Horizon 2020

Societal Challenge: Improving the air quality and reducing the carbon footprint of European cities

Project: 690105 – ICARUS

Full project title:

Integrated Climate forcing and Air Pollution Reduction in Urban Systems

D7.2: Report on the design of technical framework and system architecture of the ICARUS DSS

WP7: Motivating citizens towards the vision

Lead beneficiary: UPCOM

Date: June 2017

Nature: Report

Dissemination level: Public

TABLE OF CONTENTS

Document Information

Grant Agreement Number	690105	**Acronym**	ICARUS
Full title	Integrated Climate forcing and Air pollution Reduction in Urban Systems		
Project URL			
Project Officer	Mirjam Witschke - Mirjam.Witschke@ec.europa.eu		

Delivery date	**Contractual**	April 2017	**Actual**	June 2017
Status	Draft □		Final √	
Nature	Demonstrator □	Report √	Prototype □	Other □
Dissemination level	Confidential □	Public √		

Responsible Author (Partners)	Theodoros Nikolakopoulos (UPCOM)			
Responsible Author	**Name** Theodoros Nikolakopoulos		**Email**	tnikolakopoulos@upcom.eu
	Partner	AUTH	**Phone**	
Other partners (Institution)	Alberto Gotti, Denis Sarigiannis (AUTH) Emmanouil Tsiros (KARTECO) Emmanouella Siora (ARTEMIS)			

Document History

Name (Institution)	Date	Version
Theodoros Nikolakopoulos (UPCOM)	March 2017	1.0 Draft
Emmanouella Siora (ARTEMIS)	April 2017	1.1
Emmanouil