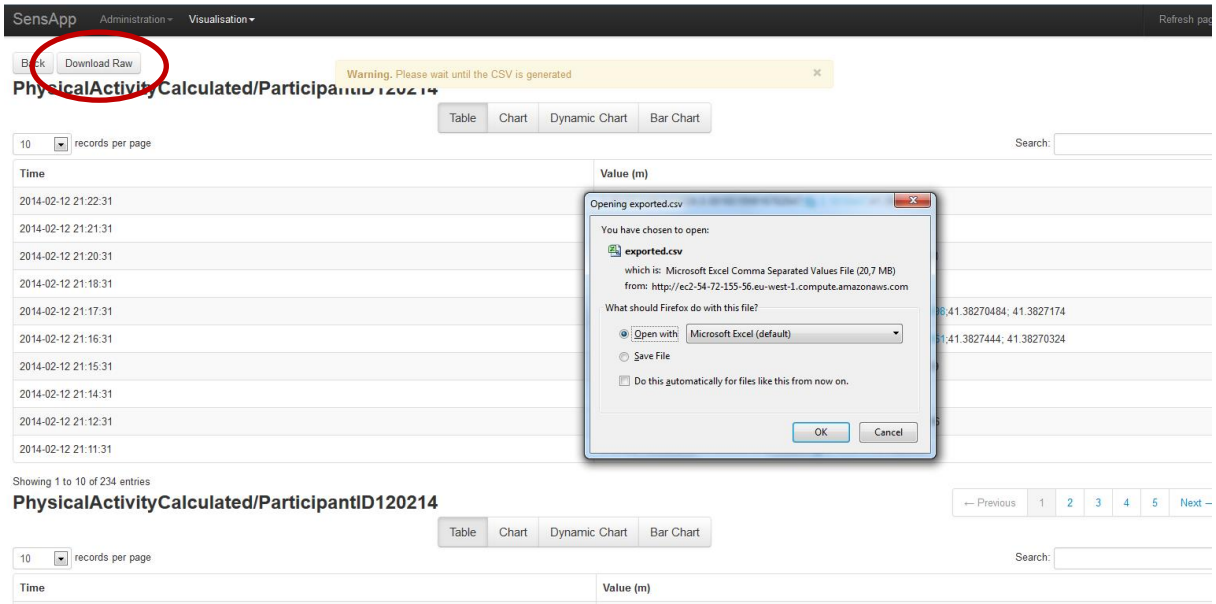


Figure 77 CSV export

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))
```

7.3.2 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 are 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 52North[18].

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.

[18] http://52north.org/

- **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 PostGIS[19] 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.

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**

[19] http://www.postgis.org/

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_lnk": <Dataset URL>
}
```

The observation data are given as:

```
{
  "bn": "<Sensor ID>",
  "bt": <Base time in seconds after EPOC 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 there are not enough information from SenML to fill all the fields of SensorML (SOS/52North) (However, we can note that the provided information can be obtained within the SenML definitions by using the concept of `"tags"`.). When faced with these gaps, there are three alternatives: 1) provide additional information (from outside of the SensApp definitions), 2) generate the data, or 3) leave fields blank.

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 it taken for the handling of missing information. This can enable support of interoperability between an approach using SenML and an approach using SensorML.

8 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 the appropriate visual representation of sensory data coming from heterogeneous sources. Hereafter by sensors we understand both physical sensors, such as temperature, ozone, nitrogen dioxide and other sensors, as well as perception sensors, which are implemented in form of questionnaires inquiring the subjective perception of environmental conditions experienced by users. 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 smartphone applications. 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 relevant features of collected (raw) and processed data, such as temporal/ spatial behaviour, and other available dimensions.

8.1 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 are 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 maximise the impact of the project activities and to reach out effectively to 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 based on the state of the art technologies for the target platforms.

In order to provide appropriate and rich visualizations, the visualization widgets will support various visual rendering forms, including maps, time series, different types of charts (such as gauge, line, pie, column, bar, area, etc.) as detailed in the next sections.

8.2 Logical Service Interfaces and Information Model

Visualization widgets are provided to interact with the data/product CITI-SENSE module in order to retrieve the data to be visually rendered. In order to reduce the computational overhead required for rendering visualization on the end-user side, be it web portal or mobile application, data has to be appropriately processed upfront. The data preparation consists in the ability to extract from the CITI-SENSE Storage all measurements of a given sensor, the latest measurement, the measurements for a specified temporal period, etc. As described in the previous chapters, the sensor data are collected, stored and retrieved from the CITI-SENSE Spatial and Environmental Services Platform by using Data Storage and Processing Services. Storage Services provide functionality to retrieve sensor values and results of perceptions acquisitions in form of observations, encoded in XML or JSON format. Hence, given a sensor identifier, a visualization component should be able to call Data Storage and Processing Services and construct the appropriate visual representation of the retrieved data.

Logically, the visualization widgets can be divided into two groups: (i) visualization of real-time sensor measurement, and (ii) visualization of historical sensor measurements, possibly restricted to a specified temporal interval. Besides the depicted measurement, both types of visualizations shell provide the possibility to visually depict the environmental impact of the value, which in particularly can be achieved through the use of different colours for showing for example that a measurement is in norm, or can potentially lead to environmental risks, etc. In the next section, we describe implementation details of the provided CITI-SENSE widgets.

8.3 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.
Two solutions were developed by consortium partners and made available for integration purposes.
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 further description of the usages of relevant user interface components is provided in D7.5 part 2 Operations.

8.3.1 Jquery plugin

Given the requirement of configurability and ease of deployment, one option consists in implementing widgets as jquery plugins (http://jquery.org), due to the fact that most of modern web portals with dynamic and interactive content are already based on jquery JavaScript extension. This does not require any modification server-side.

8.3.2 PHP framework

An alternative Framework for Widget Visualization, based on PHP, has also been developed. The following Figure describes the architecture of the framework, which requires the use of the DNET PHP based widget wrapper within the CITI-SENSE WFS server platform.

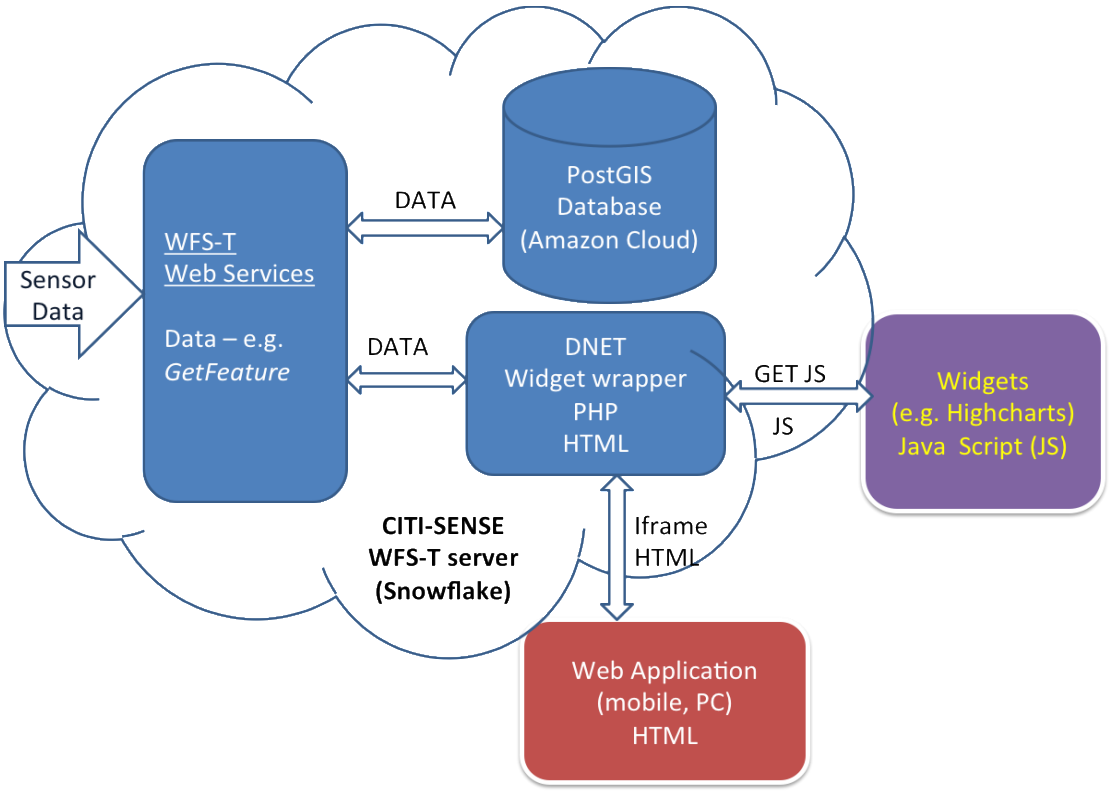


Figure 8-1: Php-based widget visualization framework

In order to embed the real-time sensor reading visualization into a web page iFrames can be used. As an example, the page http://citisense.dunavnet.eu/share.php?dev=1&sensor=humidity uses the following html code.

<iframe src="http://citisense.dunavnet.eu/share.php?dev=1&sensor=humidity" width="240" height="320" scrolling="no" frameborder="0"></iframe>

8.4 Basic Visualization Widgets

Visualization of sensors readings is achieved through the use of sensor statistics widget. The widget makes use the Highcharts library (http://www.highcharts.com). Generally speaking, we consider two types of widgets, one for visualizing real-time information (i.e., one single value) and one visualizing historical information (i.e., a time series). In order to visualize the real-time value of a given sensor, a Gauge chart has been selected. To visualize the historical timeline of sensor measurements, a line chart has been selected. In the following we consider, for each of them, how they can be generated using the two approaches outlined in the previous section.

8.4.1 Real-time sensor reading visualization widget

8.4.1.1 Through Jquery plugin

In order to embed the real-time sensor reading visualization into a 3rd party web page or in a mobile application, a developer needs to include the sensor statistics widget library and call the visualization rendering JavaScript code.

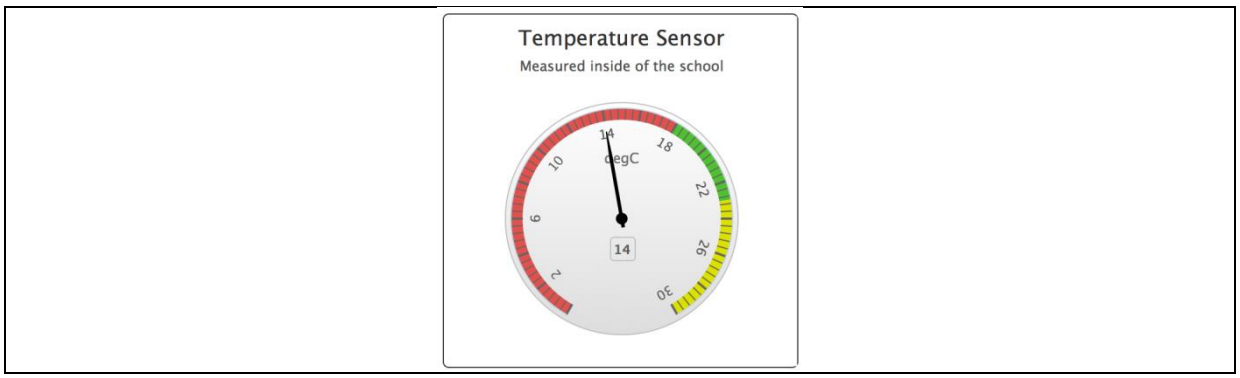
Given a sensor identified by its identifier, the visualization of the real-time value is achieved through the following lines of code:

```
<script src="http://citisense.u-hopper.com/widgets/sensors/jquery.sensor.statistics.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
 $(function() {
  $("#YOUR_CONTAINER_ID").sensorstatistics({
   type: 'realtime',
   id: "CITI-SENSE-CS-000AF190240240413",
   title: "Temperature sensor",
   subtitle: "Measured inside of the school",
   min: 0,
   max: 30,
   bands: [{
      from: 0,
      to: 18,
      color: '#DF5353' // red
   },{
      from: 18,
      to: 23,
      color: '#55BF3B' // green
   },{
      from: 18,
      to: 23,
      color: '#DDDF0D' // yellow
   }
   ],
   refresh: 1000,
  });
 });
</script>
```

Where the widget configuration parameters are:

- type – mandatory visualization type, needs to be set to realtime;
- id – mandatory sensor identifier;
- title – optional widget title;
- subtitle – optional widget subtitle;
- min/max – optional minimum and maximum extremes;
- bands – optional colored bands defined by intervals of values, in order to give the possibility to visually represent the environmental impact of the present sensor value;
- refresh – optional periodic refresh rate of the visualization given in milliseconds, when set re-retrieves the sensor data and reconstructs the chart.

When the widget code is executed the following Gauge-type visualization is constructed:



8.4.1.2 Through PHP-based framework

The following subsections give details on how to use PHP-based Widget Framework to generate gauge-type visualizations of measurements, including installation, configuration and getting data from the remote services (i.e. CITI-SENSE data server) (see also http://srv.dunavnet.eu/citisense/). First, download and unzip citisenseGF.zip to your project folder. Create a blank index.php file and simple 'include' Citisense.php in /libs/ directory

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

There are two ways to use and process data. The first one is to configure Database.php class. Input required information includes database driver, database hostname, user account, password etc. Several database types are supported (e.g. mysql, mssql, postgresql and mongoDB). The second, and more convenient way, is to use already hosted web services, and parse data. Several services can be used to parse data from xml and json files. For instance, for parsing a real-time data rdf/xml page can be used, see

```
http://srv.dunavnet.eu/rephandler/ecobus/355255040755580
```

where 355255040755580 is the device ID.
When citisense.php library is included, simply make two instances of Citisense and Gui classes. Citisense class contains all required methods to handle widgets. The Gui class is responsible for drawing HTML block and widgets.

```
<?php
$citi = new Citisense();
$gui = new Gui();
```

When instances are created it is necessary to assign data to variables It is possible to parse any kind of data from a service, xml or json file. The first step is to configure Config.php file (/libs/Config.php). Enter your basic database info (such as, hostname, dbname, port, username, password) and continue. Parsing data directly from database example:

```
<?php
$pressure = $citi->getLastMeasurement('measurements', 'pritisak', 'measurement_id');
```

Parsing data from a real time web service (rdf/XML):

```
<?php
$prWS = $citi->parseLiveXML('http://89.216.116.166/rephandler/ecobus/355255040755580', 'pressure');
```

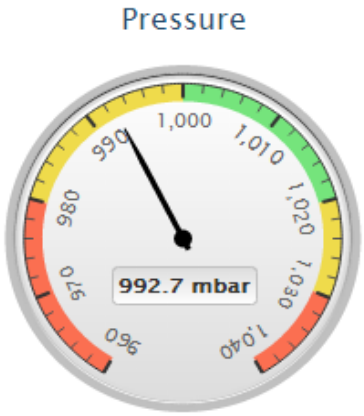
where 'measurements' are stored in the database table named, 'pritisak', is row name that we want to parse and 'measurement_id' is the table id row.
System also supports fetching data from Snowflake web services (in JSON format)

```
<?php

include_once ("libs/Citisense.php");

$citi = new Citisense();
$gui  = new Gui();

// parsing data from web service
// ******************************
$co2 = $citi->parseRemoteJSON('http://ec2-54-171-68-180.eu-west-1.compute.amazonaws.com:8080/citisense/webproxy/lastrecord/sens
$co = $citi->parseRemoteJSON('http://ec2-54-171-68-180.eu-west-1.compute.amazonaws.com:8080/citisense/webproxy/lastrecord/sensc

// // draw html header structure
$gui->drawHeader();

// // draw widgets parsed from web service
$gui->drawCO2Widget($co2);
$gui->drawCOWidget($co);

// // draw html footer structure
$gui->drawFooter();
```

parseRemoteJSON() function has two parameters. The first parameter is the actual link to the web service data on Snowflake's web server, and the second is the sensor whose measurements are to be displayed. The system supports different types of sensors (e.g. CO, CO2, SO, SO2, O3, NO, humidity, temperature). Please note: the second parameter in the function must be named as already specified in service.
After we store data into variable, the next step is to draw required widget. Firstly it is necessary to setup basic html structure with drawHeader() and drawFooter() methods. It is very important that all other calls to functions which draw widgets goes between those two methods. In this example (i.e. demo library), the system can only draw Pressure, Humidity, Temperature and WindSpeed.

```
<?php
$gui->drawHeader();
$gui->drawPressureWidget($pressure);
$gui->drawFooter();
```

Result should be



Positioning and sizing the widget can be done directly in html and css. Simply in CSS file, reference #container id, (*this is the location at which highcharts inject their javascript*) and add a style width and height.

```
$(document).ready(function(){
    $("#container").css("width","400px");
    $("#container1").css("width","200px").css("height","600px");
    $('#container2').css("margin-left","20%");
});
```

8.4.2 Historical sensor readings visualization widget

8.4.2.1 Through Jquery plugin

In order to embed the historical timeline of sensor readings into a 3rd party web page or in a mobile application, a developer needs to include the sensor statistics widget library and call the visualization rendering javascript code.

Given a sensor identified by its identifier, the visualization of the historic timeline of values is achieved through the following lines of code:

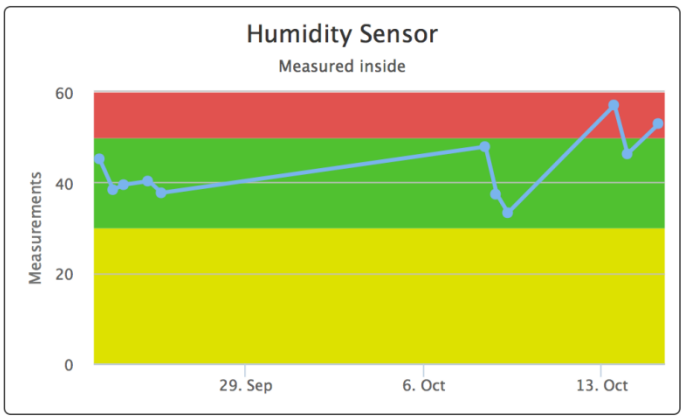
```
<script src="http://citisense.u-hopper.com/widgets/sensors/jquery.sensor.statistics.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
 $(function() {
  $("#YOUR_CONTAINER_ID").sensorstatistics({
   type: historic,
   id: "CITI-SENSE-CS-000HM190240260510",
   title: "Humidity sensor",
   subtitle: "Measured inside of the school",
   from: 1401260350,
   to: 1401260473
   min: 0,
   max: 100,
   bands: [{
      from: 0,
      to: 30,
      color: '#DDDF0D' // yellow
   },{
      from: 30,
      to: 50,
```

```
        color: '#55BF3B' // green
    },{
        from: 50,
        to: 100,
        color: '#DF5353' // red
    }
    ],
    refresh: 1000,
  });
 });
</script>
```

Where the widget configuration parameters are:

- type – mandatory visualization type, needs to be set to historic;
- id – mandatory sensor identifier;
- title – optional widget title;
- subtitle – optional widget subtitle;
- from/to – optional timestamps for limiting the historic period to be visualized
- min/max – optional minimum and maximum extremes;
- bands – optional colored bands defined by intervals of values, in order to give the possibility to visually represent the environmental impact of the present sensor value;
- refresh – optional periodic refresh rate of the visualization given in milliseconds, when set re-retrieves the sensor data and reconstructs the chart.

When the widget code is executed the following line-type visualization is constructed:



8.4.2.2 Through PHP-based framework

The procedure follows the steps outlined for the real-time visualization. The difference is in the request of data to be fetched.

On DNET server, encoded query for parsing historical data were create directly into service. For example in:

```
http://srv.dunavnet.eu/rephandler/ecodb/
```

query can be done through (in SQL):

```
SELECT co_ppm FROM history_data WHERE timestamp_db>= DATEADD(day, -1, GETDATE())and bus_identifier = 355255040755580
```

that after encoding looks like:

```
SELECT%20co_ppm%20FROM%20history_data%20WHERE%20timestamp_db%3E%3D%20DATEADD(day%2C%20-1%2C%20GETDATE())and%20bus_identifier%20%3D%20355255040755580
```

and generated xml is given (follow the link):

```
http://srv.dunavnet.eu/rephandler/ecodb/SELECT%20co_ppm%20FROM%20history_data%20WHERE%20timestamp_db%3E%3D%20DATEADD(day%2C%20-1%2C%20GETDATE())and%20bus_identifier%20%3D%20355255040755580
```

and consequently all yesterday's CO data for 355255040755580 device are retrieved.

In the Php framework historical data can be plotted using the function drawHistoricalPlot() which takes two or more input parameters, first the array of upper limits of colour range (e.g. green, orange and red), and after that arrays of data to be plotted (e.g. one or more).
drawHistoricalPlot([3,5,8], [1,2,3,4,5,6,7,8,9,10,11,12], [5,4,25,37,13,28,71], ...);
For example, if the range is defined as follows: $range = [3,5,8]
whereby in the range from 0 to 8, green takes values to 3 and orange to 5.

The PHP framework in its current form does not support integration with WFS for drawing historical data. Server-side modifications on the WFS platforms are required for retrieving historical data series.

8.5 Application to CITI-SENSE sensors

In this section we present how the basic visualization widgets presented in the previous sections can be used and combined to properly visualize data from the CITI-SENSE sensors. All applications described in this section are based on the jQuery plugin approach.

8.5.1 CITI-SENSE Kestrel sensor: historic data

Given a sensor identified by its identifier, the visualization of the historic timeline of values is achieved through the following lines of code:

```
<script>
    $(function () {
        $("#YOUR_CONTAINER_ID").linegraphkestrel({
            url: "http://ec2-54-72-154-32.eu-west-
1.compute.amazonaws.com/sensapp/databases/raw/data/",
            kestrelid: "Nexus_5KOT49H",
            datatype: "WS",
            from: 1402476985,
            to: 1402477253,
            refresh: 0
        });
    });
</script>
```

Where the widget configuration parameters are:

- url – mandatory url where data are stored
- kestrelid – mandatory sensor identifier
- datatype – mandatory data type, the options are RH (Relative Humidity- default), T (Temperature) or WS (Wind Speed)
- from – mandatory timestamp
- to – optional timestamp
- refresh – optional periodic refresh rate of the visualization given in milliseconds, when set re-retrieves the newest sensor data and adds it the chart.

This widget configuration can be used to produce chart as a static chart or a chart that updates and retrieves new data.

Example of creating an historical graph that updates and looks for new data:

```
    <script>
    $(function () {
        $("#demo5").linegraphkestrel({
            url: "http://ec2-54-72-154-32.eu-west-
1.compute.amazonaws.com/sensapp/databases/raw/data/",
            kestrelid: "Nexus_5KOT49H",
            datatype: "WS",
            from: 1402476985,
            refresh: 4000
        });
    });
    </script>
```

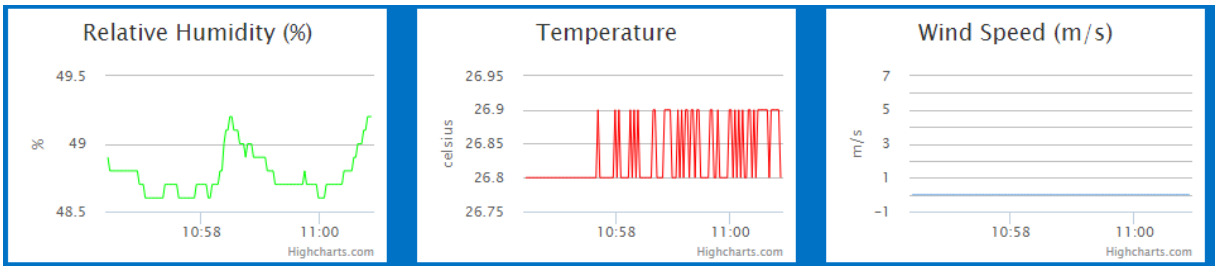
The library also supports a functionality for stopping the automatic update of the graph.

```
    <script>
        $(function () {
```

```
            $("#stopWSRefresh").linegraphkestrelstoprefresh({
                RevokeWS: true
            });
        });
    </script>
```

Where the widget configuration parameters are:

- RevokeT – optional, stops the temperature graph from updating
- RevokeRH – optional, stops the relative humidity graph from updating
- RevokeWS - optional, stops the wind speed graph from updating



8.5.2 CITI-SENSE Physical Activity Visualization Widgets

The computation of levels of physical activity is performed on the basis of accelerations measured through the citizens' mobile phone.

In order to embed the physical activity visualization into a 3rd party web page or in a mobile application, a developer needs to include the physical activity widget library and call the visualization rendering javascript code. Two types of visualization can be constructed, showing the level of physical activity over time (graph) as well as displaying on a map the positions where recordings were taken (map).

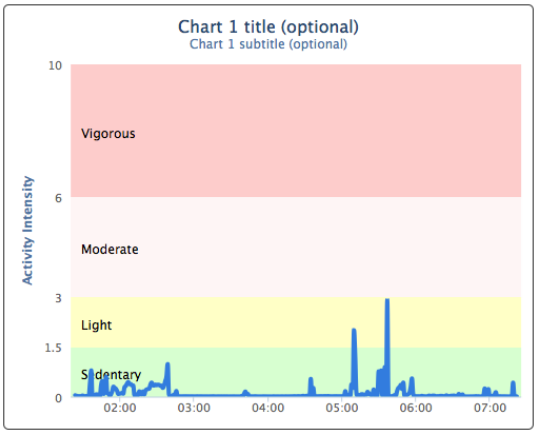
Given a participant's identifier, the visualization of the physical activity levels as a graph is achieved through the following lines of code:

```
<script src="http://citisense.u-hopper.com/widgets/sensors/jquery.sensor.statistics.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
  $(function() {
    $("#YOUR_CONTAINER_ID").physicalactivity ({
      type: 'graph', // or 'map'
      id: "CITI-SENSE-PARTICIPANT-001234562",
      title: "Physical activity ",
      subtitle: "Participant Thomas ",
});
  });
</script>
```

Where the widget configuration parameters are:

- type – mandatory visualization type, needs to be set to graph or the map;
- id – mandatory sensor identifier;
- title – optional widget title;
- subtitle – optional widget subtitle;

Once the widget is rendered it constructs the following visualizations:



8.5.3 CITI-SENSE Perceptions Visualization Widgets

Besides the environmental measurements performed with physical sensors, such as temperature, humidity, pressure, levels of carbon monoxide, nitrogen dioxide, etc., the CITI-SENSE project measures, through the use of questionnaires, the subjective perception of environmental conditions as experienced by citizens. Such questionnaires provide insight on how users perceive the quality of the air, pollution level and how this impacts their physical conditions. For creating and delivering questionnaires to citizens, CITI-SENSE leverages the CivicFlow tool (http://www.civicflow.com), developed by a Consortium partner (U-Hopper). The CivicFlow usage details can be found in pilots' description deliverable D3.2 and the technical details can be found in the Annex dedicated to the CITI-SENSE CivicFlow Questionnaire Service Platform.
Two types of visualization are foreseen. The first one shows on a timeline the level of engagement by users. The second one provides handy methods for visualizing the perceived quality of air.

8.5.3.1 Participation timeline visualization widget

The widget displays on a timeline the number of responses to a given questionnaire. In order to embed the participation timeline into a 3rd party web page or in a mobile application, a developer needs to include the questionnaire statistics widget library and call the visualization rendering javascript code.
Given a questionnaire's identifier, the visualization of the participation timeline is achieved through the following lines of code:

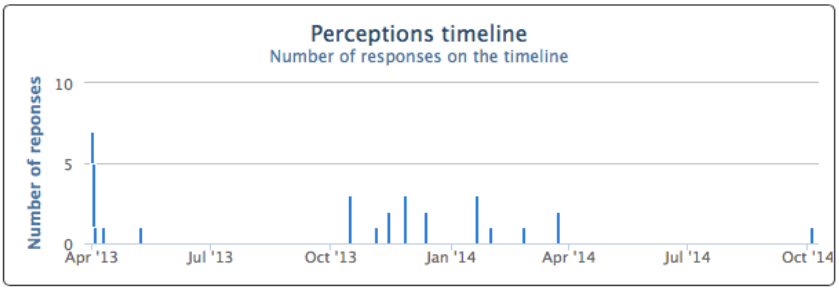
```
<script src="http://citisense.u-hopper.com/widgets/perceptions/jquery.questionnaire.statistics.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
 $(function() {
  $("#YOUR_CONTAINER_ID").questionnairestatistics({
   type: 'timeline',
   id: "QUESTIONNAIRE_ID",
   title: "Perceptions timeline",
   subtitle: "Number of responses on the timeline",
   refresh: 1000,
  });
```

```
  });
</script>
```

Where the widget configuration parameters are:

- type – mandatory visualization type, needs to be set to 'timeline';
- id – mandatory questionnaire identifier;
- title – optional visualization title;
- subtitle – optional visualization subtitle;
- refresh – optional periodic refresh rate of the visualization given in milliseconds, when set re-retrieves the participation data and reconstructs the participation timeline.

Once the following code is displayed by a browser or mobile application the following visualization is constructed:



8.5.3.2 Perception statistics visualization widget

The visualization of the results of the perception questionnaire is given on a per-question basis. In order to embed the perception statistics visualization widget into a 3rd party web page or in a mobile application, a developer needs to include the questionnaire statistics widget library and call the visualization rendering javascript code.

Given a questionnaire's identifier and a question number, the visualization of the responses is achieved through the following code snippet:

```
<script src="http://citisense.u-hopper.com/widgets/perceptions/jquery.questionnaire.statistics.js"></script>
<div id="YOUR_CONTAINER_ID" style="width: 300px; height: 500px;"></div>
<script>
  $(function() {
    $("#YOUR_CONTAINER_ID").questionnairestatistics({
      type: 'question',
      id: "1",
     question: "1",
     refresh: 1000,
});
  });
</script>
```

Where the widget configuration parameters are:

- type – mandatory visualization type, needs to be set to question;
- id – mandatory questionnaire identifier;
- question – mandatory question order in a questionnaire, starting from 1;
- refresh – optional periodic refresh rate of the visualization given in milliseconds, when set re-retrieves the responses data and reconstructs the chart.

Depending on the type of a question, list of options or open text, the visualization widget correspondingly constructs the pie chart demonstrating the distribution of answers between different answer options or a list of open answers provided by the citizens, as displayed in the following diagrams:

9 Security Services and Management

9.1 Introduction

Computer security, or cyber-security, is a highly-specialised domain in itself and applies to computers, mobile devices, data and networks, including the Internet as a whole. Within the CITI-SENSE project many components are used that have an inherent vulnerability to security breaches and therefore require a specific strategy.

WP2 requirements state that access to identifiable information about a citizen will be available only to authorised users and the individual themselves upon request. No identifiable information will be held on a portable computing device.

9.2 Security landscape

Security is an important aspect of the various components that form the CITI-SENSE Platform and covers not only the protection of the physical infrastructure (servers) in the CITI-SENSE Platform, but also securing the web services that interact with the data, the data stored in the data repositories, as well as the data that is exchanged over these web services.

In a more holistic approach the security landscape[20] covers a number of domains, which cover, but are not limited, to:

- **IT Service Security**, which covers service management and development
- **IT Application Level Security**, which covers architectural styles and collaboration
- **IT Infrastructure Security**, which covers network, communication and application platforms
- **Central Security**, which covers authorisation, authentication, backup and recovery
- **Governance**, which covers policies, strategy, metrics, performance, legal and education

[20] Open Security Architecture, http://www.opensecurityarchitecture.org/

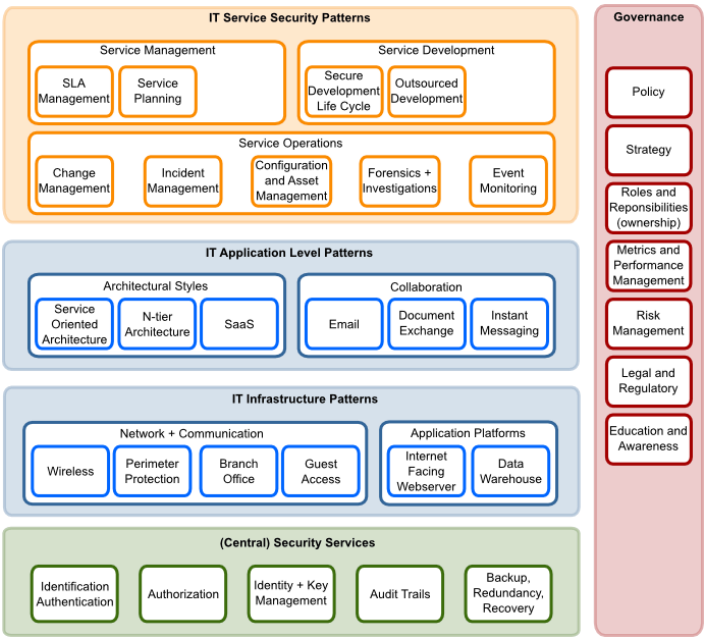


Figure 9-1 Security landscape

9.3 Challenges

CITI-SENSE will develop "citizens' observatories" to empower citizens to contribute to and participate in environmental governance, to enable them to support and influence community and societal priorities and associated decision making. To achieve this objective, data and the derived information play a pivotal role.

With an expected large number of sensor data providers capturing data and a large number of users consuming data, it will put additional constraints on the data platforms when it comes to performance, stability and flexibility. For example, it is expected that as part of the operational pilot phase there will be in excess of 500 individual sensors capturing data at time intervals between 1 and 30 minutes.

Also, as the nature of the data and information that is being exchanged in CITI-SENSE may contain privacy sensitive data, it is important that this data is exchanged in a safe and secure manner.

9.4 Scope

In this deliverable we only cover the security aspects that have been or will be implemented in the CITI-SENSE Data Platforms.

9.5 Security requirements

By combining the relevant, specific requirements from the pilot case studies in Work Packages 2 and 3, and common industry standard requirements from designing public data server platforms, the following security requirements have been identified.

Note that these requirements are not final and will evolve during the CITI-SENSE project's lifetime.

Requirement ID	Description
SR.1	The Platform shall only accept connections from authorised sources
SR.2	The Platform shall only publish data to authorised data consumers
SR.3	Data consumers shall only have read only privileges
SR.4	The physical infrastructure on which the Platform is implemented shall be securely housed, accessed, managed and protected.
SR.5	The Database, that forms part of the Platform, shall adopt a policy of restricted permissions for user accounts. Non-standard user credentials shall be applied.
SR.6	No non-essential software components or services shall be present or running on the Platform.
SR.7	Every component within the Platform shall utilise its individual firewall, and all ports not used will be blocked.
SR.8	Privacy sensitive data will be stored and exchanged securely.
SR.9	The Platform shall utilise load balancing to secure the availability and stability of the Platform.
SR.10	The Platform shall utilise redundancy so that single points of failure are minimised.

9.6 Legislation

In 1995 the EU Data Protection Directive (Directive 95/46/EC) came in force. This Directive focuses in particular on any personal information which can be used to identify an individual citizen, directly or indirectly. Driven by technological developments and globalisation, in January 2012 the European Commission proposed a comprehensive review of this directive, which will result in a single law that will help reinforce citizens' confidence in online services.

In addition to European legislation around the protection of data, which has been transposed into national legislation by EU member states, there may be additional national legislation in force that has an impact on which data is captured, exchanged and shared in the CITI-SENSE project as a whole. For example, the UK Data Protection Act[21] from 1998 has strong legal protection around exchanging data around sensitive issues such as health and requires anyone who handles this type of information to register in a central registry.

9.7 System architecture

The technical design of the Spatial Data and Services Platform is based on a tiered architecture with separated individual layers for logic (web services) and data (database). In addition to benefits for developers who can work on or modify a particular layer instead reworking the total infrastructure, this approach has the added benefit that any risk to the Platform is divided over multiple, decoupled, components. This means that there is no single point of failure.

[21] http://www.legislation.gov.uk/ukpga/1998/29/contents

9.8 Logical Service Interfaces and Information Model

In the CITI-SENSE SEDS Platform design individual components have been identified, each with a distinct purpose. On a high level, one can describe Data Ingestion Services, Data Storage Services and Data Publication Services. All three components are realised web services that are deployed in a cloud infrastructure.

In practise, the CITI-SENSE SEDS Platform is deployed on the Amazon Cloud using AWS (Amazon Web Services). AWS allows the Platform to be rapidly deployed on an established and mature cloud infrastructure (IaaS). AWS comes as standard with comprehensive security functionality and capabilities. This not only covers security of the physical infrastructure (e.g., servers), but also includes user access privileges, performance management and data encryption.

9.8.1 Physical and Environmental Security

The data centres from Amazon are physically housed over a large number of secure sites across the world. Access to these facilities is tightly controlled and monitored.
As described in Amazon's paper "Amazon Web Services: Overview of Security Processes[22]" the AWS infrastructure is compliant with a number of IT security standards, including ISO 27001[23], which covers a series of international open standards around information security management.

At this stage of the project (situation October 2014) the following service modules have been implemented:

- Data Storage Services
- Data Ingestion Services
- Data Publication Services.

As mentioned previously, all services are realised as web services and deployed on AWS. In the next section the security details will be discussed for each module.

9.8.2 Security per module

General

- The AWS Console service, which allows users to manage their AWS instances, is only accessible by authorised staff.
- All access credentials are stored and managed in the secure, password manager Lastpass (managed in-house by Snowflake Software).
- The NGINX reverse proxy server protects the Data Ingestion and Publication services from direct exposure to clients.
- The Primary and Secondary instances (e.g., WFS1 and WFS2) each reside in a separate subnet.
- The architecture includes a Network Address Translator (NAT) instance, which allows for remapping one IP address space into another by modifying network address information.
- None of the AWS instances have internet access, except the NAT instance.

[22] Amazon Web Services: Overview of Security Processes, November 2013, Amazon
[23] http://www.iso.org/iso/home/standards/management-standards/iso27001.htm

Data Storage Services

- Using Amazon Relational Database Service's (Amazon RDS) in-built functionality for:
  - Back-up: automatic back -ups of the database instance are made daily.
  - Fail-over: if the default database instance fails, a secondary database instance will be initiated immediately.
  - Maintaining database software: Minor updates to the database software are installed automatically.
- Automatic back-ups are made daily and retained for one day.
- Standby database instances are deployed in a separate AWS Availability Zone.
- Access control to the database instance via a master user account, which is used only within the context of Amazon RDS to control access to the database instance.
- As a default setting, no applications can connect to RDS. Security privileges have been applied in AWS to only allow access to the CITI-SENSE Data Ingestion and Data Publication services.
- Secure connections between application servers (Data Ingestion Services and Data Publication Services) via SSL.

Known issues, which will be addressed in further developments of the SEDS Platform

- All data is stored as clear text (not encrypted).
- Database uses default communication port.
- No distinct database user profiles with individual privileges have been set up.

Data Ingestion Services & Data Publication Services

- The Data Ingestion and Data Publication services are deployed on the Amazon Elastic Compute Cloud (EC2). Access to the Data Ingestion service requires a secure HTTP (https) connection. Clients can only access this service if they provide the correct credentials in their requests. Snowflake Software, who manages the SEDS Platform, is managing client credentials as well.
- AWS EC2 provides various levels of security, including security on:
  - the operating system (OS) of the host platform
  - the virtual instance OS, or guest OS
  - a firewall
  - signed API calls.
- All inbound ports are closed and only opened when required. This prohibits any unauthorised access to the web applications.

Known issues, which will be addressed in further developments of the SEDS Platform

- Opening EC2 ports is currently a manual task as no user registration and authentication component has been implemented yet. The current thinking is to implement an existing off-the-shelf technology, such as 3Scale (www.3scale.net), which provides online API management functionality.
- The application server, Apache Tomcat, uses default communication ports
- The data that is exchanged over the web services is not encrypted.

9.8.3 Security in SensApp

In order to secure communications between sensors and the SensApp Platform an encryption mechanism has been implemented. The data is encrypted on the client side and decrypted on the SensApp side using the software developed by the CITI-SENSE partner Ateknea.

9.8.4 GEOSS

The CITI-SENSE architecture will be connected to the GEOSS platform. Within the GEOSS Data Sharing Guidelines Working Group (DSGWG) there is current ongoing research into Authentication and Single Sign-On (SSO) to the GEOSS platform by external users and system, such as CITI-SENSE. In their report[24] the Working Group reports that a SSO federation has been established and demonstrated as a proof of concept by the COBWEB[25] project. The report mentions that "the federation was established as a SAML-2 federation that accepts certain OpenID visitors and allows them to be seen as SAML-2 users within the SAML-2 federation".

Security Assertion Markup Language (SAML) is an XML-based open standard data format for exchanging authentication and authorisation data between parties, in particular, between an identity provider and a service provider.

9.8.5 Data transport security in WFS

There are two main areas of interest to secure data transport using WFS web services:

1. Transport level security
2. Payload level security

Transport level security

This level covers the security of the technology that is used to transport the data over the web service. Existing standards such as HTTP and HTTPS are commonly adopted. HTTPS adds an additional layer of security to the standard HTTP standard. The Transport Security Layer (TSL) cryptographic protocol has been designed to provide communication security of the internet.

Payload level security

This level covers the security of the dataset (payload) itself. Data payload encryption can be achieved using any number of secure cryptographic methods, for example many data compression applications provide options to encrypt data whilst compressing.

9.9 Future developments

In the next releases of the CITI-SENSE Platform, described in deliverables D7.5 and onwards, further security improvements will be implemented.

[24] Data Sharing Guidelines Working Group (DSGWG) Engineering Report GEOSS Architecture Implementation Pilot Phase 6 (version 0.95), 28 April 2014.
[25] www.cobwebproject.eu

10 Ubiquitous Public Access Services

10.1 Overview

The web-based technology and mobile computing technology enable people to consume and produce a variety of forms of data and information. In particular, the location is one of the basic requirements that allows people to produce, distribute, and consume a wider range of geographic information. In this regards, ISO19154, "Geographic information – Ubiquitous public access – reference model (UPA)", was proposed and accepted as an international standard for "a seamless access at any time and any place that enables easy usage of geographic information and service."

The UPA aims to standardize a linking channel to provide spatial information in ubiquitous environment (i.e. cloud computing, sensor network etc.). To accomplish the purpose, UPA is defining service standards like Geo-context Awareness Service, Seamless access Service, Linking Open-geodata Service, Data Registration/Quality Assurance service etc. Figure 10-1 shows the ubiquitous access service in ISO19154.

Currently, Korea Institute of Construction Technology (KICT) and Saltlux are participating in the CITI-SENSE project to make a contribution to CITI-SENSE platform architecture design by making an example of the context information model based on ISO 19154 UPA available for analysis for use in CITI-SENSE. The context information model aims to provide a location-based air-pollution warning service so that citizens and decision-makers can quickly recognize and react to the air-pollution situation at their current locations or locations of interest.

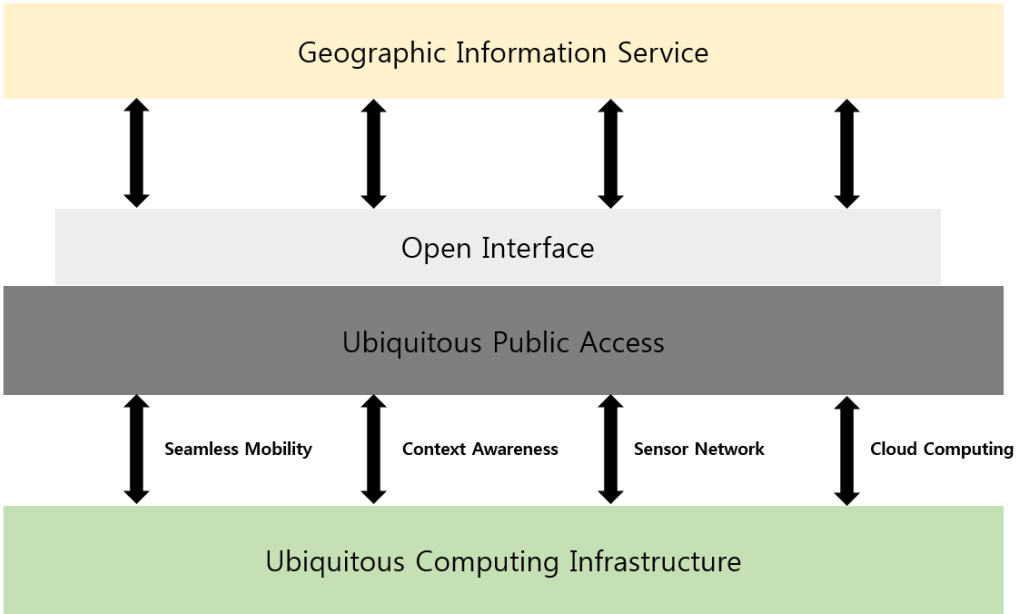


Figure 10-1 ISO19154: Ubiquitous Access Service

For verifying purposes, the UPA service model under development will be applied to test site "Seoul", the largest city in Korea. The service model will be analyzed with respect to future applicability to the

CITI-SENSE platform and EU test site, Figure 10-2. An implementation design for this example is further described in D7.5 part 2 – Annex K.

Seoul has been operating a sensor network for measuring air pollution in real-time, and the air pollution data are stored in a database to provide pubic with them as open data. The homepage shown in Figure 10-3 continuously shows the status of local air pollution on hourly-basis.

The sensor network in Seoul is measuring pollutants (PM, O_3, NO_2, CO and SO_2) from 25 urban-background sensors and 15 sensors along roads. The CAI (Comprehensive Air quality Index) along with each air pollutant data are with a map so that users can know air pollutants concentration database by the time and regions. In addition, the homepage provide one-day forecasts about PM, O_3, and yellow dust so that Seoul citizens are able to make a schedule for their outdoor activities.

In addition to the infrastructures, the 10 million population in Seoul is well suited for gathering citizens' opinion from SNS (Social Network Service). Also, as Seoul citizens' concerning and consideration on air quality is growing, Seoul is determined as an optimal test site to motivate citizens' participation to make air pollution policy.

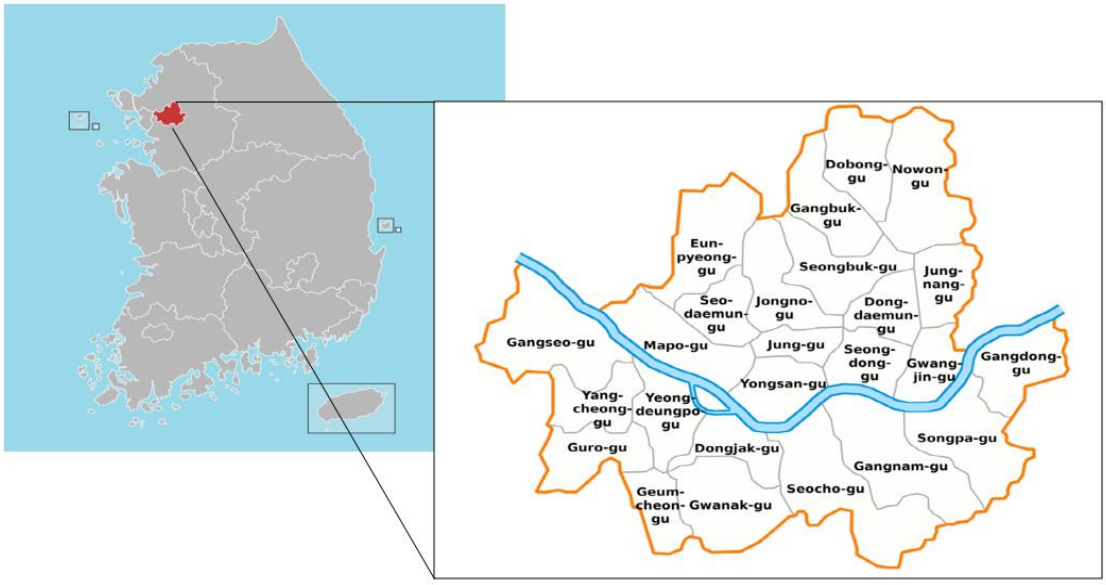


Figure 10-2 Test Site: Seoul

Table 10-1 Seoul Statistics

Item	Description
Area	605.25 km2
Number of Districts	25
Population	10,581,728
Population Density	17,000/km2

Figure 10-3 Seoul Air Quality Information (http://cleanair.seoul.go.kr/main.htm)

10.2 Development of Context Information Model for Air-Pollution Warning Service

An object-oriented analysis and design technique is used to design and implement the context information model, following the steps for defining the requirements, functional modelling based on the requirements definition, and brief architecture modelling using the functional modelling results. As a result, a use case model that consists of a use case diagram and description, and a schematic architectural design are proposed. The use case diagram and the product of the overall design process are composed on the basis of UML 2.0.

10.2.1 Requirements

10.2.1.1 Air Pollution Warning Service for Decision Maker

A decision maker, who is responsible for managing atmospheric environment information in Seoul, wants to develop a web system that allows them to quickly recognize and respond to atmospheric environmental risk status. This web system provides the overall status of atmospheric environment, current and forecast status of each air pollutant, citizens' reactions to atmospheric environment, and statistical information of atmospheric environment.

Three types of sensors are employed for the UPA services, in which the physical sensor detects air pollution by Gu (district) and air pollution along roadsides. The public data is an archived data in database for calculating air pollution history (e.g., frequency and number of days with pollution); and the social sensor detects feedback and reaction information on SNS regarding atmospheric environment. The major functional requirements are as follows.

- Comprehensive air quality information service
  - ✓ Displays summary information, including real-time information, by monitoring station, updates on forecast

and alert status, citizen reaction information, and CAI(Comprehensive Air-Pollution Index) level on main screen.

- Real-time air pollution warning service
  - ✓ Information on the fine dust, ultrafine dust, ozone, nitrogen dioxide, carbon monoxide, and sulfite gas per Gu (district) is provided.
  - ✓ Provides an action guideline according to forecast and alert status.
- Air-pollution forecast service
  - ✓ Provides forecast and alert updates by atmospheric environment monitoring item.
  - ✓ Provides an action guideline according to forecast and alert status.
- Citizens' reaction
  - ✓ Provides citizen SNS (Twitter, Facebook, personal blogs, etc.) reaction monitoring information according to the atmospheric environment risk occurrence status.
  - ✓ Provides function for checking reactions by Gu (district) and item.
- Air quality statistics information offering service
  - ✓ Tabulates average pollution level (annual) by Gu (district) in Seoul.
  - ✓ Maps and charts on graph pollution level information by zone (residential area, commercial area, industrial area, green area, etc.).
  - ✓ Charts on graph long-term trend information.
  - ✓ Tabulates and charts on graph information on number of forecast and alerts issuance.
  - ✓ Provides statistical information of citizen reactions.

10.2.1.2 Air Pollution Warning Service for Citizens

The air-pollution warning service is provided to citizens sensitive to air pollution, employees in relevant organizations, and respiratory disease patients. The major functional requirements are as follows.

- Comprehensive air quality information service
  - ✓ Displays summary information, including real-time information by monitoring station, updates on forecast and alert status, citizen reaction information, and H-CAI level on main screen.
- Location-based air pollution service
  - ✓ Provides air pollution risk information according to location.
- Schedule-based air pollution service
  - ✓ Provides air pollution information according to personal schedule (e.g., users can save their plan to play a basketball game with friends in the afternoon in the schedule in advance, and be provided with the

environment forecast service regarding the location).

- Optimal place query service
  - ✓ In case of outdoor activities, recommends the best place considering air pollution.
  - ✓ In case of outdoor activities, including picnics organized by daycare centers, hiking, and outdoor sports, recommends a place safe from atmospheric environment risk.
- Statistics of personal exposure to atmospheric environment risk offering service
  - ✓ Provides statistics service regarding the (five) atmospheric environmental risk factors to which individuals are exposed during the day.
- Comments
  - ✓ Allows users to leave comments on air pollution regarding residential area or venues visited by them.
  - ✓ Allows users to leave comments when fine dust or yellow dust problems become a social issue.
- Forecast and alert on air quality
  - ✓ Sounds risk alarm when risk alert is issued in current location or venue scheduled to visit.

10.2.1.3 UPA-related service

Based on the UPA GeoSemantic Context of ISO 19154, the following service is provided

- Air pollution alert service according to road proximity
  - ✓ Assigns weights based on the level of pollution in the Gu (district) where the user is located if the location of said user is near a road.
  - ✓ Conditions to consider when assigning weights

    - Type of road (avenue/street/road/path)
    - Road proximity (close/far, 10 m/50 m/100 m)
    - Monitoring time (rush hour/weekends/holidays)

10.2.2 Use case model

The use case model consists of a use case diagram and description. Among the requirements described in the previous section, a use case model is suggested with regard to the representative function of context awareness-based atmospheric environment risk information offering service. A use case diagram is classified into the following two subcategories: Level 1 use case diagram, where the functionality is modelled in an outline level; and Level 2 use case diagram, where the functionality is modelled in a more specific level.

10.2.2.1 Level 1 Use Case Diagram

The context-aware air pollution information service model consists of (1) a real-time air pollution warning service, (2) a schedule-based air pollution warning service, and (3) an open data-based air

pollution statistics service. Users of the context-aware air pollution information service consist of citizens sensitive to air pollution, respiratory disease patients, and employees of relevant organizations. Here, relevant organizations refer to organizations sensitive to air pollution, e.g., elementary schools or hospitals.

The use case diagram in Figure 10-4 describes the functionality of the context-aware atmospheric environment information service within Level 1. To be provided with the context-aware air pollution information service, users basically need to agree to provide a location and register affiliation, health condition, medical history, age, and future location-based movement schedule.
Utilizing the user current location information, users are provided with a real-time air pollution warning service. Here, if the user is near a road, weight is given according to road proximity. In addition, users are provided with an air pollution forecast warning service based on registered future movement schedule, and can retrieve a statistics index regarding air pollution in the region scheduled to visit.

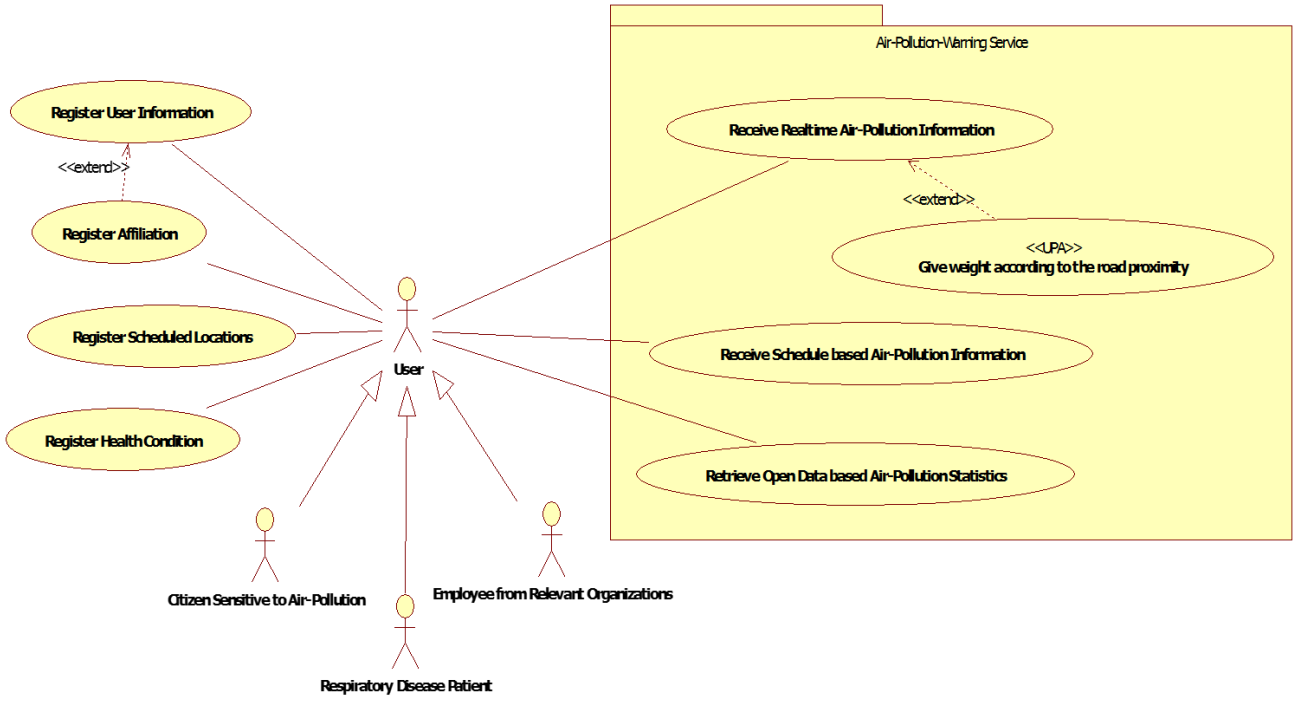


Figure 10-4 Level 1 use case diagram for context-aware air pollution service

10.2.2.2 Level 2 use case diagram

Real-time air pollution warning service based on users' location

The location-based real-time air pollution warning service provides air pollution information according to the user current location, and information on how to cope with the situation. Figure

10-5 specifies the functionalities of the location-based real-time air pollution warning service using the Level 2 use case diagram.

Based on current location information, users are provided with real-time pollution information by air pollutants such as PM, O_3, NO_2, CO, and SO_2, in addition to the real-time comprehensive air quality index. The use cases Receive Real-time PM Info, Receive Real-time O3 Info, Receive Real-time NO2 Info, Receive Real-time CO Info, Receive Real-time SO2 Info, and Receive Real-time CAI Info provide this function. Here, if a user is located near a road, weight is given in consideration of several conditions, including the type of road (avenue/street/road/path), road proximity (close/far, 10 m/50 m/100 m), and monitoring time (rush hour/weekends/holidays). The use case Give Weight According to Road Proximity provides this function.

To receive the current location-based air pollution warning service, users basically need to agree to provide their location and register personal information including affiliation, health condition, and age. The use cases Register User Information, Register Affiliation, Register Health Condition, and Register Current User Location are responsible for these functions.

If the pollution level exceeds the safe range or the acceptable level given by relevant organizations in consideration of users' health condition, real-time air pollution information and action knowledge are provided to users and employees of relevant organizations so they can respond quickly. The use cases Notify Health-based Action Knowhow and Notify Organizations' Action Knowhow provide these functions and are connected to the functions that provide real-time pollution information in an <<extend>> relationship.

In addition, users can leave comments regarding air pollution information and action knowledge information on social websites so that the information can be used as social data, and citizens and policy regulators interested in air pollution can respond properly. This function is managed by the use case Write Opinion.

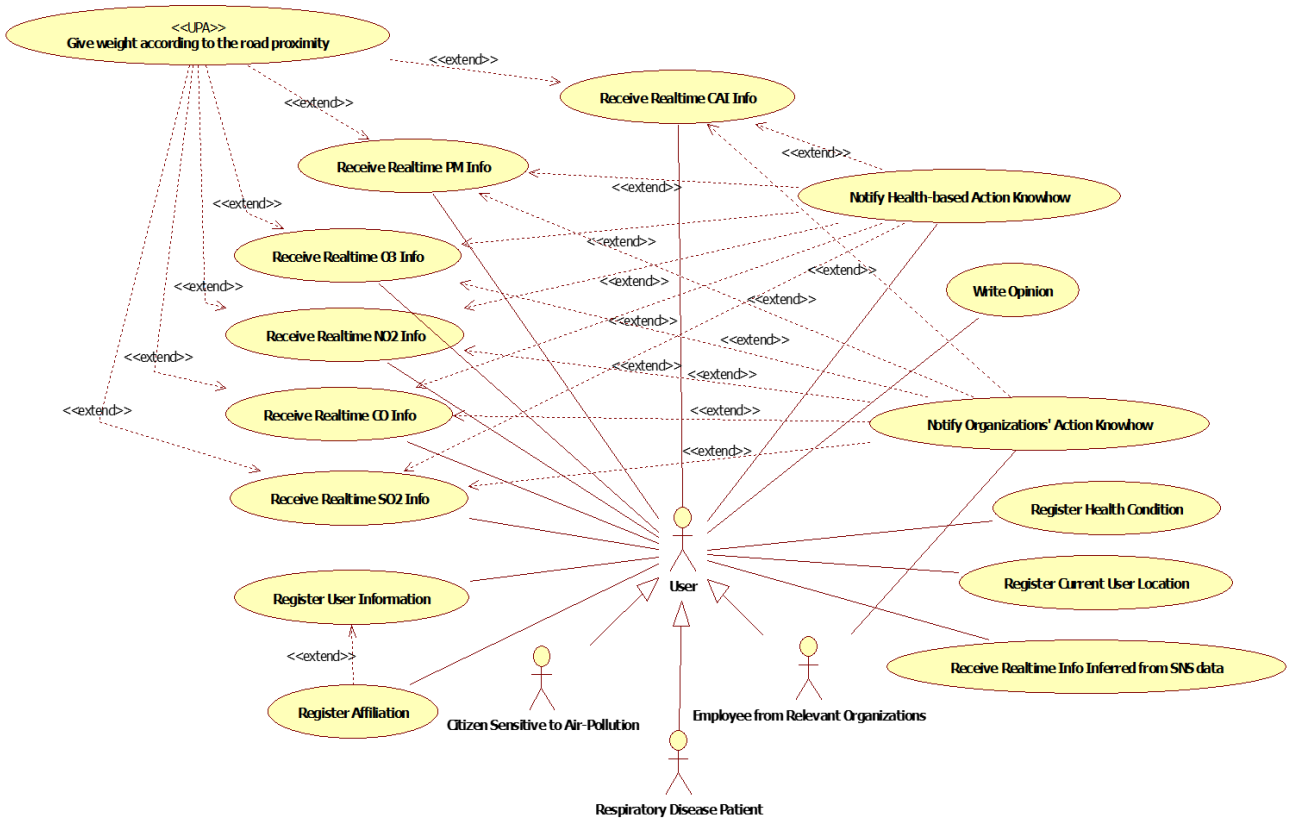


Figure 10-5 Level 2 Use case diagram for real-time air pollution warning service

Schedule-based air pollution warning service

The schedule-based air pollution warning service provides a registered schedule for users, i.e., the expected air pollution information and action knowledge according to expected movement location and time. Figure 10-7 specifies the functionalities of the schedule-based air pollution warning service using the Level 2 use case diagram.

In cases where users have planned outdoor activities, such as business trips and travel, users register their expected movement location and time information in the system and receive air pollution information of the region the users expect to visit. Employees working in schools, public health centers, or hospitals receive air pollution forecast based on workplace location and action knowledge according to the air pollution situation, and actively cope with the situation so that damage resulting from air pollution is minimized.

The use cases Receive Schedule-based PM Info, Receive Schedule-based O3 Info, Receive Schedule-based NO2 Info, Receive Schedule-based CO Info, Receive Schedule-based SO2 Info, and Receive Schedule-based CAI Info are the functions that provide pollution information based on schedule, and to use these functions, users and organization information should be registered through the use cases Register User Information, Register Affiliation, Register Health Condition, Register Scheduled Locations, etc. In addition, when the air pollution forecast exceeds the criteria according to the

registered information, action knowledge is notified to users and organizations through the use cases Notify Health-based Action Knowhow, Notify Organizations' Action Knowhow.

Furthermore, users can leave comments on the accuracy of air pollution forecast and manage situations in this regard so that citizens and relevant policy managers can properly prepare and improve the relevant system. This is managed by the use case Write Opinion.

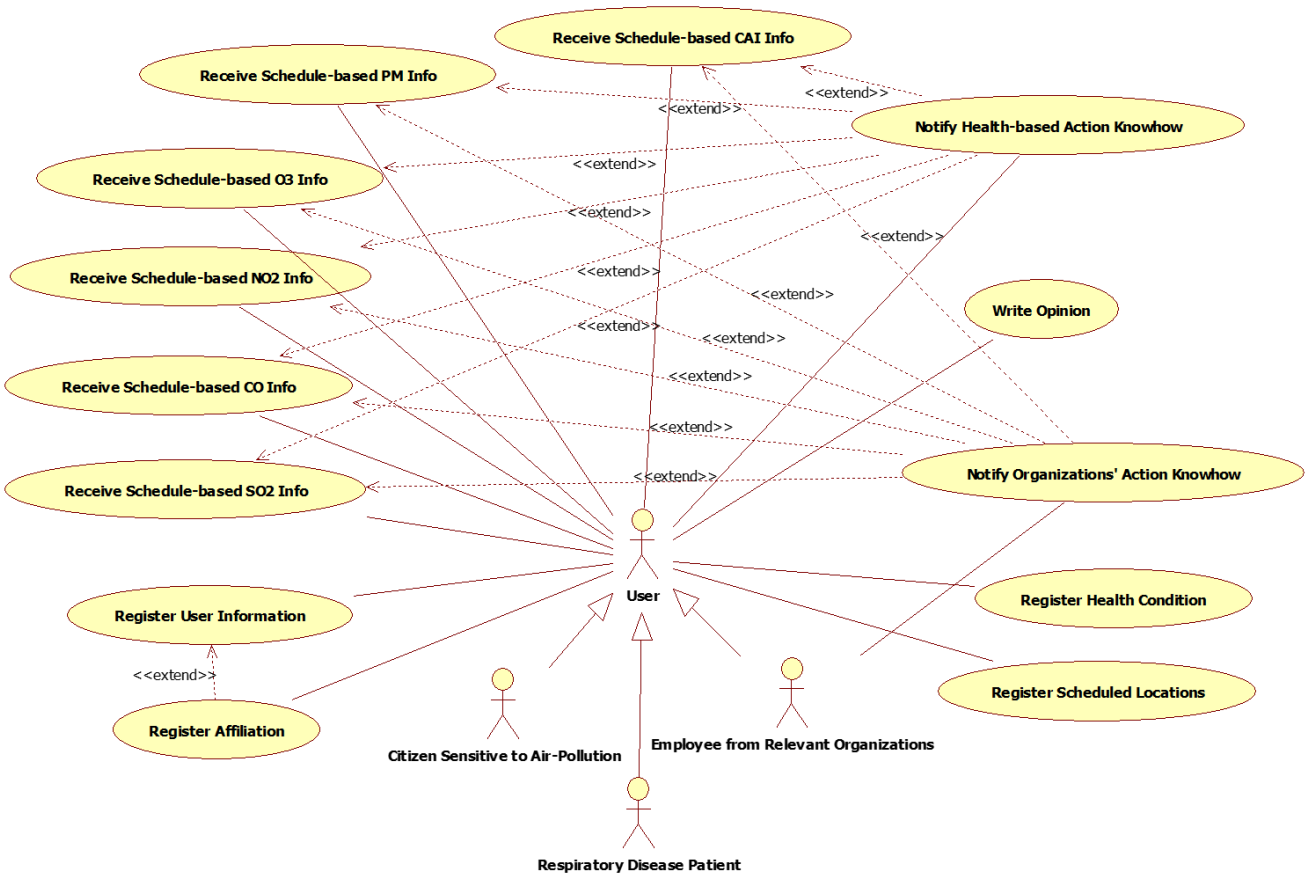


Figure 10-6 Level 2 use case diagram for schedule-based air-pollution warning service

Open data-based air pollution statistics service

The open data-based air pollution service uses data saved from the air pollution monitoring network to provide air pollution information and action knowledge according to the location, schedule, region of interest, and expected location of the user. The following figure specifies the functionality of the open data-based air pollution warning service using the Level 2 use case diagram.

Open data manages the statistics information of air pollution. Users input a region of interest and receive the statistic index (frequency and lasting time) by air pollutant of the relevant region, and the statistics index of the comprehensive air quality index (frequency and lasting time). The frequency information offering function is managed by the use cases Retrieve PM Frequency, Retrieve O3 Frequency, Retrieve NO2 Frequency, Retrieve CO Frequency, Retrieve SO2 Frequency, and Retrieve CAI Frequency; the lasting time information offering function is managed by the use cases Retrieve

PM Duration, Retrieve O3 Duration, Retrieve NO2 Duration, Retrieve CO Duration, Retrieve SO2 Duration and Retrieve CAI Duration.

Furthermore, air pollution-related precautions are provided to users in relevant regions according to the statistics index. This function is provided by the use case Notify Precaution to Local Resident, and to this end, information on the user residential area and region of interest should be registered through the use cases Register User Information and Register Location in Interest.

Lastly, users can leave comments based on the air pollution statistics index in regions of interest, such as residential areas, schools, hometowns, etc., so that policy managers from the relevant regions can use comments as reference materials to establish administration policies regarding air pollution. This part is covered by the use case Write Opinion.

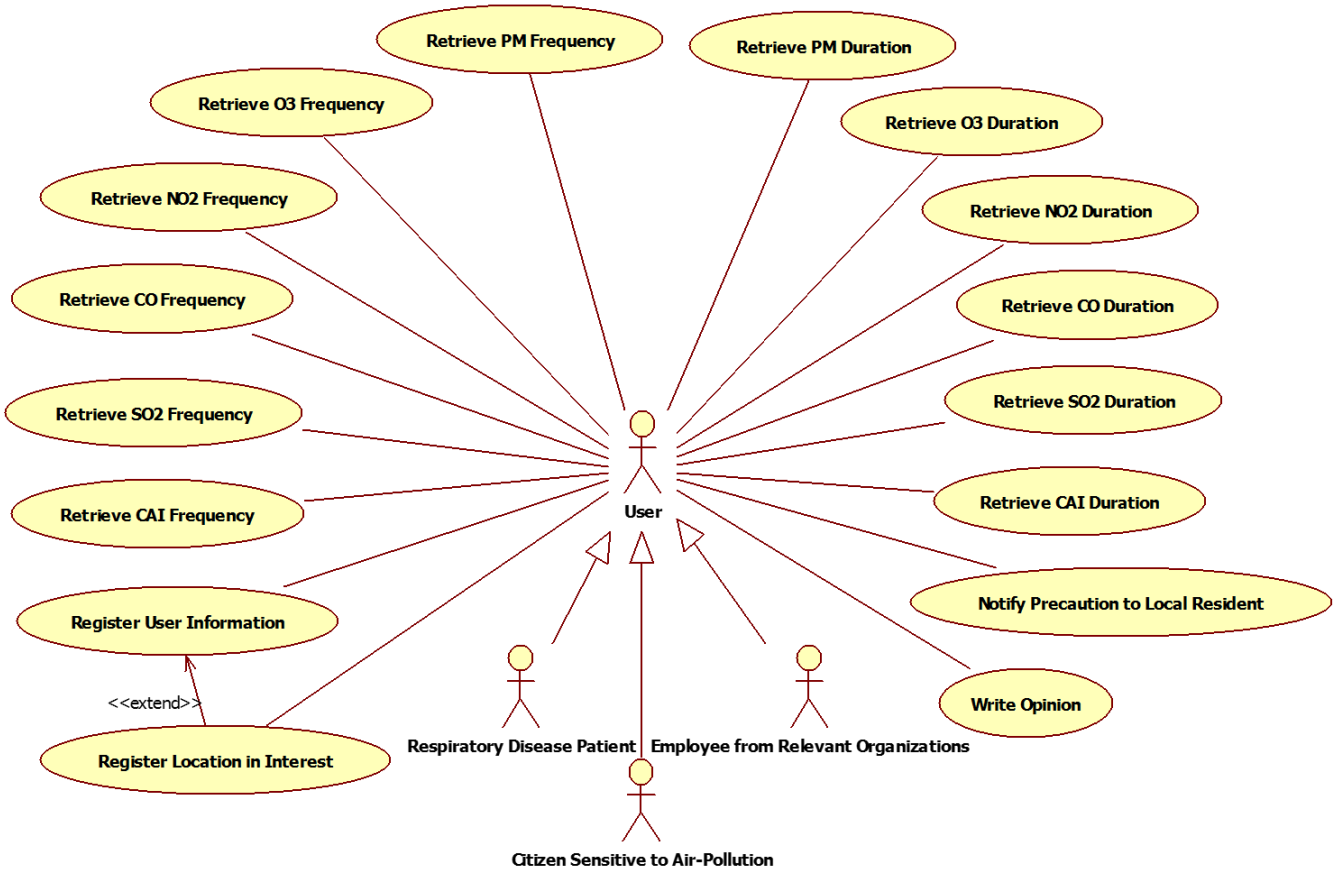


Figure 10-7 Level 2 use case diagram for Air pollution statistics GeoSemantic function

To provide UPA GeoSemantic-related services to users, a function to manage this should be provided. Figure 10-7 describes the GeoSemantic management-related function using the Level 2 use case diagram.
GeoSemantic Context is an inferred GeoSemantic Context that includes the user location context. Managers should be able to register, retrieve, correct, and delete the GeoSemantic context and be

able to register, retrieve, correct, or delete the GeoSemantic Context Rule necessary to generate GeoSemantic context.

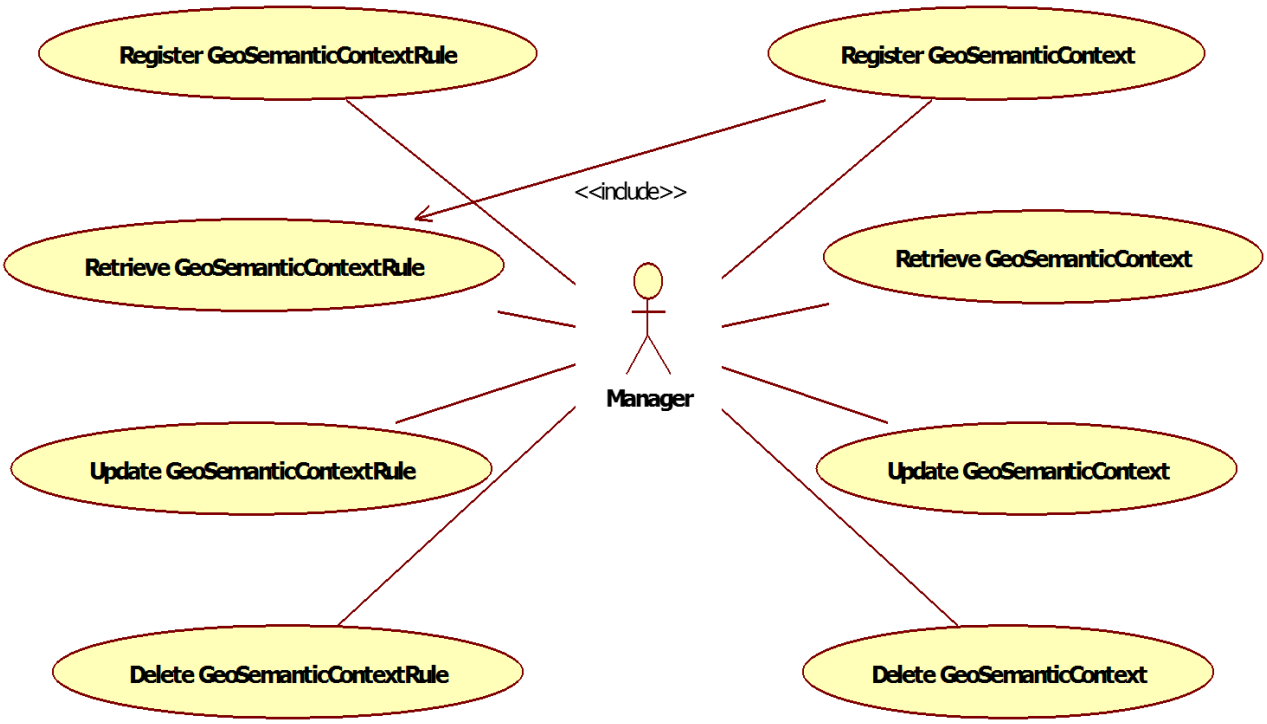


Figure 10-8 Level2 use case diagram regarding GeoSemantic management-related function

10.2.3 Use case description

In this section, the use case specifications are provided by selecting representative use cases from the Level 2 use case diagram.

10.2.3.1 Receive Real-time PM Info use case

Overview

This use case offers the function of notifying information on real-time fine dust pollution levels according to current user location or organization location to citizens sensitive to air pollution, respiratory disease patients, and employees from relevant organizations.

Pre-condition

Users should agree to provide their location so that personal location information can be recognized.

Post-condition

Fine dust levels at the current user location are sent to the user via text message or email.

Main flow

Actor	System
Register current user location. [Info: current location]	
	Retrieve current fine dust levels using registered location information. In case it is impossible to retrieve fine dust information regarding registered region, call A-1.
	If user is located near a road, call use case <<UPA>> Give weight according to road proximity.
	If user is an individual and retrieved fine dust levels are slightly bad, bad, or very bad considering patient history or health condition of registered user, call use case Notify Health-based Action Knowhow in consideration of user medical history or health condition.
	If user is an organization such as hospital or elementary school and retrieved fine dust levels re slightly bad, bad, or very bad, call use case Notify Organizations Action Knowhow.

Sub flow

- A-1

Actor	System
	Notify user that there is no fine dust information regarding their current location.
	Terminate use case.

10.2.3.2 Notify Health-based Action Knowhow use case

Overview

This use case performs the function of notifying the action knowledge according to the pollution level by pollutant to users.

Pre-condition
User information should be registered.

Post-condition
Action knowledge should be sent to users via text message or email.

Main flow

Actor	System
Request function while sending information, including user information, pollutant, and pollution index.	
	Retrieve conditions with regard to user being an asthmatic patient, child, or cardiovascular patient.
	As shown in Fig. 10-6, send users a text message or email of proper action knowledge according to pollution index by pollutant, in accordance with user condition.

Table 10-2 Comprehensive Air-quality Index (CAI) and its Implication

Pollution	Good	Moderate	Lightly Bad	Bad	Severe	
CAI	0~50	51~100	101~150	151~250	251~350	351~500
PM(μg/m³)	0~30	31~80	81~120	121~200	201~300	301~600
O3(ppm)	0~0.040	0.041~0.080	0.081~0.120	0.121~0.300	0.301~0.500	0.51~0.600
NO2(ppm)	0~0.030	0.031 ~ 0.060	0.061~0.150	0.151~ 0.200	0.201~0.600	0.601~2
CO(ppm)	0~2.00	2.01 ~ 9.00	9.01~12.00	12.01~15.00	15.01~30.00	30.01~50
SO2(ppm)	0~0.020	0.021~0.050	0.051~0.100	0.101~0.150	0.151~0.400	0.401~1

Pollution	Health Implications
Good	No health implications
Moderate	Meager impact on patients in case of chronic exposure
Lightly Bad	Harmful impacts on patients and members of sensitive groups.

Bad	Harmful impacts on patients and members of sensitive groups (children, aged or weak people), and also cause the general public unpleasant feelings
Servere	Life-threatening impacts on patients and members of sensitive groups, and harmful impacts on the general public

10.2.3.3 Notify Organizations Action Knowhow use case

Overview

This use case performs the function of notifying the action knowledge according to pollution level by pollutant to organizations, including schools, public health centers, and hospitals.

Pre-condition

Organization information should be registered.

Post-condition

Action knowledge is sent to organization employees via text message or email.

Main flow

Actor	System
Request function while sending information, including relevant location, pollutant, and pollution index.	
	Retrieve nearby organizations such as hospitals, schools, public health centers, and livestock farmhouses.
	Notify proper action knowledge according to pollution index of each pollutant, as shown in Fig. 8, to retrieved organizations.

When Asian dust has been forecast	
Home	• Since Asian dust causes such conditions as allergic conjunctivitis, rhinitis, and bronchial asthma, children and the infirm or elderly should refrain from going outside • Vulnerable persons, or elderly or those with respiratory diseases, should avoid any outdoor activity • Wash hands and feet thoroughly immediately after returning home • To prevent Asian sand from getting indoors, check the windows and other potential points of entry
Educational institutions	• Provide students and parents with guidance and tips on how to deal with the hazards of yellow sand/Asian dust
In areas for livestock or horticulture	• Prepare to move livestock from paddocks or pastures into a covered shelter • Prepare coverings to protect straw/hay for livestock feed that is stacked or lying on the ground • Inspect equipment used for cleaning off yellow sand • Inspect doors and windows of facilities
When an Asian dust advisory/warning is in effect	
Home	• Close the windows to prevent Asian sand from getting indoors. Vulnerable persons and elderly or those with respiratory diseases, should avoid any outdoor activity • Avoid going outside if possible. If it is absolutely necessary to go out, wear protective glasses, mask, and long sleeves. After returning home, immediately wash hands and feet thoroughly and brush your teeth. • Drink water frequently and keep indoor air clean with air purifiers and humidifiers • Before preparing agricultural or marine products that have been exposed to yellow sand, e.g., vegetables, fruits, and fish, wash them thoroughly
Educational institutions	• Ban outdoor activities for kindergarteners and elementary school students, and consider shortening school hours or closing schools • Suspend or postpone special outside activities like field trips and sports events
In areas for livestock or horticulture	• To prevent their exposure to yellow sand, shelter livestock in covered barns rather than leaving them outside in paddocks or pastures • Close doors and windows of barns and stables to minimize contact with external air and entry of yellow sand • Use tarps or plastic sheets to cover straw/hay for livestock feed that is stacked or lying on the ground • Close doors and windows of facilities like plastic greenhouses and hothouses.
After Asian dust has passed	
Home	• Ventilate the house to change stale indoor air • Before using any items exposed to and contaminated by yellow sand, wash them thoroughly
Educational institutions	• Remove dust by cleaning the school inside and outside • Monitor the condition of student health, and allow students suffering from disorders like colds, eye trouble, and itching skin to take it easy or go home early, and urge them to consult a specialist. • Inoculate students against infectious diseases that may result from exposure to yellow sand, and disinfect areas where students gather
In areas for livestock or horticulture	• Clean and disinfect plastic greenhouses, barns, feeding troughs in pastures, equipment that comes into contact with livestock • After dusting off yellow sand adhering to livestock, fumigate the animals • Monitor the occurrence of any diseases for 2 weeks following yellow sand

Figure 10-9 Action knowledge in case of yellow dust

10.2.3.4 Receive Schedule-based PM Info use case

Overview

This use case is called by citizens sensitive to air pollution, respiratory disease patients, and employees from relevant organizations, and provides information on fine dust pollution levels according to user personal schedule.

Pre-condition

Personal schedules should be registered.

Post-condition

Air pollution information is sent to users via text message or email.

Main flow

Actor	System
User requests function.	
	Retrieve air pollution information using region information registered on schedule of following day. In case it is impossible to retrieve air pollution information regarding registered region, call A-1.
	Predict air pollution levels using registered schedule.
	If levels are beyond dangerous level, call use cases Notify Health-based Action Knowhow, Notify Organizations Action Knowhow.

Sub flow

- A-1

Actor	System
	Notify user that prediction is impossible because of lack of air pollution information regarding registered region.
	Terminate use case.

10.2.3.5 Register User Information use case

Overview

This use case offers a function to register personal information, including user health condition, which is required to receive the air pollution service.

Pre-condition

None

Post-condition

User personal information should be registered.

Main flow

Actor	System
Input user basic information and request system to generate user account. [Info: name, gender, address, birthday, phone number]	
	Register basic information and request additional information, including health condition, medical history, and affiliation, in order to receive air pollution warning service.
Input additional personal information required to receive air pollution warning service. [Info: health condition, medical history, affiliation]	
	Save additional personal information.
	In case of organization sensitive to air pollution, such as schools and hospitals, call use case Register Affiliation.
	When user wants to additionally receive air pollution service on a certain region of interest, call Register Location of Interest.

10.2.3.6 Register Scheduled Locations use case

Overview

This use case offers a function to register user schedule in order to receive the air pollution service based on said schedule. If the user is an individual, the schedule could be personal appointments, travel, etc., and if the user is an organization, the schedule could be picnics, official outdoor events, etc.

Pre-condition
User information should be registered.

Post-condition
Schedule regarding individuals or organizations is registered.

Main flow

Actor	System
Input user schedule information. [Info: Expected movement location, date, time]	
	Save user schedule information.
	If user is an organization, go to A-1. Additional information is requested.

Sub flow

- A-1

Actor	System
	Request additional information when user wants to register organization schedule.
Input information regarding event content, number of participants, age group, characteristics (health) of people participating, etc.	
	Save organization schedule information.

10.2.4 Class diagram

Figure 10-10 shows the class diagram of air pollution services, which is combined with the class diagram of a CITI-SENSE platform in the blue zone. Different from the City class in the CITI-SENSE Platform, the City class diagram in this research describes detailed information of districts in the Seoul city. Also, users are further classified into normal citizens, patient, and organization. In the class diagram, all users are basically required to register their personal information such as name, e-mail,

and phone number. However, additional information can be further required, depending on the types of user. For example, citizens, who are sensitive to an air pollution, needs to register an air quality information of interest. Also, for the patient class, disease type and its history should be registered for air pollution warning services.

In addition to the air quality information in real-time, users can also express their feeling with a symbol or write opinions to the air quality in their locations. Four different symbols, which are good, normal, bad, and serious, can be chosen multiple times depending on different time intervals and locations. The opinion class allows users to register their opinions with a text, with which their opinions to the air quality can be written. Similar to the opinion class, users can register their opinions multiple times.

The schedule class allows users to register their personal schedules on a district basis. The LocationalContext Class, which conforms the ISO19154, manages the users' locations with UPA_LocationContextRule with UPA_LocationalContextElement. In the air quality information system, a user's locational context element is represented with a street address and a geographic coordinate system (latitude and longitude). The geospatial context class can then retrieve the user's geospatial context with the locational element as an input parameter. The GeoSemanticContext class depends on the LocationalContext class and the GeospatialContext class. Thus, the geosemantic context can be derived from each or both classes. The public class manages a set of air pollution data from a sensor network and utilizes them to compute statistics data such as average, mean, and trend.

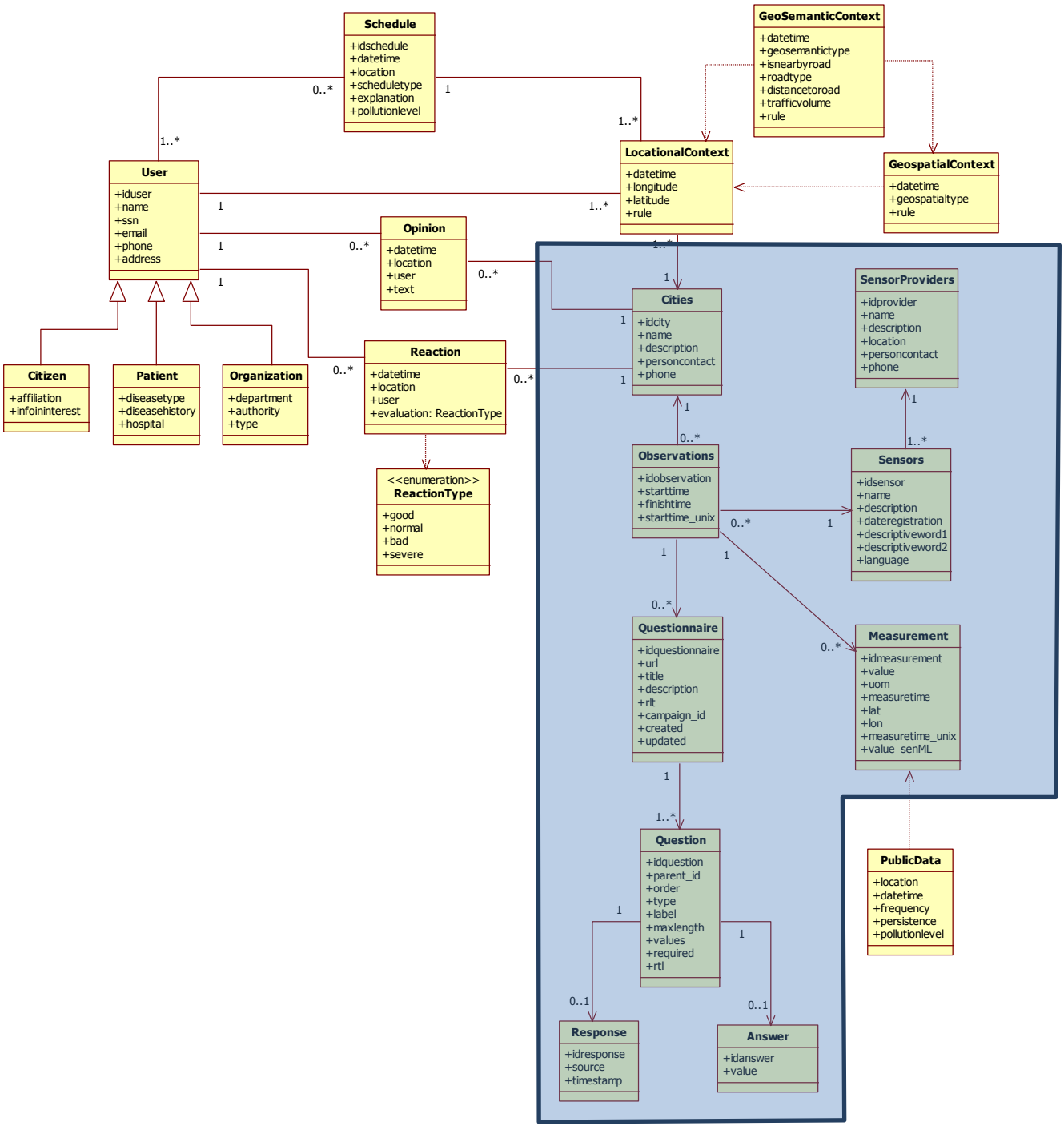


Figure 10-10 Class diagram with user context extension

10.2.5 Sequence diagram

In this section, sequence diagrams are provided for the major use cases.

10.2.5.1 Sequence diagram for receiving Real-time PM Info

In order to develop Receive Real-time PM Info based on the use case description suggested in the section 10.2.3, the interaction between the classes analyzed is modeled as shown in Figure 10-11.

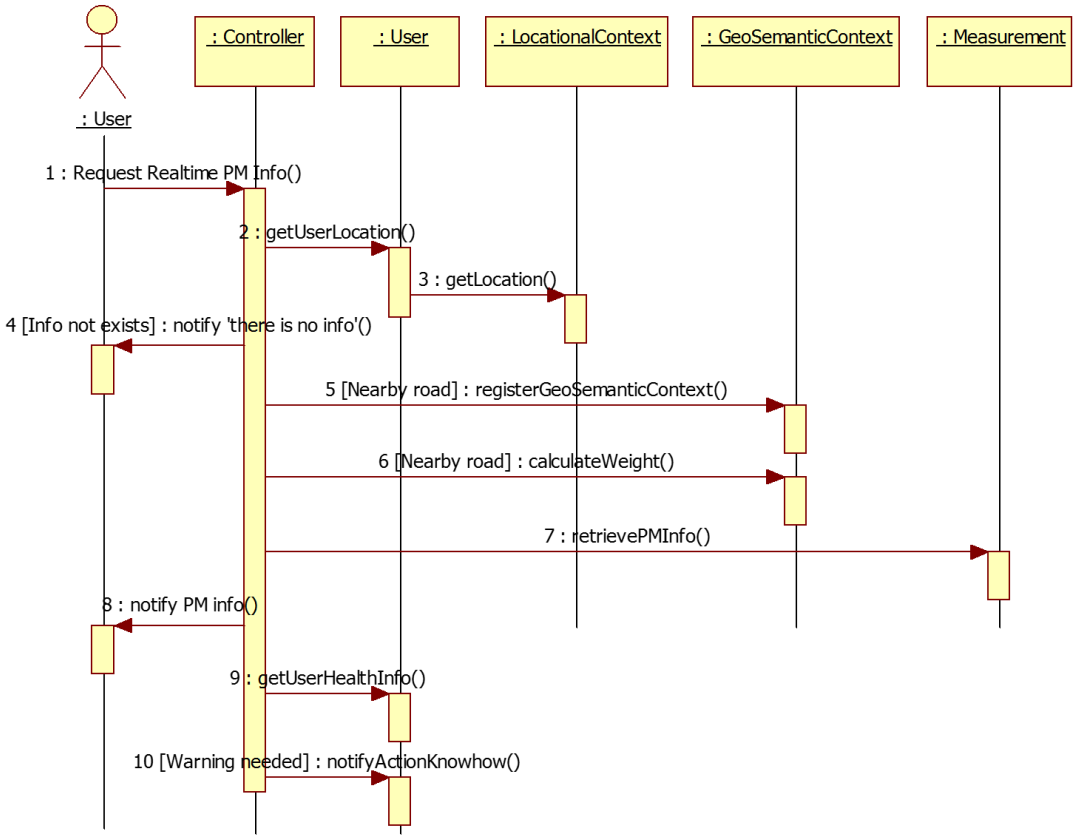


Figure 10-11 Sequence diagram for receiving Real-time PM Info

10.2.5.2 Sequence diagram for Receive Schedule-based PM Info

In order to develop Receive Schedule-based PM Info based on the use case description suggested the section 10.2.3, the interaction between the classes analyzed in the section 10.2.4 is modelled as shown in Figure 10-12.

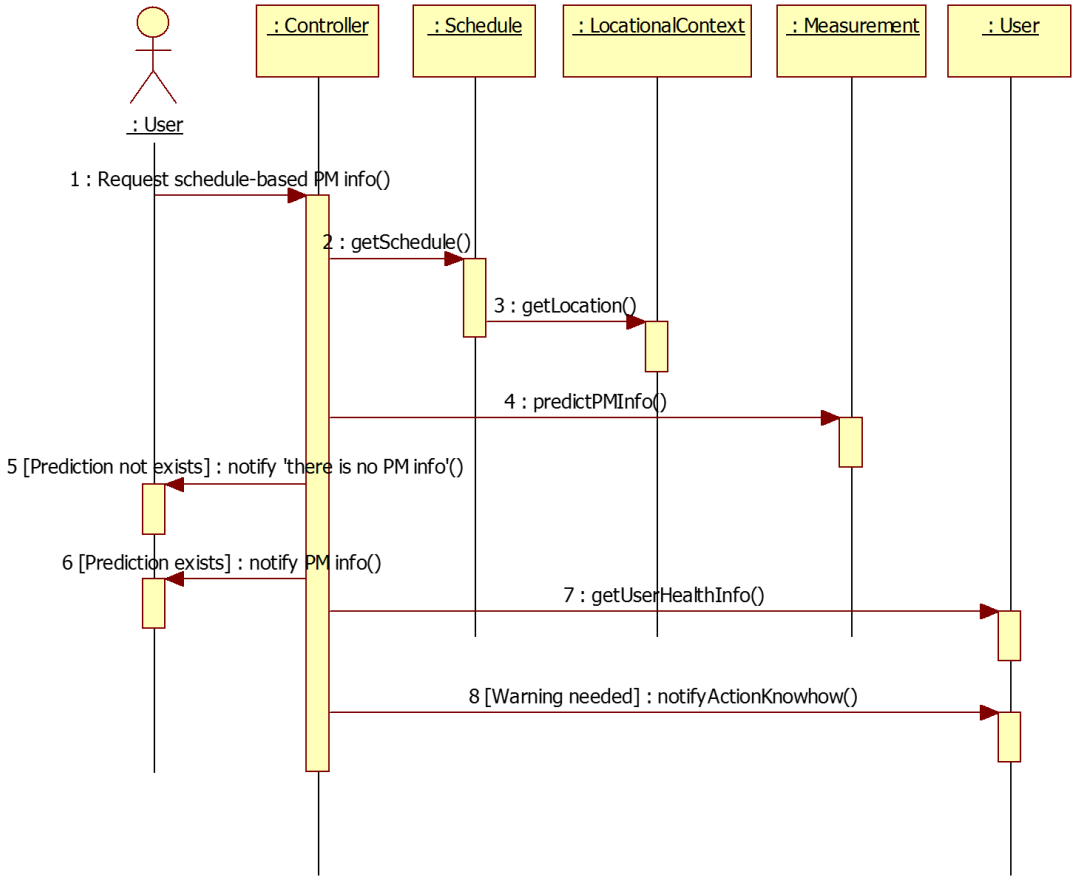


Figure 10-12 Sequence diagram for Receive Schedule-based PM Info

10.2.6 Architecture

Figure 10-13 shows an architecture overview of the pilot system in Seoul. The architecture is described by dependency relationship according to the call relationship between the main elements and the elements. The architecture is largely classified into presentation layer, logic layer, and data layer, and each layer consists of components. In addition, the Sensor Data Acquisition interface is implemented by the components from the data layer.

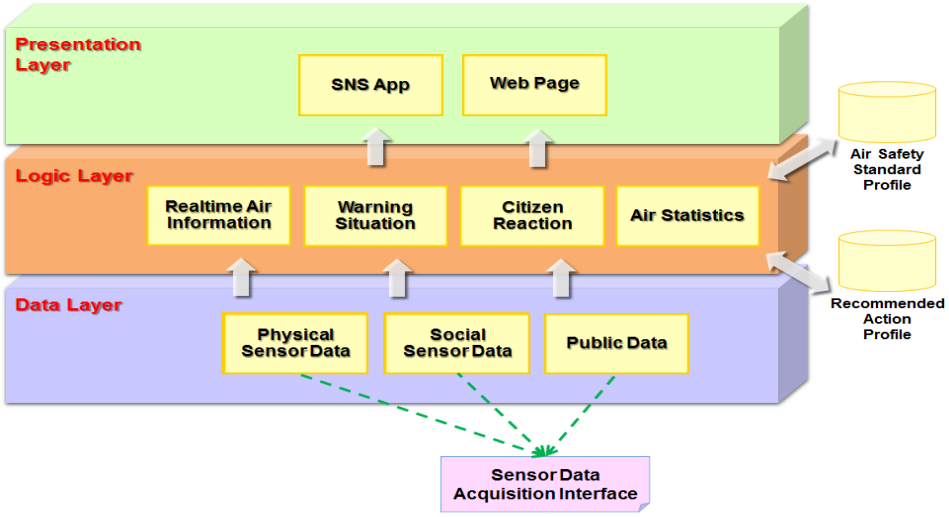


Figure 10-13 Architecture of the Pilot System

- Software configuration

The software overall configuration is as follows: Database stores the data, Web Server collects and provides data, and Web Client provides the UPA service. Here, the entire software is configured based on OpenSource.

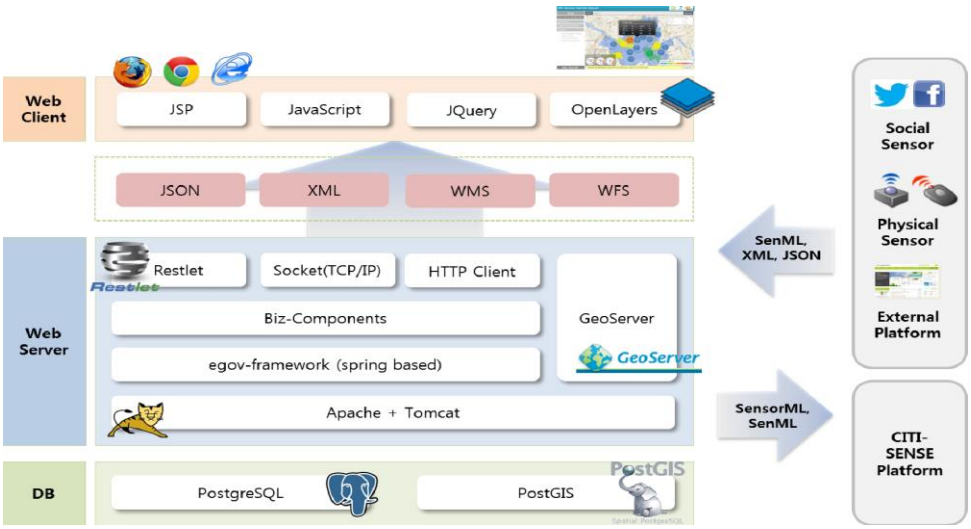


Figure 10-14 Software Configuration of the Pilot System

- Database

With regard to the database, PostgreSQL, which is the DBMS provided as OpenSource, is used to save data, and the PostGIS plug-in is installed and used in order to process the Geometry data.

- Web Server

The web server is established based on Apache/Tomcat. To provide the overall web service, the electronic government-framework (egov-framework: spring-framework-based), which is used as a public business development framework, is used. Furthermore, data are provided through web client and HTTP communication using Restlet, which provides RESTful service; a Socket module and HTTP Client module are included in the configuration to connect with external systems.

To provide service regarding Geographic Information System (GIS) data, Web Map Service (WMS) and Web Feature Service (WFS) are provided using GeoServer, and GeoServer is used in connection with the OpenLayers of the web client.

- Web client

The web client is developed on the basis of JSP and Javascript. Its User Interface (UI) is supplemented by adding the JQuery library, and map data rendering is facilitated using the OpenLayers library.

- Data exchange

Ordinary data delivered from the web server to the web client is provided using a data type that can be easily managed with Javascript, such as JSON and SML, through the RESTful web service. Geospatial data is provided through GeoServer using a type of WMS and WFS data, which are the Open Geospatial Consortium (OGC) standards.

In the case of collecting sensor data, such information is gathered using the data types SenML, XML, JSON, etc., depending on the connection method. When providing data to the CITI-SENSE Platform is necessary, such data are provided using a data type (SenML, etc.) required by the Platform.

- Screen layout

The overall screen layout of the UPA Service Test Site is as follows: a menu bar on the left, a map on the right, and information regarding real-time forecast and alert at the bottom of the screen.
The menu on the left is configured to allow users to select the type of information to be provided, and the selections provided on the menu are as follows:

  - Real-time Air Quality Info
  - Alert Status
  - Citizen Reaction Status
  - Statistics
  - Annual Average Air Pollution
  - Pollution trends
  - Annual Statistical Alert
  - Citizen Reaction Statistics

If users select air pollution information such as H-CAI, fine dust, ultrafine dust, ozone, nitrogen dioxide, carbon monoxide, sulfite gas, etc., from the map on the right, real-time information is displayed on the map for each Gu (district) in Seoul. Here, different colors are displayed according to pollution range. Furthermore, if the mouse is placed on the symbol marked on the map, comprehensive information regarding air quality in the relevant Gu (district) appears in a pop-up window. At the bottom left of the map, a gauge shows the feelings of Seoul citizens so that citizen reaction according to pollution level can be verified.

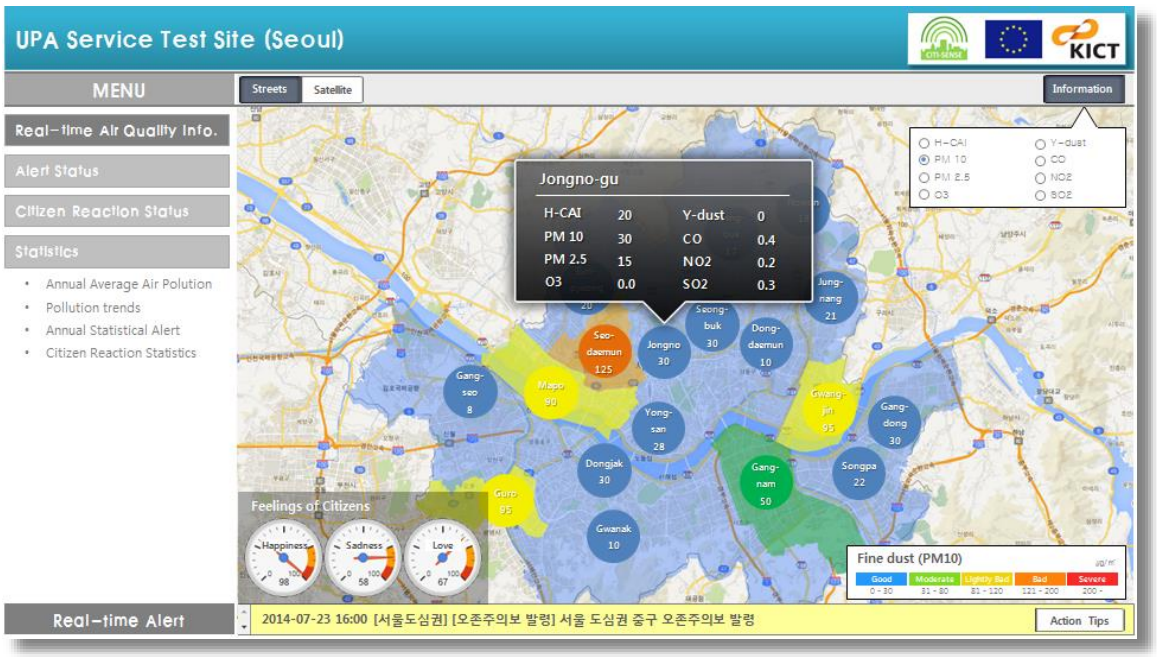


Figure 10-15 GUI of the pilot system under development

10.3 Summary and future work

The object-oriented conceptual analysis and design were performed prior to implementing the air-pollution warning service based upon the context information model. The functionality of the air-pollution warning service is designed to raise the air-pollution awareness of citizens as well as to promote their collective participation into a decision making.

The air-pollution warning services, which consists of the location-based real-time air pollution warning service, schedule-based air pollution warning service, and open data-based air pollution statistics service, are shown with the use case model, the class diagram, and the sequence diagram.

The suggested context information model will be a significant guideline to realize the web-based and mobile-based applications. In the further work, the context information model will be partially realized with the web-based application, and then the modification and improvement works will be followed to provide the practical strategy to realize the air-pollution warning services based upon the context information model.

11 GEOSS Integration

11.1 Usage and requirements

The Group on Earth Observations (GEO)[26] is coordinating efforts to build a Global Earth Observation System of Systems, or GEOSS. GEOSS will provide decision-support tools to a wide variety of users. As with the Internet, GEOSS will be a global and flexible network of content providers allowing decision makers to access an extraordinary range of information at their desk.

CITI-SENSE is relating to GEOSS in the following 4 areas:

1) Provide CITI-SENSE resources into the GEOSS infrastructure
2) Input on requirements for community portals (for GEOSS AIP-6)
3) Consider use of the GEOSS architecture and components for a citizens observatories portal
4) Support for crowd sourcing and mobile applications (for GEOSS AIP-7)

One of the objectives of the CITI-SENSE project is to make suitable data and resources available through the GEOSS Common Infrastructure (GCI). It is also being considered to what extent the GCI itself, and in particular the GEOSS Portal architecture and the Data Access Broker can be suitable for use within the CITI-SENSE platform.

The CITI-SENSE project has contributed to AIP-6 and AIP-7.

11.2 Global Earth Observing System of Systems (GEOSS)

The GEO Vision: To realize a future wherein decisions and actions, for the benefit of humankind, are informed by coordinated, comprehensive and sustained Earth observations and information.
GEO Objectives

- Improve and Coordinate Observation Systems
- Advance Broad Open Data Policies/Practices
- Foster Increased Use of EO Data and Information
- Build Capacity

[26] https://www.earthobservations.org

Figure 11-1 GEOSS Social Benefit Areas and environmental domains

Figure 11-1 shows the 9 GEOSS social benefit areas, and their relationship to 5 environmental domains. CITI-SENSE is in particular focusing on the Health and Climate social benefit areas, and the air-based environmental domain.

11.3 CITI-SENSE contributing to the GEOSS Common Infrastructure

GEOSS consists of a global and flexible network of content providers and aims at providing decision-support tools to a wide variety of users. A set of core components and functions, known as the GEOSS Common Infrastructure or GCI, has been designed and deployed to:

- Allow GEOSS resources (e.g. systems, data, services) to be easily discovered and accessed – (and the goal is that the CITI-SENSE data and services can be contributed as resources to this).
- Improve interoperability for existing and future observation systems
- Build an Open Infrastructure in accordance with the GEOSS Data Sharing Principles

An overview of the GCI architecture is shown in Figure 11-2. The initial set of GCI components included the set of resources at the bottom of the figure, a registry that contained metadata required for the discovery and access of these resources, a Clearinghouse that would harvest registered metadata catalogues to support search of the resources and a GEO Web Portal to present the GCI services to the user. The GCI also included additional registries of standards and interfaces used by the GEO community and best practices based on their on-going implementation experience.

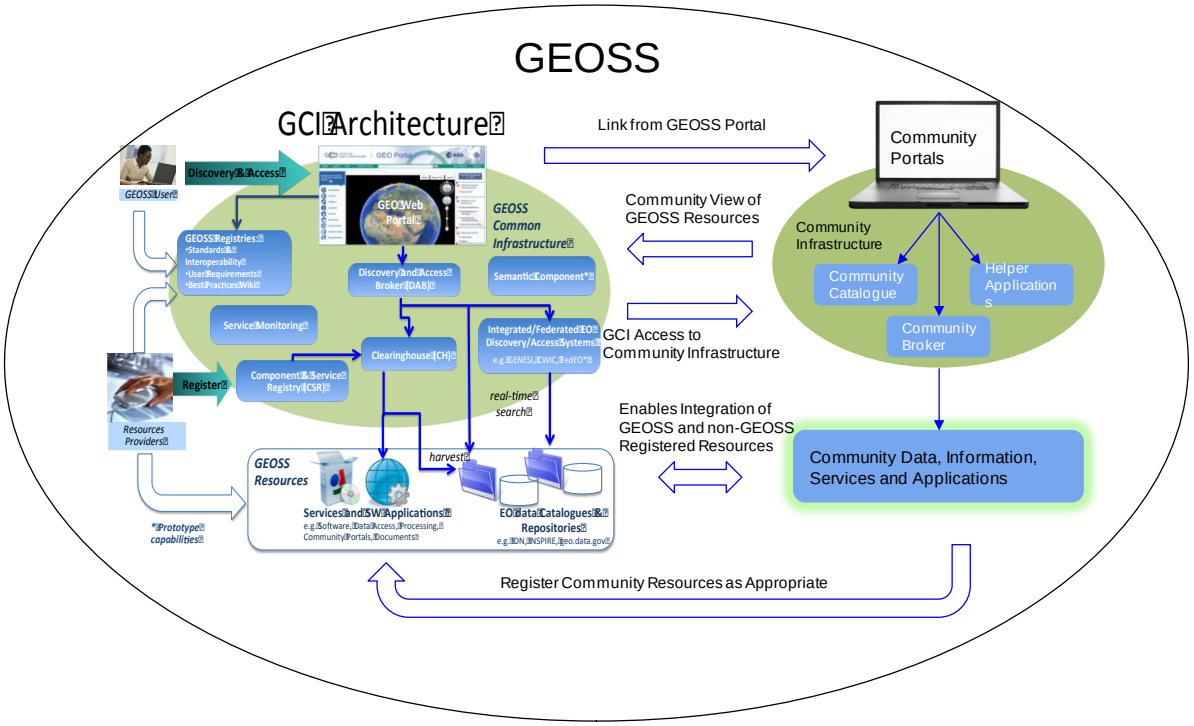


Figure 11-2 GEOSS GCI Architecture and Community Portals

The GEOSS/GEO Web Portal has evolved to interface with the Discovery and Access Broker (DAB) to provide access to the wider set of EO resources and has made advances in providing community-focused services – that CITI-SENSE might take advantage of.

The GEO DAB will provide discovery and access to a number of heterogeneous resources. The services of the GEO DAB might also be provided as support for the development of dedicated Community Portals like the Citizens Observatories from CITI-SENSE WP4, or for a common Citizens Observatory portal for all of the various EU projects that are focusing on Citizens' Observatories.

11.4 Citizen Observatories Community Objectives

A Community Portal is a community-focused portal (website) that provides a human user interface to content that may be accessed from distributed resources. The following discussion is adapted from ongoing work on a report that aims to support communities in the development of GEOSS –based community portals. CITI-SENSE has participated to this providing the requirements from a Citizens Observatories point of view.

This includes the possibility to increase the content available to a Community Portal by accessing the GEOSS Information System and in particular, the data collections that are registered in GEOSS. The recommendations will promote the standards and conventions that have been established during the development of the GCI and provide guidance and assistance to the communities on their use.

The vision of a community may be to provide its stakeholders with free, easy and open access to observational assets. Observational assets may be raw data and data products, but also metadata about observational programs, projects, and observational platforms. Stakeholders are governmental and non-governmental organisations and communities who are consumers of the data and data

products. It will also be public agencies, private companies, scientists and local (indigenous) communities who would like to have access to overviews of existing and planned observational activities in order to identify gaps in these and plan new and supplemental activities.

The providers of observational assets may or may not have their own platform for registering and storing their observational asset. In case they have their own platform, they are willing to provide access to this platform so that externals can have access to the assets. Externals can be humans or machines, and in both cases, the information providers are willing to organize the information according to a certain standard. They are not interested in sending the information to another platform.

11.5 GEOSS Architecture and CITI-SENSE

The GEOSS Information System is being developed to support distributed deployment, access and management of services on Earth Observation data and is comprised of the GEOSS Common Infrastructure and those provided by the participating GEO communities. The components are characterized as part of a service layer in a 3-tier model shown in Figure 11-3. The components directly involved in the implementation of Community Portals are in red.

- The top tier is the only one with which people deal directly. It provides the interfaces to describe and use the services offered. It includes the GEOSS Web Portal and Community Portals and Client Applications;
- The middle tier embodies all the business processes required to respond to requests issued by clients. The services in general embody everything from authentication to complex geoprocessing on sets of data from various repositories and from generation of map views to statistical charts that the client gets back at the end of the process;
- The lower tier provides read and/or write access to data, whether its geospatial data, accounting records, or catalogue entries stored in any of a dozen different types of registries.

The component types interact based upon the services as identified by bubbles on the comments in the Figure. To limit the complexity of the diagram, interactions between components are not made explicit. Services offered by the components are shown in the figure. In the figure, “ACCESS” implies the services in the Access Tier, i.e., ftp, WFS (which is the main service provided by the CITI-SENSE server), but also WMS, WCS, WFS, OPeNDAP, Order, SOS, SAS, SPS and others. Some of the ACCESS services are also provided in the Mediation Tier and maybe accessed by the Clients. For more detail on the figure including descriptions of each component and service see the GEOSS AIP Architecture[27]

[27] http://www.earthobservations.org/documents/cfp/201302_geoss_cfp_aip6_architecture.pdf

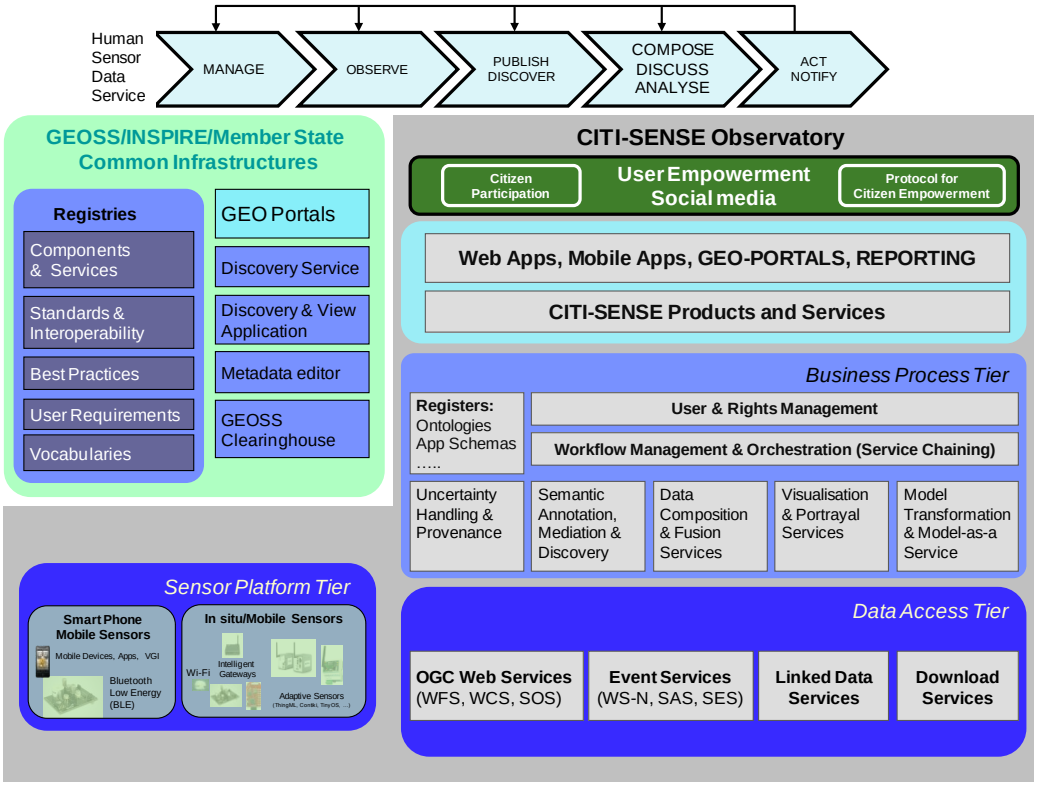


Figure 11-3 CITI-SENSE related to the GEOSS architecture

The previous figure shows how elements from the CITI-SENSE architecture can be placed in the context of the GEOSS architecture.

The Figure 11-4 elaborates more on the necessary parts of the GEOSS Common Infrastructure to support the realisation of community portals (marked in red).

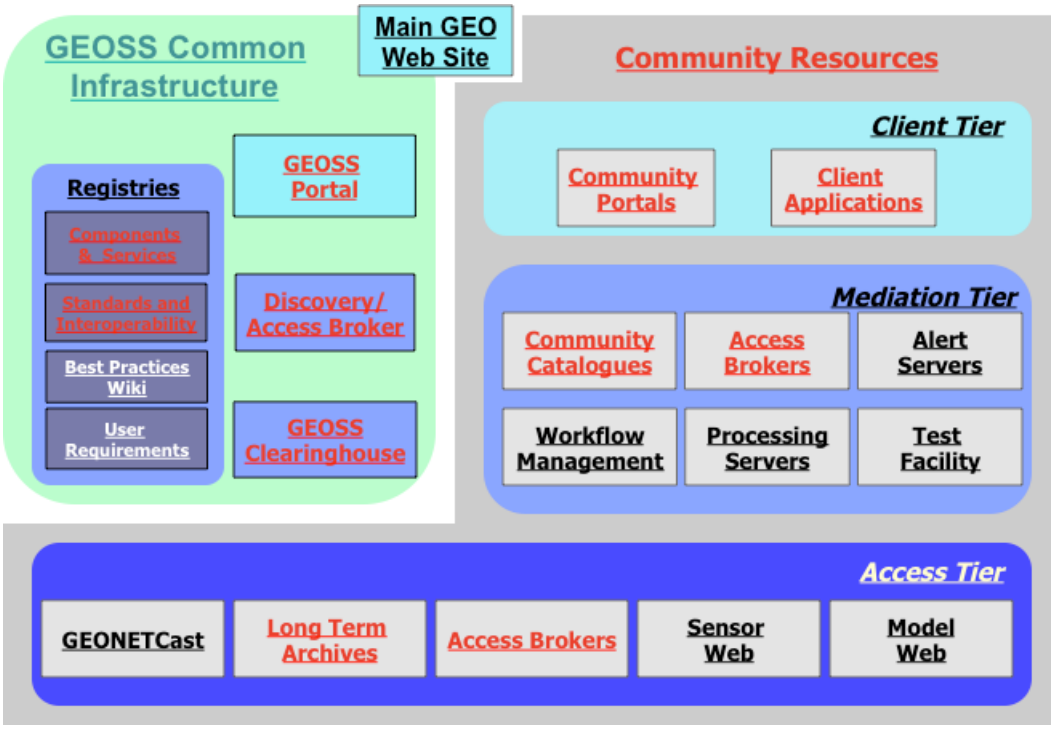


Figure 11-4 Relevant areas for community portals (marked in red)

11.6 GEOSS Use Cases

As with the Internet on which it is largely based, GEOSS will be a global and flexible network of content providers allowing decision makers access to an extraordinary range of information. *Use Cases* are reusable functions deemed most essential to using GEOSS. The GEOSS use cases provide easy, familiar processes for providers to *publish* GEOSS components, as well as for users to *discover* and *access* them across many communities of practice.

11.7 Data Sharing Principles – GEOSS Data-CORE

The GEOSS 10 Year Implementation Plan defines the GEOSS Data Sharing Principles:

> "There will be full and open exchange of data, metadata, and products shared within GEOSS, while recognizing relevant international instruments and national policies and legislation. All shared data, metadata, and products will be made available with minimum time delay and at minimum cost. All shared data, metadata, and products for use in education and research will be encouraged to be made available free of charge or at no more than the cost of reproduction."

GEOSS has committed to: 1) maximize the number of documented datasets made available on the basis of full and open access; 2) create the GEOSS Data Collection of Open Resources for Everyone (GEOSS Data-CORE), a distributed pool of documented datasets with full, open and unrestricted access at no more than the cost of reproduction and distribution; and 3) develop flexible national and international policy frameworks to ensure that a more open data environment is implemented, thus putting into practice actions for the implementation of the GEOSS Data Sharing Principles;

11.8 Strategies for Supporting Community Portals

Given Web service interfaces for both the GEOSS CSR and the Clearinghouses, it is of course possible and even desirable for a variety of client applications to be able to discover resources within them. This does introduce a variety of challenges, however, to accomplishment of a uniform and successful user discovery and access experience.

The first use case is shown below in Figure 11-5 and represents the most basic interactions between a Community Portal and GEOSS that will be addressed by CITI-SENSE initially. The CITI-SENSE portal will be registered in GEOSS to provide access to data and /or services, either directly or through a brokering service, which would be useful to the GEO community at large. The portal may search a community catalogue that contains metadata records on accessible data collections. Although this case represents the lowest level of interaction between a Community Portal and GEOSS it could be a first step toward closer integration and interaction.

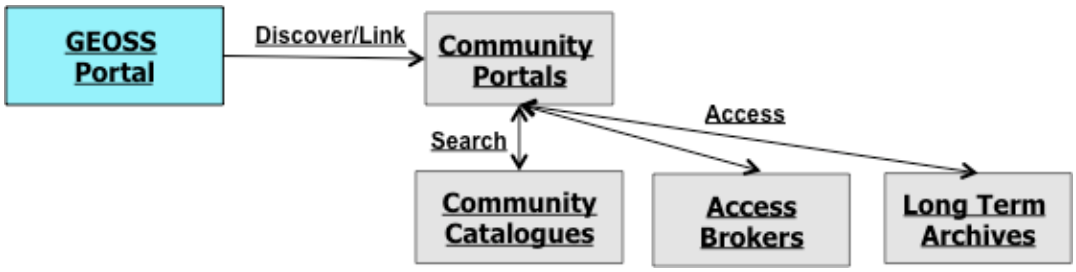


Figure 11-5 GEOSS Portal accessing available community portals

A second use case, shown in Figure 11-6, is a Community GEOSS Portal that synchronously accesses the GCI components but provides a particular thematic view of GEOSS resources. This type of portal can assist its users by highlighting those GEOSS resources that are most relevant to its community and also control how those resources are presented. These types of portal also support integration of GEOSS and non-GEOSS resources. As a result of the direct interactions with the GCI the developers of these portals are in a position to utilize and promote the GCI standards and best practices within their respective communities.

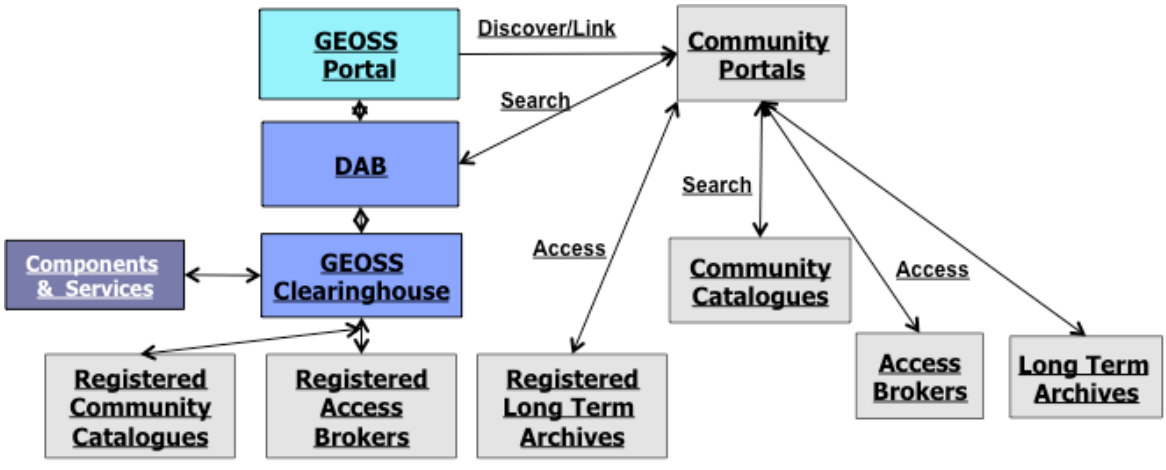


Figure 11-6 Community portal access the GEOSS Discovery and Access Broker

A more advanced use case is also a Community GEOSS Portal but one that harvests metadata from the GCI components and integrates it into its own community-focused infrastructure. It also can limit the harvested information to that which is relevant to its supported community and can support the integration of both GEOSS and non-GEOSS data and services.

11.9 Mobile Sensors, Citizen Observatories, Crowd Sourcing – in AIP-7 and AIP-8

The GEOSS Architecture Implementation Pilots (AIP) develops and deploys new process and infrastructure components for the GEOSS Common Infrastructure (GCI) and the broader GEOSS architecture. Six yearly GEOSS AIP efforts have been completed since 2007, and AIP-7 started in April 2014. The GEOSS AIP-7 work was during 2014 organised through various vertical scenario-related groups, and CITI-SENSE has been accepted as one of the project/organisations involved in this. The groups are water, oceans, land, health, agriculture and disasters management, and CITI-SENSE contributes to the health group in this context.

There is also transversal activities (the horizontals, supporting all the verticals); these includes crowdsourcing, architecture, application frameworks, tutorials, capacityBuilding, SystemDesign and Community portal. CITI-SENSE will in particular be relating to the groups on crowdsourcing, architecture, application frameworks, SystemDesign and Community portal. The CITI-SENSE project has proposed to be active in development of App/Web interfaces.

The functionality of the App/Web interfaces will be suited to use in areas with near-real time feeds and alerts on air temp, air quality, wind and other local public environmental information, coupled with local street network (directions, routing), critical facilities, basemap, utilities data, links to other data sources. Users can subscribe to timely reports or status alerts.

Data is typically gathered from various sources: Citizens (from the occasional observer to the professional surveyor) can collect a variety of environmental data properties while in the field with a location-enabled smartphones, based on pre-assigned tasks. Forms would be prepared to *collect* or *verify* (qualitative aspect) observed air quality, UV, and weather information (clouds, temperature, rainfall, etc.) and related location observations.

Sensors: With the proliferation of consumer and professional sensors, environmental data properties can be collected from fixed locations (rooftops, poles, etc.) and moving features (like buses and bikes.). Some observations should not be available for general public consumption and access can be granted to specific users only. This requires user authentication and access control. AIP-5 and AIP-6 have developed an access federation. It will be investigated how the provided applications can take advantage of this previous development.

The objective is that CITI-SENSE within the end of 2015 and into 2016 will provide a demonstration of an app/web development related to CITI-SENSE observation data as an example of Mobile Sensors, Citizen Observatories, Crowd Sourcing within AIP-8 (and in 2016 also for AIP-9), with a foundation in the CITI-SENSE architecture and platform.

12 Social Sensors

12.1 Usage and Requirements

A Social Sensor is a software agent that provides observations about an environment via communicating to other agents. The role is to understand citizens' experience about their environment. By analysing a stream of messages from citizen's an agent can detect citizen's intention and expectation. Then, by using ontologies, data can be linked semantically to other open data, geo-spatial information and other physical/social sensors data, a social sensor provides good integrated indicators about various aspects of environment.

Social media like Twitter, YouTube etc. have demonstrated the ability to provides information in real-time more efficiently than traditional other media and that could not be covered by other physical sensors. Processing these Social media with Data and Text mining technologies allow to transform them into powerful Social Sensors for detecting real-time issues or analyse sentiments and opinions.

A useful report on the baseline and previous work on social sensors has been provided through the D1.21 deliverable of the WeSenseIt project[28] , see Figure 12-1.

Figure 12-1 Social Media for Social Sensors from WeSenseIt Project

Social Sensors should be able to output a timestamp, a location and a measure for an event. Timestamp is known by the Social Media stream but the location and measure should be evaluated by the agent implementing the Social Sensor. Moreover the output should be semantically linked to other output so it does require a special handling for disambiguation between different entities specified by ontology. Because input of a Social Sensor is a stream of events and that a sensor is specialized to detect only partial information for a particular location (for example positive or negative sentiment about air pollution in Seoul or London), Social Sensors need to be aggregated by a workflow, enabling to provide aggregated measures for different scales like positive or negative Environment in South Korea or England. It requires that inputs and outputs of Social Sensor to be standardized.

That is the reason why languages like SenML (SensorML) or specifications like SWE (Sensor Web Enablement) are used to specify Social Sensors. By integrating several Social Sensor with other sensors that can be plugged together allow to build flexible Platforms offering various services like Collecting/Gathering, Discovering/Providing, Monitoring/Alerting or Planning, Visualization Services.

[28] http://www.wesenseit.com/web/guest/deliverables1

As a pilot project, sensor for measuring "Yellow Sand' pollution in South Korea has been developed, detecting keywords and clusters such as "PM10", "car Emission", "Black rain" etc. A dashboard provides an interface for visualizing the results as Gauge, HeatMap or Line Chart, helping monitoring the current status of "Yellow Sand" pollution in various regions of South Korea.

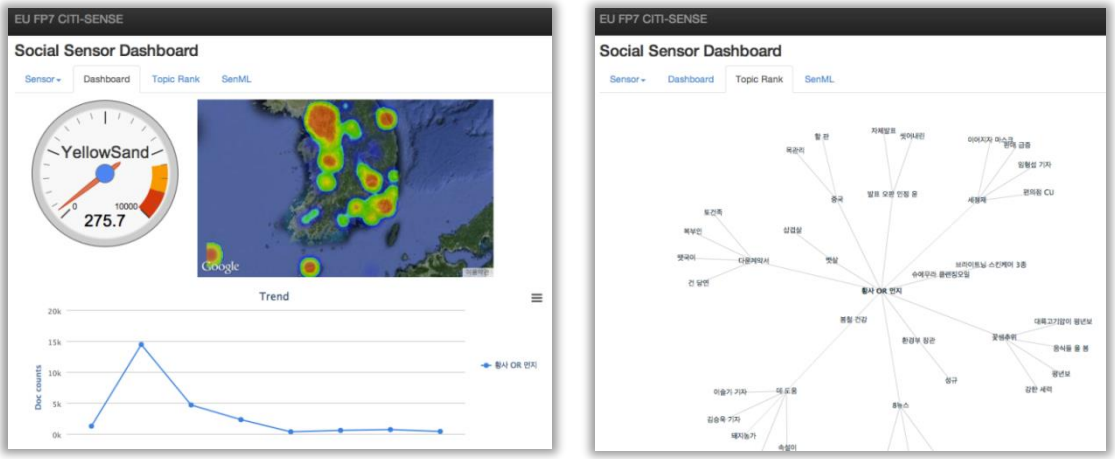


Figure 12-2 Social Sensors Dashboard

This section also describes about *PLUQI (Personalized and Localized and Localized Urban Quality Index)* and its requirements as one of the use cases using social sensors. We can define PLUQI as:

> A customizable index model and mobile/web application that can represent and visualize the level of well-being and sustainability for given cities based on individual preferences.

This index model will represent:

- *Daily life satisfaction*: weather, transportation, community, living density, shopping space, entertainment venues …
- *Safety and Healthcare level:* number of police, doctors, hospitals, sports clubs, fire stations, crimes per capita …
- *Mental health and safety:* suicide statistics, mental hospitals, social sentiments, average working hours …
- *Financial satisfaction:* prices, incomes, housing, savings, debt, insurance, pension …
- *Level of opportunity*: jobs, unemployment, education, re-education, economic dynamics …
- *Environmental needs and efficiency:* green space, water and electricity consumption, air quality …
- *Political satisfaction*: civic engagement, NGO, voter turnout …
- *Cultural satisfaction*: number of theatres, museums, art centres …

A composite index of urban quality derived from all sub-indices will be represented for each city. It is impossible to represent a composite index based on a single model to cover all aspects of different applications and individual preferences, and this is the reason for introducing personalization and model customization capability for PUQI. An individual user or organization can apply their preferences and rate the importance of each sub-index to re-analyse a PUQI composite index and determine city ranking.

To collect all the data needed for getting indices, PLUQI uses various data sources such as physical sensors, social sensors and open data. The following 7 cities and 9 regions have been defined for social sensors: Seoul, Busan, Daegu, Incheon, Gwangju, Daejeon, Ulsan, Gyeonggi-do, Gangwon-do, Chungcheongbuk-do, Chungcheongnam-do, Jeollabuk-do, Jeollanam-do, Gyeongsangbuk-do, Gyeongsangnam-do, Jeju-do.

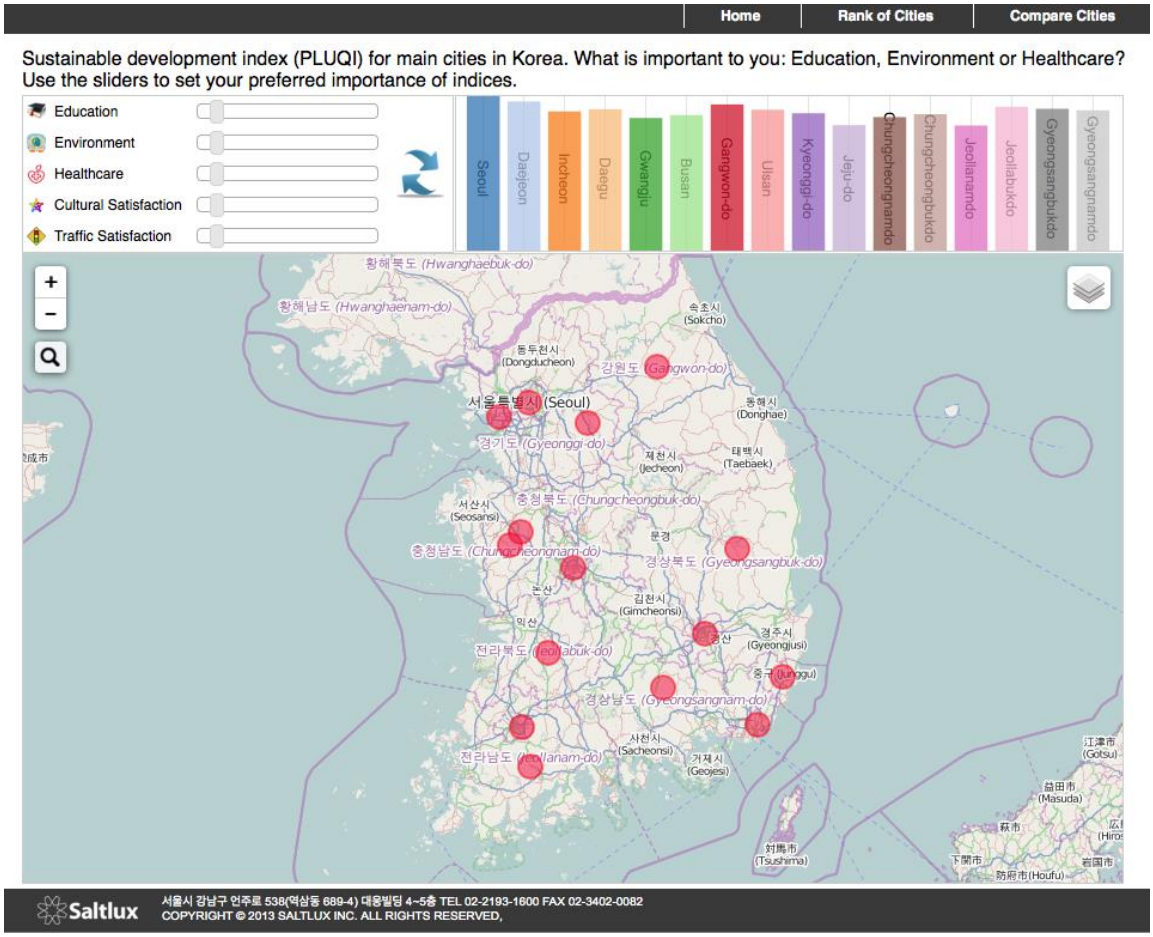


Figure 12-3 PLUQI Service Web Application

PLUQI web services shows composite index and the sub-indices for each cities and regions on map and bar charts, and users can adjust their preferences using slide (Figure 12-3 12-3).

12.2 Logical Service Interfaces and Information Model

As described in section 12.1, PLUQI uses various sensor data from CITI-SENSE platform and open data from DaPaaS[29] platform. And the data sets combined based on PLUQI ontology schema (will be described in section 13.2). It means that the data gathered by social sensors are mapped to the ontology so that it will be combined with other data sets to calculate sub-indices.

[29] http://project.dapaas.eu/

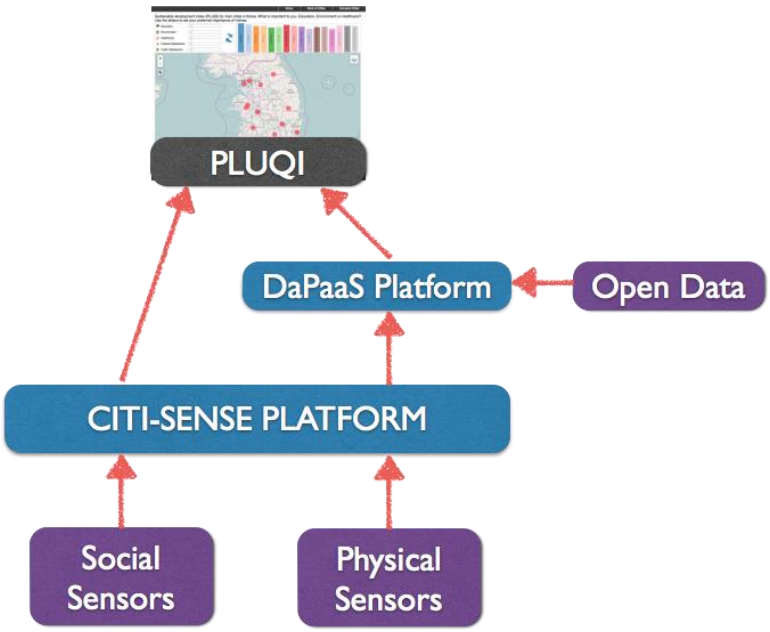


Figure 12-4 PLUQI Use Case System Architecture

Social sensor data is pushed to the CITI-SENSE platform regularly, and it will be mapped to the CITI-SENSE core ontology to be combined with the PLUQI schema, so that social sensor data is used for calculating indices.

The OpenGIS® Sensor Model Language Encoding Standard (SensorML) specifies models and XML encodings that provide a framework for defining geometric, dynamic, and observational characteristics of sensors and sensor systems[30]. Within SensorML, all processes and components are encoded as application schema of the feature model in the Geographic Markup Language (GML) Version 3.1.1. This is one of the OGC Sensor Web Enablement (SWE) (OGC_SWE) suite of standards[31] The primary focus of SensorML is to define processes and processing components associated with the measurement and post-measurement transformation of observations.

The purpose of SensorML is to:

- Provide descriptions of sensors and sensor systems for inventory management.
- Provide sensor and process information in support of resource and observation discovery.
- Support the processing and analysis of the sensor observations.
- Support the geolocation of observed values (measured data).
- Provide performance characteristics (e.g., accuracy, threshold, etc.).
- Provide an explicit description of the process by which an observation was obtained (i.e., it's lineage).
- Provide an executable process chain for deriving new data products on demand (i.e., derivable observation).

[30] http://www.opengeospatial.org/standards/sensorml
[31] http://schemas.opengis.net/sensorML/

- Archive fundamental properties and assumptions regarding sensor systems.

The primary focus of SensorML is the description of sensors, sensor systems and the processing components associated with the measurement and post-measurement transformation of sensor observations.

In SensorML, all components are modeled as processes, that can be connected and participate equally within a process chain or system. This includes components normally viewed as hardware, including transducers, actuators, and processors (which are viewed as process components) and sensors and platforms (which are modeled as systems). All components are modeled as processes that take input, and which, through the application of an algorithm defined by a method and parameter values, generate output. The inputs, outputs, and parameters are all defined using SWE Common data types. Process metadata includes identifiers, classifiers, constraints (time, legal, and security), capabilities, characteristics, contacts, and references, in addition to inputs, outputs, parameters, and system location.

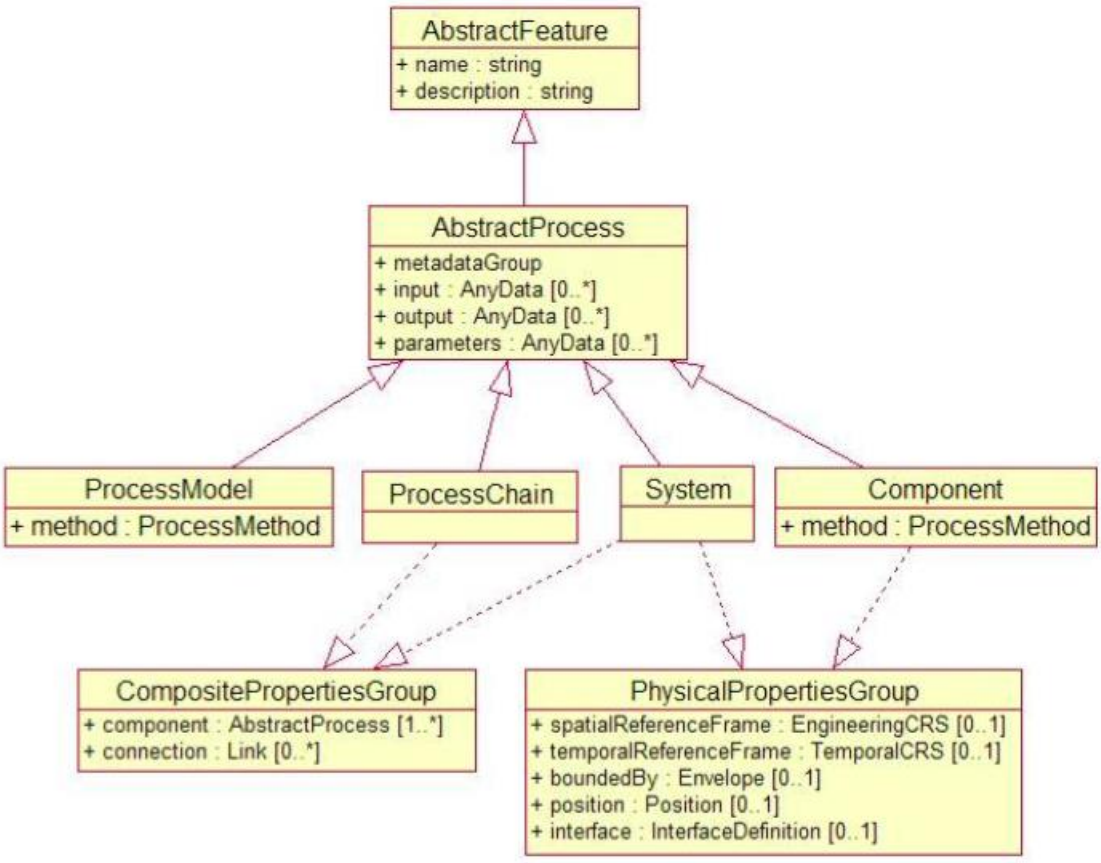


Figure 12-5 SensorML Conceptual Model for Processes

Social Sensors Interface is using SensorML by specifiying a <<ProcessChain>> specifying one or several <<input>> like ”Tweets” and several <<output>> specifying values like ”Count”, ”Time”, ”Location”. In the <<ProcessChain >> several <<component>> can be aggerated in flow. The Figure 12-6 illustrate the specification for a Social Sensor Interface.

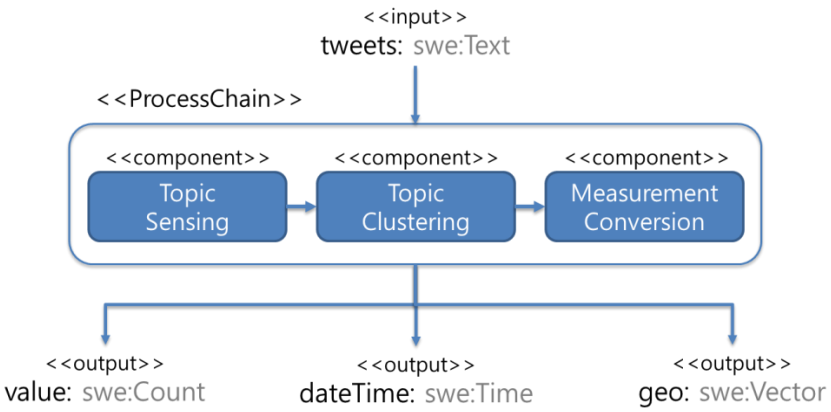


Figure 12-6 Social Sensors Interface

12.3 Implementation Technologies

This section describes the actual technology and implementation elements related to these services. As Social Sensor should process Natural Language expressed in free texts from Social Media, UIMA (Unstructured Information Management) is a good framework for implementing Social Sensor. UIMA is used for the ability to scale-out and the support of aggregating output of our components.

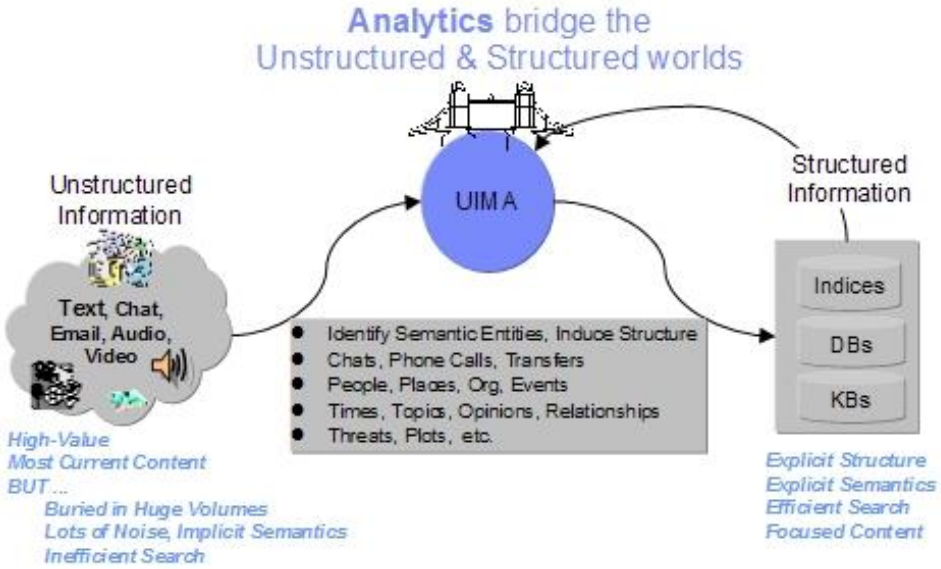


Figure 12-7 UIMA Overview

Core components needed for implementing a Social Sensor are the Crawling Infrastructure for collecting Tweets stream in real-time and the NER (Named Entities Recognition) module that can use various algorithms. One challenge related to Natural Processing Language (NLP) is to support multiple languages for the entire NLP stack including lexical, syntactic and semantic components. As we intend to support languages for all CITI-SENSE cities and partners, even if we start with support of only English and England, it is important to have easy trainable components independently of languages so changing only the input that is a training corpus of annotations for a specific language. Rules based

and dictionaries based methods cannot be applied because it can simply not scaled and are too expensive to construct. So machine learning methods are chosen to meet this requirement so we have implemented our using Latent structural SVM (Support Vector Machine) because it can be trained fast at large scale and have high precision/recall performances. We did train it for Korean and plan to apply it to other languages.

Latent structural SVM:

$$\min_{\mathbf{w}} \frac{1}{2} \| \mathbf{w} \|^2 + \frac{C}{n} \sum_{i}^{n} \left(\max_{(\mathbf{y},\mathbf{h}) \in \mathbf{Y} \times \mathbf{H}} \left(\Delta(\mathbf{y}_i, \mathbf{y}, \mathbf{h}) + \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}, \mathbf{h}) \right) \right) - \frac{C}{n} \sum_{i}^{n} \left(\max_{\mathbf{h} \in \mathbf{H}} \mathbf{w} \cdot \Phi(\mathbf{x}_i, \mathbf{y}_i, \mathbf{h}) \right).$$

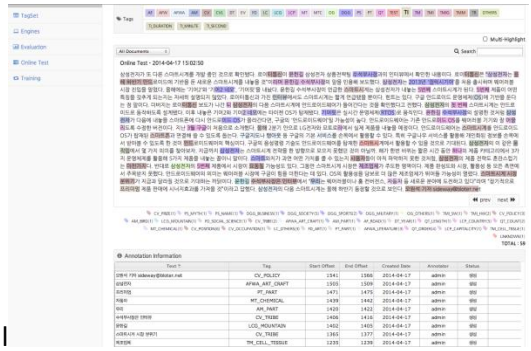


Figure 12-8 NER Component

For crawling Social Media, an infrastructure of 100 servers has been used for crawling Korean tweets. Processing more than 1.5 million tweets per days requires a high computing power for crawling, storing, analysing, indexing and monitoring the all infra. The figure below shows the main indicators of the collecting infrastructure.

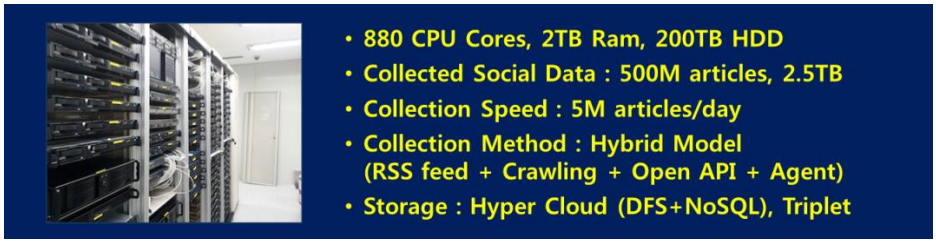


Figure 12-9 Crawling Infrastructure

For PLUQI, five social sensors are defined, three for environment related sensors, two for life quality related ones as below:

- Environment
  - Yellow Sand & PM10
  - Temperature
  - Humidity

- Life Quality
  - Hapiness
  - Anxiety

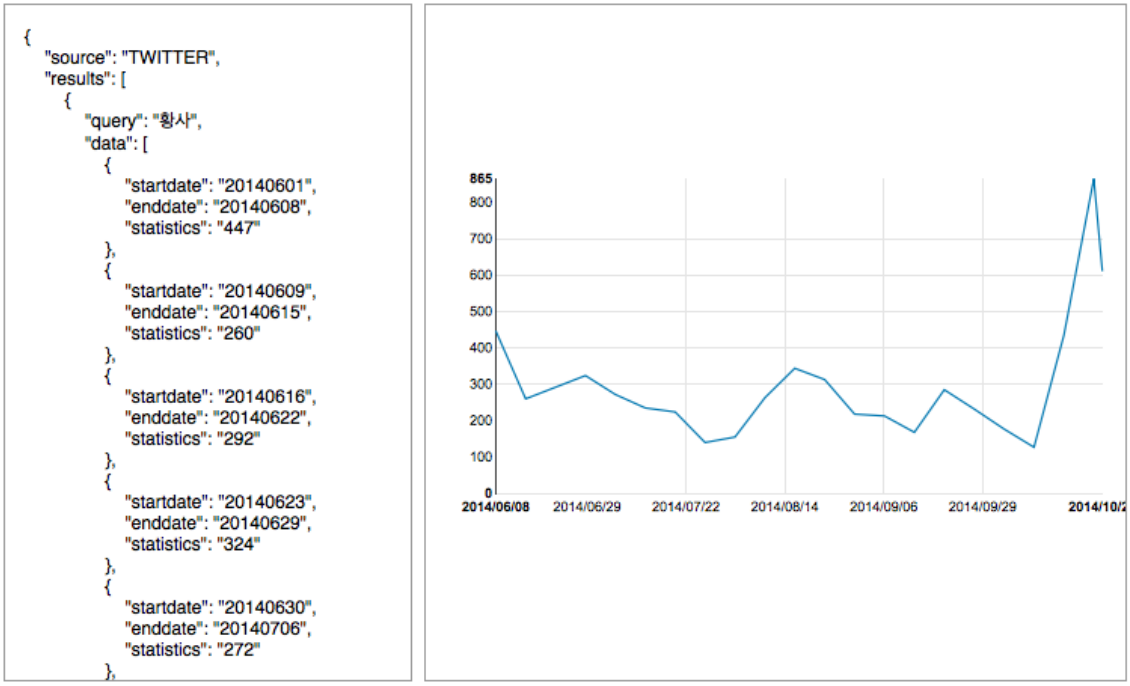
Basically each of social sensors measures intension or interest of people for selected duration (day, week, month or year), on the topic based on the search formula, for an example:

Formula for meauring Yello Sand & PM10

Korean:
황사OR 미세먼지OR (미세먼지AND 농도OR (미세먼지AND (예방OR 호흡기OR 꽃가루)

English:
"yellow sand" OR "fine dust" OR ("fine dust" AND concentration) OR ("fine dust" AND (prevention OR "respiratory organ" OR "pollen")

As the example shows, it includes the terms related *Yello Sand*, and also for the ones people mentions when they tweet about the *Yello Sand* or *PM10*. So we can get the frequencies of tweets (Figure 12-10) and its related topics, and these are the information to be pushed to the CITI-SENSE platform via WFS-T service.



```
{
  "source": "TWITTER",
  "results": [
    {
      "query": "황사",
      "data": [
        {
          "startdate": "20140601",
          "enddate": "20140608",
          "statistics": "447"
        },
        {
          "startdate": "20140609",
          "enddate": "20140615",
          "statistics": "260"
        },
        {
          "startdate": "20140616",
          "enddate": "20140622",
          "statistics": "292"
        },
        {
          "startdate": "20140623",
          "enddate": "20140629",
          "statistics": "324"
        },
        {
          "startdate": "20140630",
          "enddate": "20140706",
          "statistics": "272"
        },
```


Figure 12-10 Social Sensor Data

Figure 12-11 Social Topics

Social sensor is basically a service to push sensing data to the WFS server, and it pushes the data daily so users can get statistical (tweet counts) and semantic(topic keywords) information from the WFS service.

12.4 Future Work

For the first step, social sensors are used for practical use case, but there are still remaining tasks to be done:

- domain coverage
    - For now it is concentrating on environment domain, so we should expand the domain which cannot be detected (We added life quality social sensors, but they are experimental ones).
- location coverage
    - Just around 10% of tweet messages has geo location information. So to get the locational data, we need to extract it from the tweet messages with natural language processing, and using administrative division information so that social sensors can detect more of tweets as a sensible data.
- accuracy
    - Basically, social sensors are based on the formula which is dependent on experiences of the domain experts. So to gather accurate and enough amount of tweets, the formula should be advanced.
    - And many of the tweet messages have noise which blocks the accuracy of it, so we need to use filter to get rid of the noise.

13 Linked (Open) Data Services

13.1 Usage and Requirements

This section describes the usage context and use cases for Linked (Open) Data services for CITI-SENSE.

Linked Open Data[32] 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. In order for this envisioned strategy to become a reality, it is necessary to advance the publication of existing environmental data, most of which is owned by public bodies. How Linked Open Data can be applied generally to spatial data resource and specifically to public open data portals, GEOSS Data-CORE, GMES, INSPIRE and voluntary data (OpenStreetMap, GEP-WIKI, etc.), and how it can impact the economic and sustainability progress in European Environment research are currently being addressed in other parallel projects like SmartOpenData and DaPaaS.

The SmartOpenData project[33] (2013-2015) is addressing these questions by defining mechanisms for acquiring, adapting and using Open Data with a particular focus on biodiversity and environment protection in rural and European protected areas and its National Parks. CITI-SENSE partner SINTEF is a member of the SmartOpenData project.

The DaPaaS project[34] (2013 – 2015) is providing a platform for the semi-automatic support of making existing data sets available as linked open data. CITI-SENSE partners SINTEF and Saltlux are members of the DaPaaS project.

The CITI-SENSE project might take advantage of these developments in order to publish relevant CITI-SENSE data in form of Linked Open Data. The project will also be able to contribute to the ongoing discussions in the GEO and GEOSS communities about the use and support for Linked Open Data.

The vision of Linked Open Data for environmental data is that environmental and related geospatial data can be more readily available and re-usable, better linked with data without direct geospatial reference so different distributed data sources could be easily combined together. An objective is making INSPIRE/GMES/GEOSS infrastructure better available for citizens, as well as for public and private organizations. On the one hand, Europe and the EU invest hundreds of millions of Euros in building the INSPIRE infrastructure. On the other hand, it is public and private organizations, as well as citizens use for their applications Google maps. National and regional SDIs offer information which is not available on Google, but this potential is not used. One of the main goals of extended usage of Linked Open Data is making European Spatial Data easily re-usable not only by GIS experts but also by various organizations and individuals at a larger scale.

Using linked data for spatial data means identifying possibilities for the establishment of semantic connections between INSPIRE/GMES/GEOSS and Linked Open Data spatial related content in order to generate added value. This may be achieved by making existing “INSPIRE based” relevant spatial data sets, services and appropriate metadata available through a new Linked Data structure. In addition,

[32] Berners-Lee, T., Hendler, J. and Lassila, O. (2001). The Semantic Web. Scientific American Magazine, 2001.
[33] http://www.smartopendata.eu/
[34] http://project.dapaas.eu/

the proposed infrastructure can provide automatic search engines that will crawl additional available geospatial resources (OGC and RDF structures) across the deep and surface web. The main motivation to utilise the potential of Linked Data is to enrich the INSPIRE spatial content to enable improved related services to be offered and to increase the number, performance and functionality of applications. In many cases querying data in INSPIRE (GEOSS) based data infrastructure (driven mainly by relational databases) is time consuming and often it is not sufficient and understandable for common Web users. In large databases such queries can take minutes or hours. In the cases of distributed databases such a query is almost impossible or very complicated.

The main focus on Linked Open Data within CITI-SENSE will be is on utilising existing and emerging results from other projects, for how to convert the following sources to RDF/Linked data:

- Data in the WFS server from Snowflake
- the Questionnaire from U-Hopper
- and the Social sensor data

Linked Data is an approach that aims at contributing to the development of the Web of data. As opposed to developing data in silos and duplicating data of various topics for their integration, Linked Data is oriented towards connecting data of different sources, subjects, and repositories via the Web. As a result, this eliminates duplication of data and, when data that are linked to others are updated, the updates are then available to all at the same time. Linked Data is a large scale integration of data over the Web that allows reasoning and making inferences taking in consideration multiple aspects. As a consequence, Linked Data implies publishing data on the Web in a way that it is machine-readable. This means that data shall follow a common model and format, i.e. RDF. Additionally, to understand what data mean and to reason with, it is essential that the meaning of data is explicitly defined as well as machine readable. This is possible by the way of ontology and languages such as OWL.

One of the benefits of Linked Data and especially Linked Open Data is that data is sharable, extensible, and easily re-usable. It supports multilingual functionality for data and user services, such as the labelling of concepts identified by a language-agnostic URIs. These characteristics are inherent in the Linked Data standards and are supported by the use of Web-friendly identifiers for data and concepts.

Like the linking that takes place today between Web documents, Linked Data allows anyone to contribute unique expertise in a form that can be reused and recombined with the expertise of others. The use of identifiers allows diverse descriptions to refer to the same thing.

Christian Bizer and Tom Heath in a tutorial about “how to publish Linked Data”[35] explain how adding RDF links enable Linked Data browsers and crawlers to navigate between data sources and to discover additional data. Some inter-linking can be done manually, some by mining techniques for identifying equivalences generally expressed by the predicate owl:sameAs like Berlin defined in DBPedia or Geonames data.

[35] http://wifo5-03.informatik.uni-mannheim.de/bizer/pub/LinkedDataTutorial/#links

Linked Open Data can be applied not only in domain of sensors but virtually for any domain. Figure 13-1 shows how data from various areas and domains are interlinked through linked data representations in the Internet.

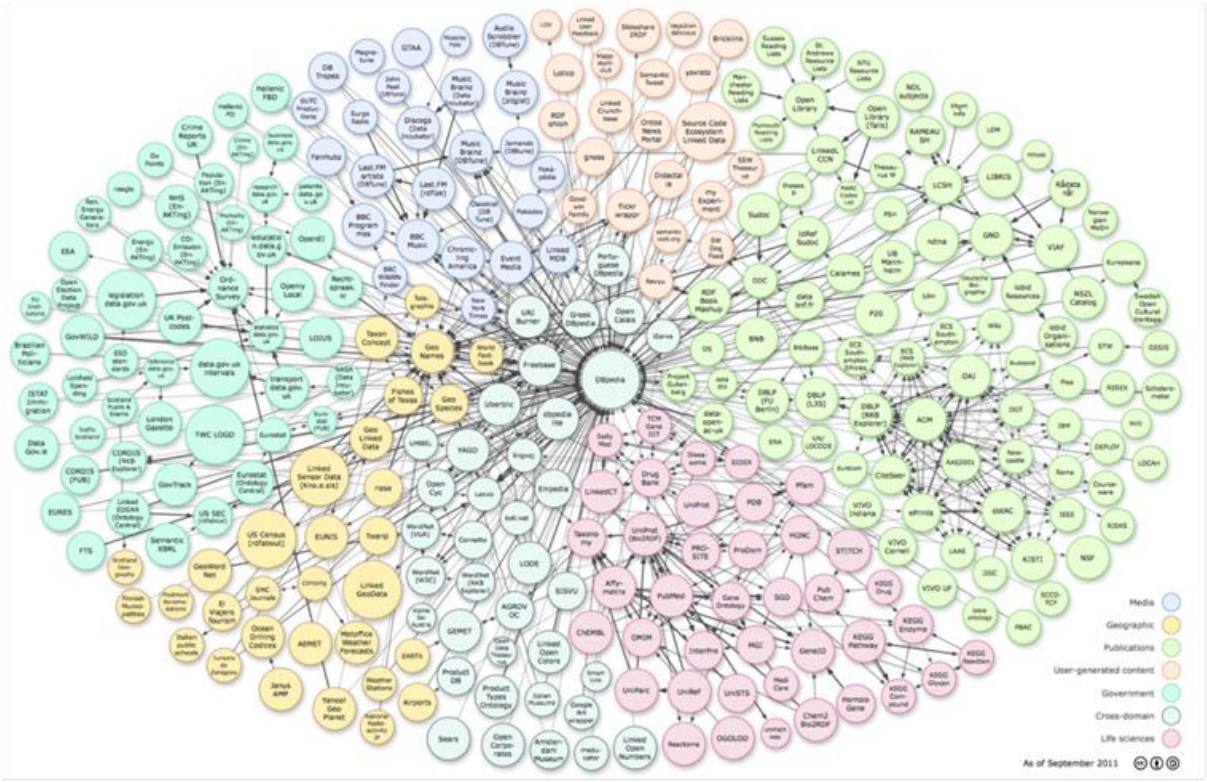


Figure 13-1 Linked Open Data Cloud

The SmartOpenData Project is motivated by the fact that Linked Open Data is becoming a source of high visibility for environmental data and the focus is on how Linked Open Data can be applied generally to spatial data resource and specifically to public open data portals.

Sheth et al[36]. defined semantics of sensor web within space, time, and theme scopes. There have been different approaches to provide semantic models for each of these attributes independently or in relation to sensors used. Some of the common ontologies are the SIMILE location ontology, the DAML location ontology for spatial attributes, OWL time ontology for time and common ontologies and vocabularies such as CyC, DBpedia for thematic data

The location of each sensor is crucial to provide a context to the observation it performs. Different resources are in place to help formalising location and space as CityGML, GeoNames, GeoSPARQL, Mobile ontology, World Geodetic System 1984, NGEO and Core Location Vocabulary. Each of them are devoted to represent an exact location with GPS coordinates or geometry based on polygons and points for instance, while others provide geographical names to locate an object / person.

For sensor domain, The W3C Semantic Sensor Network (SSN) Incubator Group has developed a formal OWL DL ontology[37] for modelling sensor devices (and their capabilities), systems and

[36]Semantic Sensor Web: https://www.scss.tcd.ie/Rob.Brennan/SemanticSensorWeb.pdf
[37] Semantic Sensor Network Ontology: http://www.w3.org/2005/Incubator/ssn/ssnx/ssn
Name Space: http://purl.oclc.org/NET/ssnx/ssn

processes. This ontology describes sensors and observations, and related concepts. It does not describe domain concepts, time, locations, etc. These are intended to be included from other ontologies via OWL imports.

It is based in part on the **ISO 19156 "Observations and Measurements"** conceptual model. The ontology is based around concepts of **systems, processes, and observations**. It supports the description of the **physical and processing structure of sensors**. Sensors are not constrained to physical sensing devices: rather a sensor is anything that can estimate or calculate the value of a phenomenon, so a device or computational process or combination could play the role of a sensor. The representation of a sensor in the ontology links together what it measures (the domain phenomena), the physical sensor (the device) and its functions and processing (the models).

The SSN ontology is organized, conceptually but not physically, into ten modules as shown in Figure 13-2.

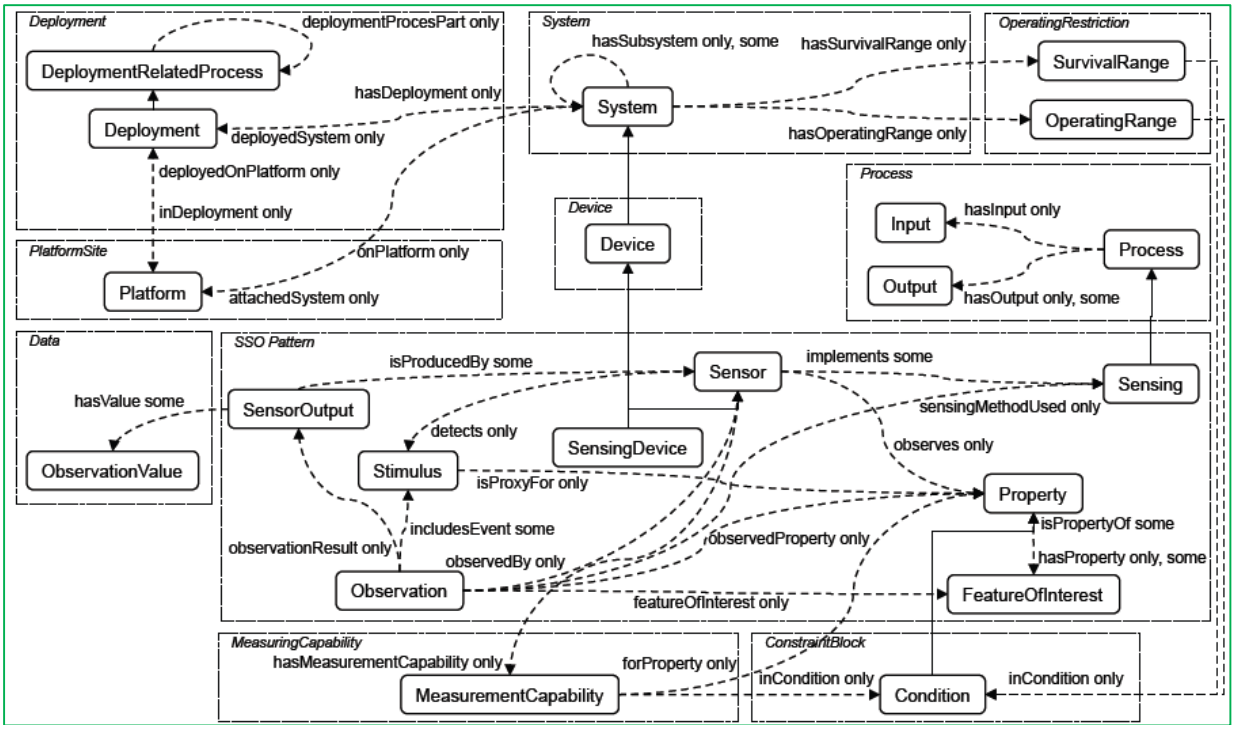


Figure 13-2 SSN ontology, key concepts and relations, split by conceptual modules

As mentioned in previous chapter, PLUQI uses not only *Social Sensor* data but also open data to get composite index and sub-indices, and the data is combined based on the PLUQI ontology schema. This process is supported by DaPaaS project[38] (2013–2015), providing a platform for the semi-automatic support of making existing data sets available as linked open data. CITI-SENSE partners SINTEF and Saltlux are members of the DaPaaS project.

From DaPaaS platform, PLUQI uses virous data sets from datasources for Korea, as below:

- Seoul Metapolitan Government - http://data.seoul.go.kr

[38] http://project.dapaas.eu/

- Korean Open Information Portal - http://wonmun.open.go.kr
- Ministry of Security and Public Administration - http://gwanbo.korea.go.kr
- Korean Statistical Information Service- http://kosis.kr
- Apartment Management Info System - http://www.k-apt.go.kr
- Public Data Portal - https://www.data.go.kr
- Statistics Korea - http://sgis.kostat.go.kr
- DB Store- http://www.dbstore.or.kr

The data sets collected from the data sources are categorized and mapped to the PLUQI ontology schema. In the Figure 13-3 13-3, it shows the data flow comparison between two approaches in perspective of using linked data service:

a) With linked data service to combine social sensor data and open data sets based on PLUQI ontology, PLUQI web application can use composite index and also its data.
b) Without linked data service, PLUQI web application can use only the data itself.

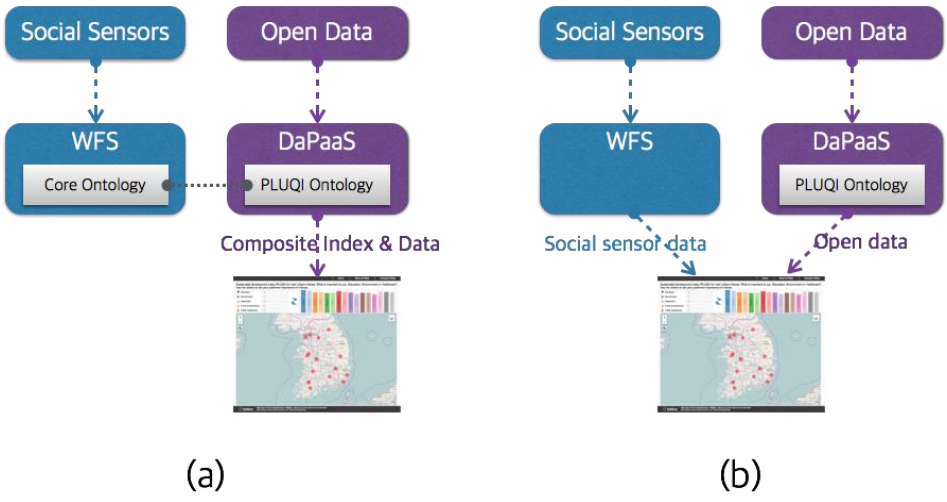


Figure 13-3 Comparison of the data flow with or without the linked data service

DaPaaS provides linked data technologies so that social sensor data can be converted into RDF format based on PLUQI ontology schema.

13.2 Logical Service Interfaces and Information Model

When considering the development and use of a semantic representation framework three general requirements need to be considered:

- Interoperability requirements: it is often desirable (or even necessary) to receive or provide information to external systems, therefore the representation framework should be clearly specified and, as far as in possible, based upon the use of standards. This enables the sharing of knowledge with external systems, which use the same standards or which can translate information represented in such standards to/from their internal representation.
- User requirements: the framework must be able to represent all the knowledge to satisfy the requirements imposed by the users of the system in which is framework is applied.

- Domain requirements: It is necessary for a system to be able to accurately represent the environment with which it must interact. This information may not be directly access by the user or exported to external systems, however it is necessary to represent the information, as it is used by the internal processes of the system to determine what information the user (or an external system) receive. For example, the provenance of knowledge captured from multifarious sources may be represented in order to determine the knowledge's validity: knowledge only being exposed to the user or external systems when it is validated to a satisfactory degree.

The RDF and Linked Data model provide the most flexibility in terms of representing, interlinking and querying data. Data stored as RDF on the platform will be accessible via arbitrary complex and expressive SPARQL queries, may be queried in a federated manner, or may be interlinked with other semantic data on the platform (depending on data provider preferences). SPARQL endpoints are Web based services for query data and return results in standard RDF serialization form.

For managing the data in CITI-SENSE, the lifecycle of Linked Data must be fully supported, from data extraction, enrichment, interlinking, to maintenance as the following figure taken from the LOD2[39] project (LOD2 Project, 2012) illustrates this in Figure 13-4.

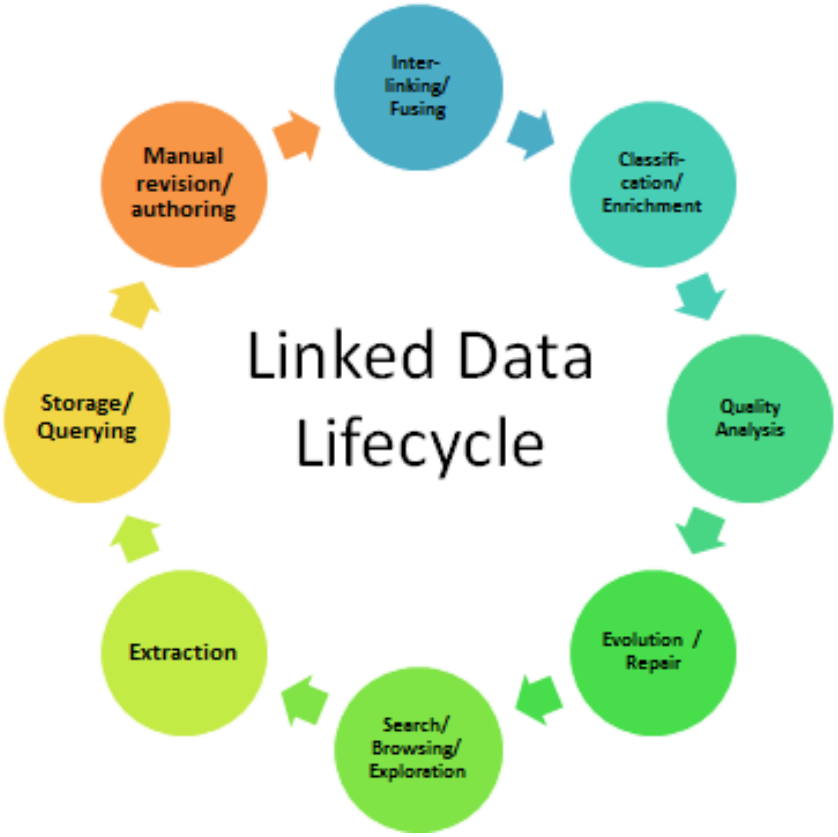


Figure 13-4 LOD2 lifecycle for Linked Data Management, (c) LOD2 project

Services that must be provided for data management are such as:

[39] http://lod2.eu/

- Import from sources and transform in RDF: Sources can be from sensors, relation databases (RDB), Comma-Separated-Values (CSV) or from Natural Language form as in Social Media.
- Extract RDF Data from NLP: NLP support like Apache Stanbol or Open NLP for Named Entities Recognition and linking to Ontologies instances (like DBPedia Spotlight) are needed,
- Browse or Visualize Data: Tool for browsing or visualizing (like CubeViz or SemMAP for Spatial data browsing) also are needed,
- A Centralized Knowledge repository: A RDF Storing the Knowledge for storing and querying data that can be accessed externally through SPARQL Endpoint is also needed
- A Restful framework for allowing services to be accessible each other independently of languages,
- Other services for validation, quality checking, statistical views etc. are also required.

CITI-SENSE core ontology network is composed of several components as in Figure 13-5. And PLUQI ontology follows the structure to have *Domain*, *Sensor Observations* and *Location* as *Quality Index*, *Value* and *Location*, respectively. So PLUQI ontology easily can be combined with the CITI-SENSE core ontology, it means that if the other 3rd party applications or services follow the structure, they could exchange the information between the systems.

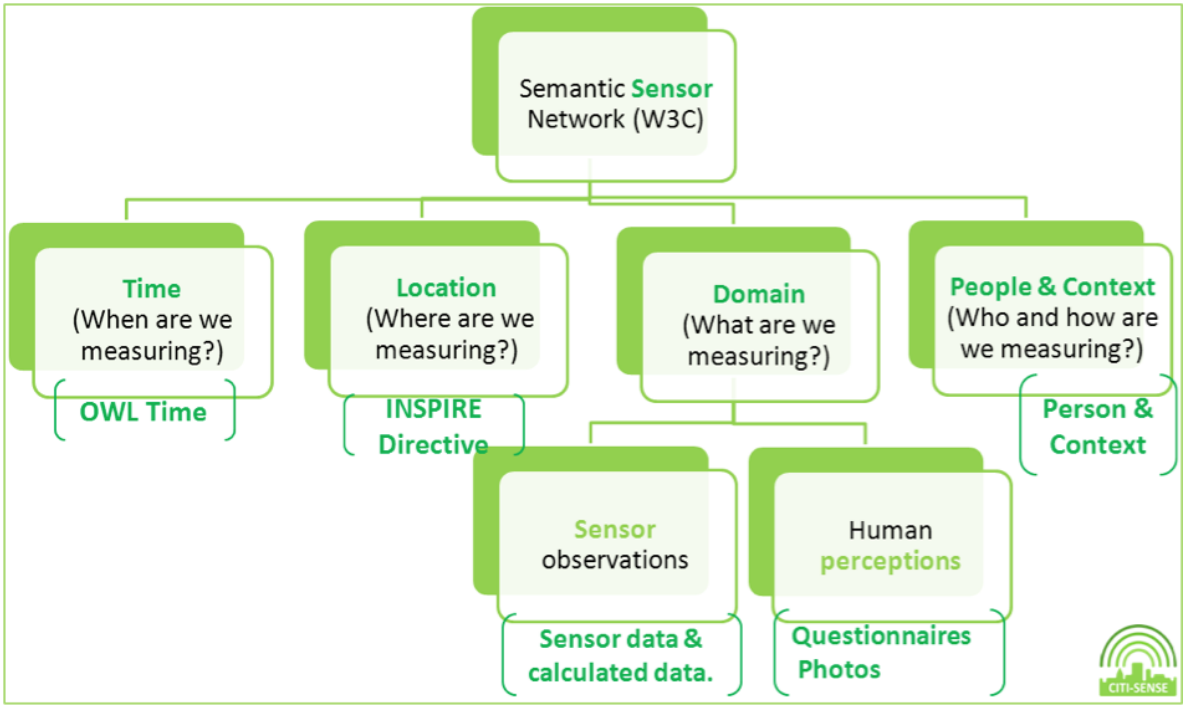


Figure 13-5 CITI-SENSE Core Ontology Network

The PLUQI ontology schema is designed as Figure 13-6. It can represent sub-indices and its values with location and date-time information. And it uses other standard ontology model or linked data like *WGS84* and *Korean Administrative Divisions* ontology[40].

[40] http://lod.seul.go.kr

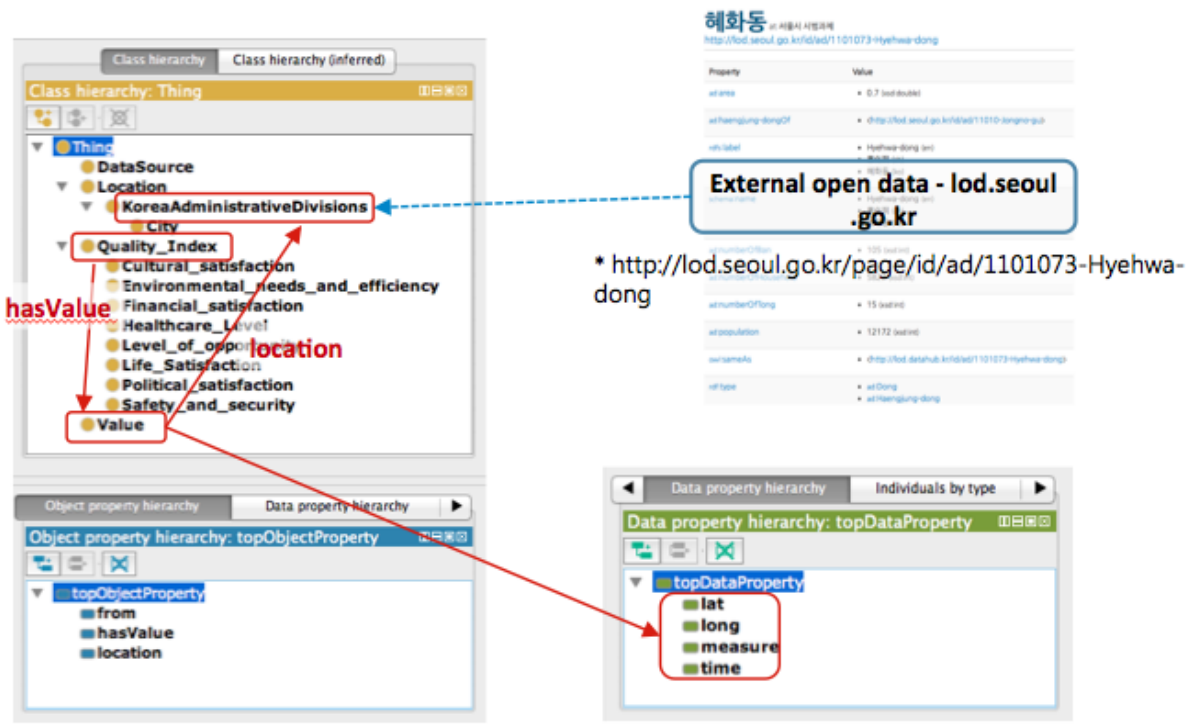


Figure 13-6 PLUQI ontology schema

Following the schema, we can describe the information of *highschool in Seoul* as a data of *Level of opportunity*, in Turtle format:

```
:v1 rdf:type :Value ,
      owl:NamedIndividual ;
      :measure 151 ;
      :time "2011-00-00T00:00:00Z"^^xsd:dateTime ;
      rdfs:comment "type:total"^^xsd:string ;
      :hasValue <http://project.dapaas.eu/schema/2014/5/pluqi/2011-2013_highschools> ;
      :location :Seoul .

:number_of_coed_high_schools rdf:type :Level_of_opportunity ,
      owl:NamedIndividual ;
      :hasValue :v1 .
```

13.3 Implementation Technologies

For import from sources and transform in RDF many tools exist in order to convert data from their original form to RDF and there is an interesting list on the W3C wiki[41].

For data in relational databases the db2triples library is a software tool for extracting data from relational databases and loading data into an RDF triple store. It implements R2RML and Direct Mapping standards defined by W3C's RDB2RDF. The implementation has been validated by the RDB2RDF working group and it has served as a bench test during the different phases of definition and validation of the RDB2RDF recommendations.

In a nutshell, db2triples accepts as input RDBMS connection parameters and a R2RML transformation document (and no mapping in case of *Direct Mapping* mode). The output is the RDF dump of the database data based on the transformation mode used. The tool manages the memory efficiently, which in turn enables it to process large amounts of data. Db2triples is an open source software implemented in Java, published under the LGPL license. The source code and building instructions are available at Github[42].

cvs2rdf4lod[43] is a quick and easy way to produce an RDF encoding of data available in Comma-Separated-Values (CSV). In its advanced form, csv2rdf4lod is a custom reasoner tailored for some heavy-duty data integration.

One core component for any semantic platform is the storage part often deeply related to reasoning capabilities. A semantic repository must offer a query mechanism for data retrieval and data maybe accessed remotely. Semantic data are often store in form of triples so the repository is often also called a triplestore. A triplestore is a purpose-built database for the storage and retrieval of triples, a triple being a data entity composed of subject-predicate-object.

Sesame[44] is an open source Java framework for storage and querying of RDF data. The framework is fully extensible and configurable with respect to storage mechanisms, inferencers, RDF file formats, query result formats and query languages. Sesame offers a JBDC-like user API, streamlined system APIs and a RESTful HTTP interface supporting the SPARQL Protocol for RDF.

Out of the box, Sesame supports SPARQL and SeRQL querying, a memory-based and a disk-based RDF store and RDF Schema inferencers. It also supports most popular RDF file formats and query result formats. Various extensions are available or are being worked at elsewhere.

Here follows a high-level overview of Sesame's components:

[41] http://www.w3.org/wiki/ConverterToRdf
[42] https://github.com/antidot/db2triples/
[43] http://logd.tw.rpi.edu/technology/csv2rdf4lod
[44] http://www.openrdf.org/

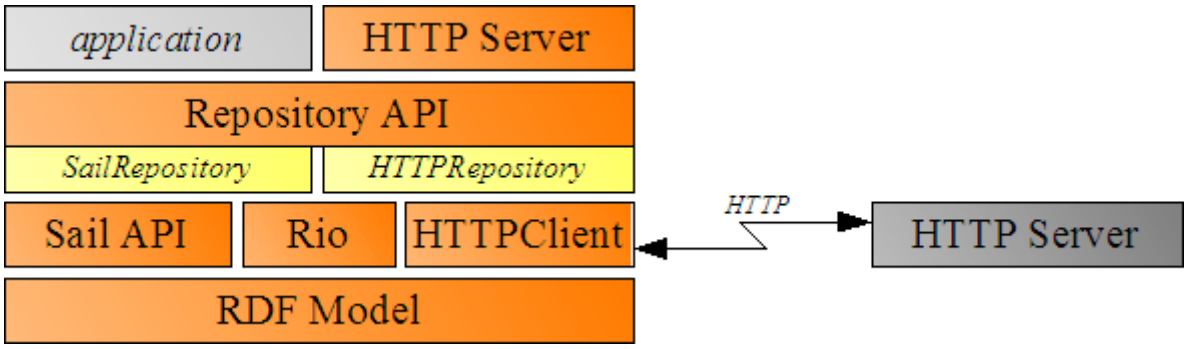


Figure 13-7 Sesame Architecture

All the way at the bottom of the diagram in Figure 13-7 is the **RDF Model**, the foundation of the Sesame framework. Being an RDF-oriented framework, all parts of Sesame are to some extent dependent on this RDF model, which defines interfaces and implementation for all basic RDF entities: URI, blank node, literal and statement.

Rio, which stands for "RDF I/O", consists of a set of parsers and writers for various RDF file formats.

The Storage And Inference Layer (SAIL) API is a low level System API for RDF stores and inferencers. Its purpose is to abstract from the storage and inference details, allowing various types of storage and inference to be used (mainly of interest for triplestore developers).

The **Repository API** is a higher level API that offers a large number of developer-oriented methods for handling RDF data. It offers various methods for uploading data files, querying, and extracting and manipulating data. There are several implementations of this API, the ones shown in this figure are the *SailRepository* and the *HTTPRepository*. The former translates calls to a SAIL implementation of choice, the latter offers transparent client-server communication with a Sesame server over HTTP.

The top-most component in the diagram is the **HTTP Server**. The HTTP Server consists of a number of Java Servlets that implement a protocol for accessing Sesame repositories over HTTP. The details of this protocol can be found in Sesame's system documentation, but most people can simply use a client library to handle the communication. The HTTPClient that is used by the HTTPRepository is one such library.

Similar to Sesame, Jena[45] is another open source Semantic Web framework for Java. It provides an API to extract data from and write to RDF graphs. The graphs are represented as an abstract "model". A model can be sourced with data from files, databases, URLs or a combination of these. A Model can also be queried through SPARQL and updated through SPARUL.

Unlike Sesame, Jena provides support for OWL (Web Ontology Language). The framework has various internal reasoners and the Pellet reasoner[46] (an open source Java OWL-DL reasoner) can be set up to work in Jena.

Jena architecture overview:

- The **Graph layer** is the base layer in Jena. It is very granular and is a very minimal implementation of the RDF specification. It permits a wide range of implementations, such as in-memory or persistent triple stores.

[45] http://jena.apache.org/
[46] http://clarkparsia.com/pellet

- The **Model layer** extends the core functionality in the Graph layer in such a way, that by allowing developers to work with objects of type "Resource" or "Property" or "Statement", instead of "Node" or "Triple".
- The **OntModel Layer** supports inference capabilities, that is, the ability to work with triples that are implied, in addition to the triples that have been explicitly defined.

As the principle of LOD is to link data or interlink datasets semantically, components are needed for providing this service. RDF Refine[47] can be used to reconcile and link data (against SPARQL endpoints, or search for related RDF datasets) or export data as RDF format.

The Silk Link Discovery Framework[48] is a tool supporting data publishers in accomplishing the task of discovering relationships between data items within different Linked Data sources. Using the declarative Silk - Link Specification Language (Silk-LSL), developers can specify which types of RDF links should be discovered between data sources as well as which conditions the data items must fulfil in order to be interlinked. These link conditions may combine various similarity metrics and can take the graph around a data item into account, which is addressed using an RDF path language. Silk accesses the data sources that should be interlinked via the SPARQL protocol and can thus be used against local as well as remote SPARQL endpoints.

The main features of the Silk Framework are:

- Flexible, declarative language for specifying linkage rules
- Support of RDF link generation in RDF
- Employment in distributed environments (by accessing local and remote SPARQL endpoints)
- Usable in situations where terms from different vocabularies are mixed and where no consistent RDFS or OWL schemata exist
- Scalability and high performance through efficient data handling:
    - Reduction of network load by caching and reusing of SPARQL result sets
    - Multi-threaded computation of the data item comparisons
    - Optional blocking directive, which allows users to reduce the number of comparisons on cost of recall, if necessary.
- Active Learning of expressive linkage rules using genetic programming[49]
- Silk Workbench - a web application which guides the user through the process of creating link specifications, results evaluation, and linkage rules learning.

For Social Sensor, Mendes et al examined the process of generating RDF triples from tweets, seen in Figure 13-8, based on the metadata from the tweets (entity mentions, hashtags, and URLs). The system called Twarql[50] has the following architecture:

[47] http://refine.deri.ie/
[48] http://www4.wiwiss.fu-berlin.de/bizer/silk/
[49] http://dws.informatik.uni-mannheim.de/fileadmin/lehrstuehle/ki/pub/IseleBizer-ActiveLearningOfExpressiveLinkageRules-JWS2013.pdf
[50] http://wiki.knoesis.org/index.php/Twarql

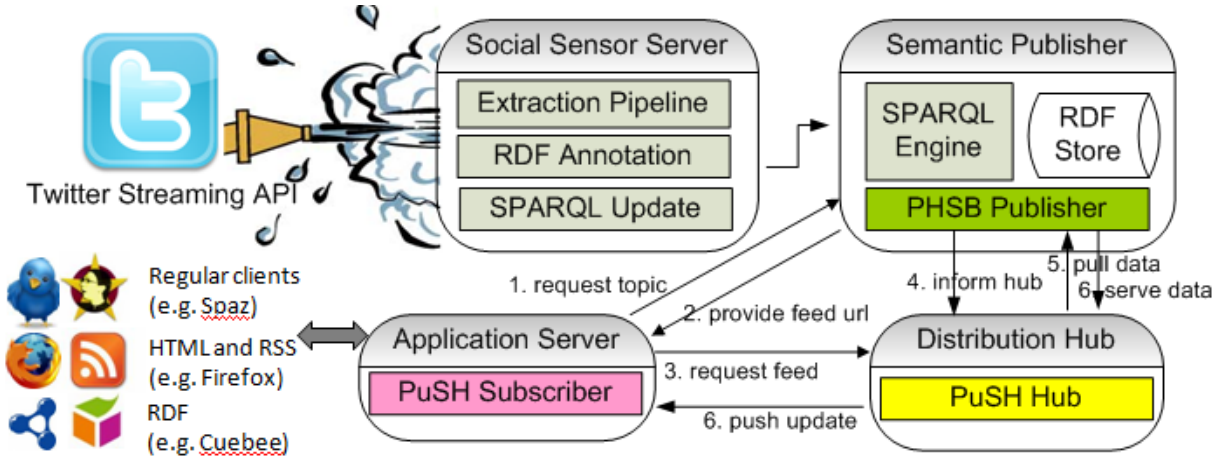


Figure 13-8 Generating RDF triples from tweets

Especially for Social Sensors, recognizing Named Entities like locations is the first step but to get it in Linked Data format, the second step is to link it to the instance defined in the ontology, but the process is difficult because of ambiguities of language. For example mentioning "Berlin", is it referring to the city in Germany or in USA?

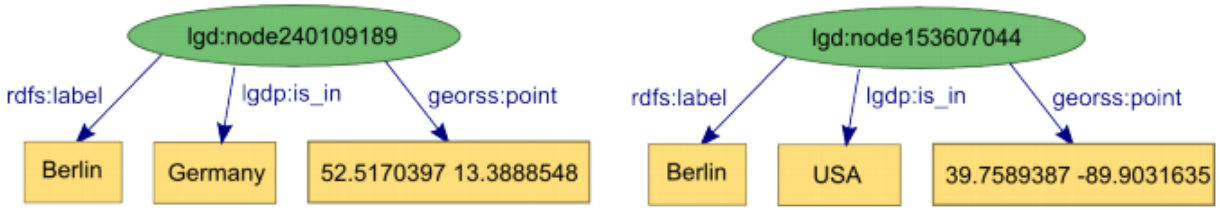


Figure 13-9 Example of an entity pair in a geographical data set.

Machine Learning technologies are used for making this automatically from training sets.

Algorithm 1 Pseudocode of the ActiveGenLink algorithm.

```
P ← generate initial population
U ← generate unlabeled pool
R ← empty set of reference links

while(|R| < maximum links to be labeled) {
  u ← query strategy selects link candidate from U
  r ← ask user to label u
  R ← R ∪ r
  P ← learn linkage rules from R based on population P
}

return best linkage rule from P
```

Figure 13-10 Machine Learning for linking learning

The plan is to introduce support for linked data representation of the available CITI-SENSE data from the WFS server for D7.6 – due in end of September 2016. The structure for this will be further related to the ontology approaches described earlier in D7.2.

Many of the open data sets are written in CSV format, and Social Sensor data from WFS service provides the data with JSON format. To convert those data in RDF format for PLUQI use case, we use Grafter[51], which is one of the linked data technology converting tabular data sets into RDF linked data format. Grafter provides DSL (Domain Specific Language) to define transformation pipelines so to convert social sensor data into PLUQI ontology, we follows the step as below:

1. Getting social sensor data form WFS service
2. Convert JSON data to CSV format
3. Use Grafter to convert into linked data format, to be mapped to the PLUQI ontology

And also for open data, it should be converted to linked data. For example, Figure 13-11 shows an example of open data which is written in spread sheet format (Excel file), and Grafter convert this data as a tabular data (Figure 13-12), and into linked data format (Figure 13-13).

	A	B	C	D	E	F	G	H	I	J	K	L
1	administrat	type	2013									
2			# of highsc	# of man hi	# of girl's h	# of coed h	# of teache	# of wamar	# of office v	# of female	# of new st	# of female
3	Seoul	total	318	73	88	157	23,190	11,063	2,162	702	110,478	53,541
4		national	3	-	-	3	142	94	27	7	651	424
5		public	115	11	11	93	8,891	5,291	859	346	39,108	16,731
6		private	200	62	77	61	14,157	5,678	1,276	349	70,719	36,386
7	Busan	total	144	40	36	68	8,940	4,083	1,053	390	39,762	18,728
8		national	4	2	-	2	303	132	131	67	963	219
9		public	62	16	11	35	4,008	2,362	436	184	17,685	7,572
10		private	78	22	25	31	4,629	1,589	486	139	21,114	10,937
11	Daegu	total	92	23	18	51	6,966	2,951	627	225	34,175	16,047
12		national	1	-	-	1	71	29	3	1	359	120
13		public	42	7	4	31	3,262	1,850	300	145	15,110	6,959
14		private	49	16	14	19	3,633	1,072	324	79	18,706	8,968
15	Incheon	total	122	39	35	48	7,798	4,283	712	286	35,297	16,984
16		national	1	1	-	-	47	17	20	6	123	-
17		public	89	27	23	39	5,770	3,560	499	210	25,663	12,218
18		private	32	11	12	9	1,981	706	193	70	9,511	4,766
19	Gwangju	total	67	16	19	32	4,281	1,784	372	107	22,883	11,174
20		national	1	-	-	1	67	47	4	-	307	155
21		public	24	3	4	17	1,568	899	122	55	7,724	3,411
22		private	42	13	15	14	2,646	838	246	52	14,852	7,608

Figure 13-11 An example of open data - Highschools in Korea

```
Seoul,total,number of coed high schools,151.0,2011
Busan,total,number of coed high schools,67.0,2011
Daegu,total,number of coed high schools,50.0,2011
Incheon,total,number of coed high schools,45.0,2011
```

Figure 13-12 CSV format converted from Excel file

[51] http://grafter.org

<http://project.dapaas.eu/pluqi/data/number-of-coed-high-schools> a <http://project.dapaas.eu/pluqi/schema/Level_of_opportunity> ,

<http://www.w3.org/2002/07/owl#NamedIndividual> ;

<http://project.dapaas.eu/pluqi/schema/hasValue> <http://project.dapaas.eu/pluqi/data/number-of-coed-high-schools-seoul-2011> .

<http://project.dapaas.eu/pluqi/data/number-of-coed-high-schools-seoul-2011> a <http://www.w3.org/2002/07/owl#NamedIndividual> ,

<http://project.dapaas.eu/pluqi/data/Value> ;

<http://www.w3.org/2000/01/rdf-schema#label> "Number of coed high schools in Seoul in 2011."@en ;

<http://project.dapaas.eu/pluqi/schema/measure> "151"^^<http://www.w3.org/2001/XMLSchema#int> ;

<http://project.dapaas.eu/pluqi/schema/location> <http://project.dapaas.eu/pluqi/data/Seoul> ;

<http://project.dapaas.eu/pluqi/schema/hasValue> <http://project.dapaas.eu/pluqi/data/2011-2013_highschool> ;

<http://project.dapaas.eu/pluqi/schema/time> "2011-01-01T00:00:00.000Z"^^<http://www.w3.org/2001/XMLSchema#dateTime> ;

<http://www.w3.org/2000/01/rdf-schema#comment> "total"@en .

<http://project.dapaas.eu/pluqi/data/2011-2013_highschool> a <http://project.dapaas.eu/pluqi/schema/DataSource> ,

<http://www.w3.org/2002/07/owl#NamedIndividual> .

Figure 13-13 Linked data format converted from open data

13.4 CAQI and CAI

In CITI-SENSE a common air quality index (CAQI) has been defined. In Korea there is, however, another index developed, called CAI – for Comprehensive Air-quality Index - representing the information easy to understand by human and to be related with daily life. Several countries defines their own Air Quality Indexes, specially South Korea uses the Comprehensive Air Quality Index (CAI) based on the health risks of air pollution as shown in Figure 13-14. Note that this index is defined slightly differently from the CITI-SENSE, but used in this situation because of the local usage of this index in Korea.

One of the next step for PLUQI could be combining air quality data and social sensing data with AIQ, mapping with PLUQI composite index and providing the data so that users can get and understand the air quality data meaningful. And for this, PLUQI ontology schema and linked data technologies could support to combine between data and AIQ to be represented.

CAI	Description	Health Implications
0–50	Good	A level that will not impact patients suffering from diseases related to air pollution.
51–100	Moderate	A level that may have a meager impact on patients in case of chronic exposure.
101–150	Unhealthy for sensitive groups	A level that may have harmful impacts on patients and members of sensitive groups.
151–250	Unhealthy	A level that may have harmful impacts on patients and members of sensitive groups (children, aged or weak people), and also cause the general public unpleasant feelings.
251–350	Very unhealthy	A level that may have a serious impact on patients and members of sensitive groups in case of acute exposure.
351–500	Hazardous	A level that may need to take emergency measures for patients and members of sensitive groups and have harmful impacts on the general public.

Figure 13-14 Korean CAI (Comprehensive Air-quality Index)

14 Potential Supporting Services

14.1 Communication and Composition services

Further need for support for communication and composition services will be analysed for the final phase of the CITI-SENSE project based on the final needs of the case studies. This might be relevant for composition of data products and services. This will be related to potential services on data fusion and service discovery, composition and execution. This can support the reuse, integration, and enhancements of components for discovery, composition, mediation, and execution of environmental data and services.

14.1.1 Event Services

The platform can be enhanced to support event services which allow clients to receive notifications about changes to data held within the CITI-SENSE platform.

WS-Notification will be considered to allow clients to register an interest in particular sets of content (topics in WS-Notification terminology). The platform will then push notification messages to subscribers in response to changes in the CITI-SENSE data according to their interests. This type of notification can been used alongside OGC services to keep decision support clients up to date with a changing situation. WS-Notification can make use of reliable messaging services to ensure that all events reach their intended recipient. This makes it suitable for applications where it is important that the client is aware of all relevant changes.

For lightweight clients a GeoRSS feed will be considered. This is a simple protocol which allows clients to poll the service to see recent changes. This allows users to browse recent changes to a service and provides metadata to allow the client to filter for changes in their area or topic of interest. Unlike WS-Notification GeoRSS does not provide for guaranteed message delivery.

Further need for support for communication and event/notification services will be analysed for the final phase of the CITI-SENSE project based on the final needs of the case studies. This might be relevant in order to generate alerts and notifications if observed data is outside of stated boundaries. This can both be useful in the context of quality control, as well as for alerting users.

The platform will potentially be able to support event services which allow clients to receive notifications for instance about unusual data values held within the CITI-SENSE platform.
WS-Notification will be considered to allow clients to register an interest in particular sets of content (topics in WS-Notification terminology). The platform will then push notification messages to subscribers in response to changes in the CITI-SENSE data according to their interests. This type of notification can be used alongside OGC services to keep decision support clients up to date with a changing situation. WS-Notification can make use of reliable messaging services to ensure that all events reach their intended recipient. This makes it suitable for applications where it is important that the client is aware of all relevant changes.
For lightweight clients a GeoRSS feed will be considered. This is a simple protocol which allows clients to poll the service to see recent changes. This allows users to browse recent changes to a service and provides metadata to allow the client to filter for changes in their area or topic of interest. Unlike WS-Notification GeoRSS does not provide for guaranteed message delivery.

14.2 Communication in SOA and Event driven architectures

Event-Driven Architecture (EDA) and Service-Oriented Architecture (SOA) are both methods for the architecture and component coupling within a system. How the components are integrated and interact between each other is described with these concepts.

The CITI-SENSE architecture has been created based on service-oriented architecture principles. It is also an objective to allow for the support of an event-driven architecture, which can be viewed as an extension of a service-oriented architecture.

An event based system (EDA) basically communicates using event notifications. Thereby the events are sent to event handlers over a direct connection or through a broker using a publish/subscribe mechanism. Each handler decides which events to receive by subscribing to topics of interest.

In contrast, a service based system (SOA) uses service requests. Different to an EDA system, the request is sent, a certain service is executed and replies with results. The basic principle is that the consumer invokes a certain service provider. For EDA the provider sends events to several consumers, without receiving any result and without even knowing if there are consumers or what they are doing.

14.3 Notification with Publish/Subscribe

The publish/subscribe (pub/sub) or notification service component can be viewed as part of communication services in order to support an event driven architecture.

Pub/sub, as a communication and messaging pattern allows system components to disseminate information to other components without knowing them, by communicating through an Event Notification manager. This means a sender (publisher) can publish any message without specifying the receiver. These published messages are classified into categories, for example related to certain topics or matching patterns, and a receiver (subscriber) can dynamically register interest for one or more of these categories and further on asynchronously receive category related events.

Both the subscriber and publisher do not have any information of the other part, meaning it is possible that messages are published but not received by any component as nobody has subscribed for that topic. One characteristic of a pub/sub system is the event categorization, which can be either subject-based or content-based. With a subject based approach all events are related to a subject/topic, while for a content-based approach the events are matched against a set of patterns. The system architecture can differ between a server-client and a peer-to-peer structure. Further on, pub/sub components can differ in: matching algorithm (defines when events are delivered to a subscriber), multicast algorithm, reliability and security.

14.4 Complex Event Processing (CEP)

An event is can be defined to be: "anything that happens, or is contemplated as happening" [08]. Events can be related to other events, for example by cause or membership and have properties such as a time stamp. Furthermore, a complex event is created by a set of events, thus representing all aggregated actions, meaning a complex event happens, when all its underlying low-level events happened. In terms of citizens' observatories, a low-level event is a single measurement send to the server, for example the UV radiation. This event has properties, such as a time stamp or a location.

Based on these properties the event can be related to other events, like the weather forecast for this location and time. A complex event, for example an alarm to avoid being in a certain area without sun protection, can be generated out of several related low-level events. This complex event can provide additional benefit to the user compared to raw unrelated measurements.

The described process of generating complex events by detecting and aggregating related low-level events is part of CEP, which is a technology including several techniques for processing and thereby making use of events and their relationships within an event driven system.

According to the Event Processing Glossary - Version 2.0 [08], CEP is defined as "computing that performs operations on complex events, including reading, creating, transforming, abstracting, or discarding them". The main techniques for generating complex events are filtering, aggregating and matching low-level events. In this context matching events means to detect complex patterns of many single events. CEP support can be added to a sensor data framework in order to make use of sensor data by aggregating it, calculating indexes or detecting patterns, as mentioned before.

The Sensor Data Event Processing (SDEP) Architecture described next illustrates how the service-oriented architecture of CITI-SENSE also can be supporting event processing.

14.5 Sensor Data Event Processing Architecture

The Sensor Data Event Processing Architecture has been created in order to show how complex event processing can be combined with a service-oriented architecture such as the CITI-SENSE architecture.
An experimental Virtual Sensor Framework (VSF) with complex event processing through Esper[52] and ThingML[53] has been created in a thesis work by Jessica Tretter, “Management Framework for Mobile Sensor Data”[54] as a part of a Sensor Data Event Processing (SDEP) Framework.

[52] http://www.espertech.com/
[53] http://thingml.org/
[54] Jessica Tretter, “Management Framework for Mobile Sensor Data”, Master thesis, SINTEF and University of Augsburg, April 2014

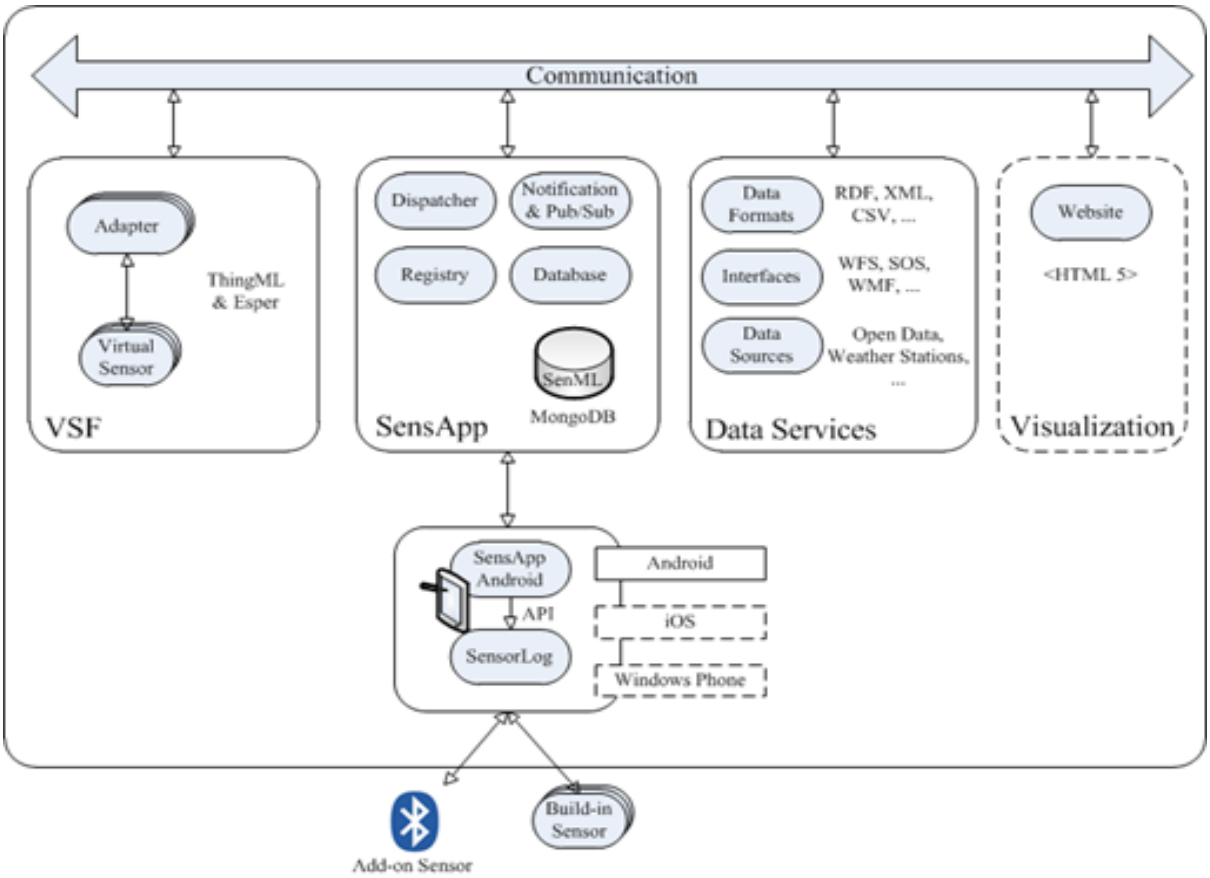


Figure 14-1 Extending the architecture to support event detection and notifications

The Sensor Data Event Processing (SDEP) Framework has validated that it is possible to use the CITI-SENSE sensor storage architecture as a foundation for complex event processing of the managed sensor data.

14.6 Participation and Empowerment services

In parallel with the CITI-SENSE project and the other Citizens' Observatories project a group of CATS projects has been established, to focus on platform support for social innovation. A strategy for relating to these is being established in WP4 as described in D4.2, and it will be considered how the CITI-SENSE platform shall relate to this. One aspect of this is also further relationship to use of the Social Web (i.e. Web 2.0) in addition to the Semantic Web (with Linked Data etc.) (also sometimes referred to as Web 3.0)

Web 2.0

It is still difficult to agree on what the Web 2.0 really is. One definition is *"a set of economic, social, and technology trends that collectively form the basis for the next generation of the Internet—a more mature, distinctive medium characterized by user participation, openness, and network effect"*. Comparing some of the most known applications some commonalities that can be considered the principles behind the Web 2.0:

1. The Web As Platform

2. Harnessing Collective Intelligence
3. Data is the Next Intel Inside
4. End of the Software Release Cycle
5. Lightweight Programming Models
6. Software Above the Level of a Single Device
7. Rich User Experiences

In order to facilitate the development of the so-called Web 2.0 applications, specific strategies, patterns and technologies have been developed or improved. They are often indicated with well-known (and often misunderstood) terms.

- *Mash-up (or mashup)*: A mashup is a technique for building applications that combine data from multiple sources to create an integrated experience. In the Web 2.0 this is often done directly in the browser (consumer mashup) using open Web APIs (e.g. Google Maps API, Wikipedia API, OpenLayers, etc.). The mashup approach helps to achieve the “Web As Platform” and “Lightweight Programming Models” principles
- *Lightweight technologies*: technologies that do not require a heavy upfront investment or operational requirements. These are simpler and less cumbersome to work with. The downside is that lightweight technologies can be less feature rich than their more "heavyweight" alternatives. Examples of technologies considered lightweight are JSON (Javascript Object Notation) for semi-structures data representation, and Javascript/ECMAScript as programming language. In the Web 2.0 they help to reach the rapid development required by the “End of the Software Release Cycle” and the “Lightweight Programming Models” principles.

The social media strategy will be to make it easy to include support for environmental data and sensor data visualisation and interaction through existing social media technologies, utilising their existing and emerging integration APIs.

15 Conclusion and Further Platform Evolution

This D7.5 CITI-SENSE Platform and architecture document has been extended from the baseline D7.4 Platform and architecture documents and provides the foundation for the services provided by the CITI-SENSE platform.

The plan for D7.6 due for M48 (October 2016) is a final consolidation of the data flow support for all of the sensor providers, and also support for further security mechanisms, as well as for social sensors and linked data. 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. – related to the experiences from the pilots and locations from WP2 and WP3 during the deployment in the fall of 2015 and the spring of 2016.

The plan for D7.6 due for M48 is also 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 progress[55]. Some intermediate updates of the document might be expected before the final release of D7.6.

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.

[55] www.dapaas.eu and www.smartopendata.eu

16 Annex A: Use case template and use cases

16.1 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 Resources[56]. 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.

The figure 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.

[56] www.envirofi.eu

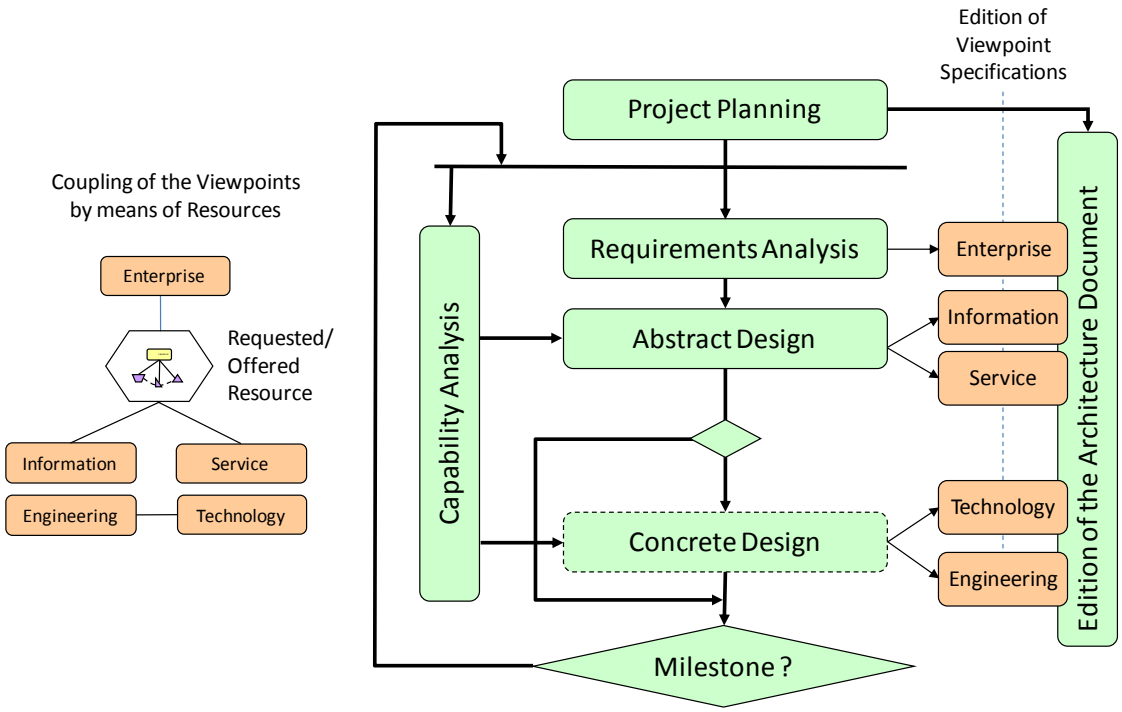


Figure 16-1 Relationship of RM-ODP viewpoints and analysis and design

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 SANY[57], ENVIROFI[58], ENVISION[59], EO2HEAVEN[60] and TRIDEC[61]) 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

[57] EU FP7 project no. 033564 Sensors Anywhere (SANY) - http:///www.sany-ip.eu
[58] EU FP7 FI PPP project, ENVIROFI, http:///www.envirofi.eu
[59] EU FP7 project no. 1234, ENVironmental Services Infrastructure with ONtologies, www.envision-project.eu
[60] EU FP7 project no. 244100 Earth Observation and ENVironmental modeling for the mitigation of HEAlth risks (EO2HEAVEN) - http://www.eo2heaven.org/
[61] 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 16-2 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.

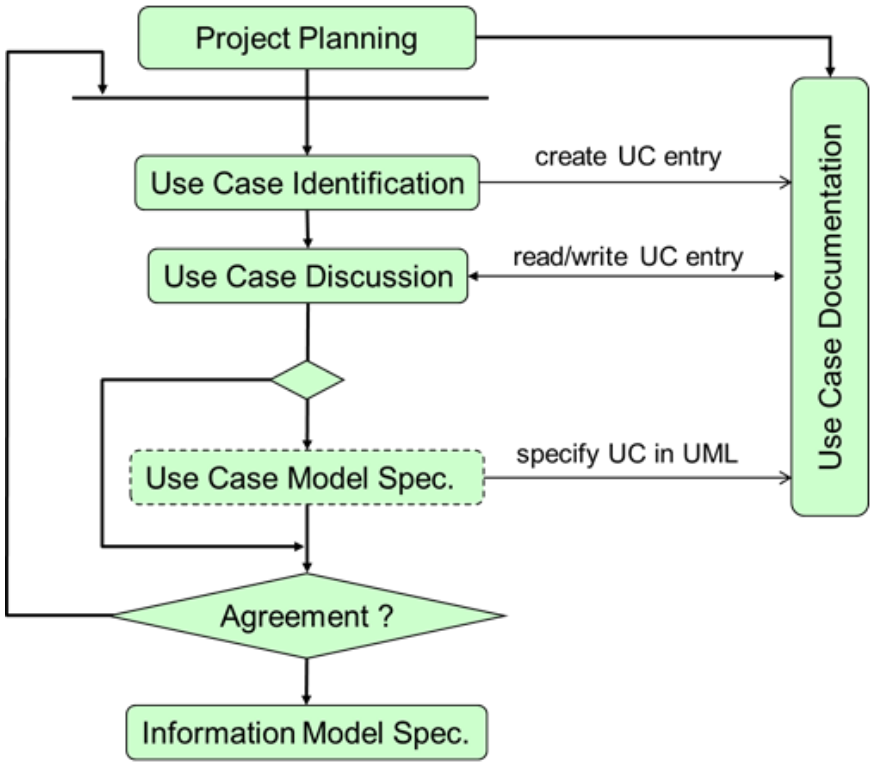


Figure 16-2 Procedure of Use Case Analysis

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.

16.2 Use case template

This template is an extended and modified version of the use case template suggested in ISO 19119[62] [3], which in particular has been extended with a possibility to describe Requested Information Resources found suitable in an SDI setting.

Table 16-1 Description of the Use case template

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.	• 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.

[62] Berre, A. (2015). ISO 19119:2 revision, www.isotc211.org

Main success scenario	Numbered sequence of actions (use case workflow) to be carried out during the execution of the use case.	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: • 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.

16.3 Use cases for the CITI-SENSE Platform

Figure 3-1 presents 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

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:	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:	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 – Question
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:	Possbility 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, anaysed 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 Resources:	Information elements required are related to both existing observation data and data from other sources to be related to.
Preconditions:	Available data is approved
Postconditions:	Aproppriate 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 servics
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 accross 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

17 Annex B CITI-SENSE XML Application Schema

```
<?xml version="1.0" encoding="UTF-8"?>
<schema
xmlns="http://www.w3.org/2001/XMLSchema" xmlns:aqi="http://www.snowflakesoftware.co.uk/example/aqi" xmlns:caqi="http://www.snowflakesoftware.co.uk/example/caqi" xmlns:cts="http:www.citi-
sense.eu/citisense" xmlns:gml="http://www.opengis.net/gml/3.2" targetNamespace="http:www.citi-
sense.eu/citisense" elementFormDefault="qualified" version="2.2">
    <annotation>
        <documentation>Draft Application
Schema for CITISENSE</documentation>
    </annotation>
    <import namespace="http://www.opengis.net/gml/3.2" schemaLocation="http://schemas.opengis.net/gml/3.2.1/gml.xsd"/>
    <import namespace="http://www.snowflakesoftware.co.uk/example/aqi" schemaLocation="aqi.xsd"/>
    <import namespace="http://www.snowflakesoftware.co.uk/example/caqi" schemaLocation="caqi.xsd"/>
    <!--XML Schema document created by ShapeChange -
http://shapechange.net/-->
    <element
name="Answer" type="cts:AnswerType" substitutionGroup="gml:AbstractObject"/>
    <complexType name="AnswerType">
        <sequence>
            <element name="answerID" type="string"/>
            <element name="value" type="string"/>
        </sequence>
    </complexType>
    <complexType name="AnswerPropertyType">
        <sequence>
            <element ref="cts:Answer"/>
        </sequence>
    </complexType>
```

```
    <element
name="Measurement" type="cts:MeasurementType" substitutionGroup="gml:Abstra
ctObject"/>
    <complexType name="MeasurementType">
        <sequence>
            <element
name="hasAQI" type="aqi:AQIvaluesPropertyType" minOccurs="0"/>
            <element
name="hasCAQI" type="caqi:CAQIvaluesPropertyType" minOccurs="0"/>
            <element name="measurementID" type="string"/>
            <element name="value" type="double"/>
            <element name="uom" type="string"/>
            <element name="observedProperty" type="string"/>
            <element name="measuretime" type="dateTime" minOccurs="0"/>
            <element name="measuretimeUnix" type="integer" minOccurs="0"/>
            <element name="latitude" type="double"/>
            <element name="longitude" type="double"/>
        </sequence>
    </complexType>
    <complexType name="MeasurementPropertyType">
        <sequence>
            <element ref="cts:Measurement"/>
        </sequence>
    </complexType>
    <element
name="Observation" type="cts:ObservationType" substitutionGroup="gml:Abstra
ctFeature"/>
    <complexType name="ObservationType">
        <complexContent>
            <extension base="gml:AbstractFeatureType">
                <sequence>
                    <element
name="canHave" type="cts:QuestionnairePropertyType" minOccurs="0" maxOccurs
="unbounded"/>
                    <element
name="sensorID" type="cts:SensorDevicePropertyType"/>
```

```
                    <element
name="hasGlobalCAQI" type="caqi:GlobalCAQIPropertyType" minOccurs="0"/>
                    <element
name="contains" type="cts:MeasurementPropertyType" minOccurs="0" maxOccurs=
"unbounded"/>
                    <element name="observationID" type="string"/>
                    <element
name="starttime" type="dateTime" minOccurs="0"/>
                    <element
name="finishtime" type="dateTime" minOccurs="0"/>
                    <element
name="starttimeUnix" type="integer" minOccurs="0"/>
                    <element
name="finishtimeUnix" type="integer" minOccurs="0"/>
                    <element
name="participantID" type="string" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="ObservationPropertyType">
        <sequence minOccurs="0">
            <element ref="cts:Observation"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
    <element
name="PilotCity" type="cts:PilotCityType" substitutionGroup="gml:AbstractFe
ature"/>
    <complexType name="PilotCityType">
        <complexContent>
            <extension base="gml:AbstractFeatureType">
                <sequence>
                    <element name="cityID" type="string"/>
                    <element name="name" type="cts:citiesType"/>
                    <element
```

```
name="description" type="string" minOccurs="0"/>
                    <element
name="contactPerson" type="string" minOccurs="0"/>
                    <element name="phone" type="string" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="PilotCityPropertyType">
        <sequence minOccurs="0">
            <element ref="cts:PilotCity"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
    <element
name="Question" type="cts:QuestionType" substitutionGroup="gml:AbstractObje
ct"/>
    <complexType name="QuestionType">
        <sequence>
            <element
name="has" type="cts:ResponsePropertyType" minOccurs="0"/>
            <element
name="needs" type="cts:AnswerPropertyType" minOccurs="0"/>
            <element name="questionID" type="string"/>
            <element name="parent_id" type="integer"/>
            <element name="order" type="integer" minOccurs="0"/>
            <element name="type" type="string" minOccurs="0"/>
            <element name="label" type="string" minOccurs="0"/>
            <element name="maxlength" type="integer" minOccurs="0"/>
            <element name="values" type="string"/>
            <element name="required" type="integer"/>
            <element name="rtl" type="integer" minOccurs="0"/>
        </sequence>
    </complexType>
    <complexType name="QuestionPropertyType">
```

```
        <sequence>
            <element ref="cts:Question"/>
        </sequence>
    </complexType>
    <element
name="Questionnaire" type="cts:QuestionnaireType" substitutionGroup="gml:Ab
stractObject"/>
    <complexType name="QuestionnaireType">
        <sequence>
            <element
name="includes" type="cts:QuestionPropertyType" maxOccurs="unbounded"/>
            <element name="questionnaireID" type="string"/>
            <element name="url" type="string" minOccurs="0"/>
            <element name="title" type="string" minOccurs="0"/>
            <element name="description" type="string" minOccurs="0"/>
            <element name="rlt" type="integer" minOccurs="0"/>
            <element name="campaign_id" type="integer"/>
            <element name="created" type="dateTime"/>
            <element name="updated" type="dateTime" minOccurs="0"/>
        </sequence>
    </complexType>
    <complexType name="QuestionnairePropertyType">
        <sequence>
            <element ref="cts:Questionnaire"/>
        </sequence>
    </complexType>
    <element
name="Response" type="cts:ResponseType" substitutionGroup="gml:AbstractObje
ct"/>
    <complexType name="ResponseType">
        <sequence>
            <element name="responseID" type="string"/>
            <element name="source" type="string"/>
            <element name="timestamp" type="dateTime"/>
        </sequence>
    </complexType>
    <complexType name="ResponsePropertyType">
```

```
        <sequence>
            <element ref="cts:Response"/>
        </sequence>
    </complexType>
    <element
name="SensorDevice" type="cts:SensorDeviceType" substitutionGroup="gml:Abst
ractFeature"/>
    <complexType name="SensorDeviceType">
        <complexContent>
            <extension base="gml:AbstractFeatureType">
                <sequence>
                    <element
name="sensorProviderID" type="cts:SensorProviderPropertyType"/>
                    <element name="identifier" type="string"/>
                    <element
name="description" type="string" minOccurs="0"/>
                    <element name="registrationDate" type="dateTime"/>
                    <element name="type" type="string"/>
                    <element name="status" type="string"/>
                    <element name="location" type="string"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="SensorDevicePropertyType">
        <sequence minOccurs="0">
            <element ref="cts:SensorDevice"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
    <element
name="SensorProvider" type="cts:SensorProviderType" substitutionGroup="gml:
AbstractFeature"/>
    <complexType name="SensorProviderType">
        <complexContent>
```

```
            <extension base="gml:AbstractFeatureType">
                <sequence>
                    <element
name="position" type="cts:PilotCityPropertyType" minOccurs="0" maxOccurs="u
nbounded"/>
                    <element name="providerID" type="string"/>
                    <element name="name" type="cts:sensorprovidersType"/>
                    <element
name="description" type="string" minOccurs="0"/>
                    <element name="location" type="string" minOccurs="0"/>
                    <element
name="contactPerson" type="string" minOccurs="0"/>
                    <element name="phone" type="string" minOccurs="0"/>
                    <element name="url" type="string" minOccurs="0"/>
                    <element name="city" type="string" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="SensorProviderPropertyType">
        <sequence minOccurs="0">
            <element ref="cts:SensorProvider"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
        <attributeGroup ref="gml:OwnershipAttributeGroup"/>
    </complexType>
    <simpleType name="citiesType">
        <restriction base="string">
            <enumeration value="Oslo"/>
            <enumeration value="Barcelona"/>
            <enumeration value="Vienna"/>
            <enumeration value="Edinburgh"/>
            <enumeration value="Haifa"/>
            <enumeration value="Ljubiana"/>
            <enumeration value="Ostrava"/>
            <enumeration value="Belgrade"/>
            <enumeration value="Vitoria"/>
```

```
        </restriction>
    </simpleType>
    <simpleType name="sensorprovidersType">
        <restriction base="string">
            <enumeration value="QU"/>
            <enumeration value="Alphasense"/>
            <enumeration value="Obeo"/>
            <enumeration value="ATEKNEA"/>
            <enumeration value="Geotech"/>
            <enumeration value="Airbase"/>
            <enumeration value="CVUT"/>
            <enumeration value="UHOPPER"/>
            <enumeration value="DVET"/>
            <enumeration value="JSI"/>
            <enumeration value="Sensapp"/>
            <enumeration value="PAQ-MAP"/>
        </restriction>
    </simpleType>
</schema>
```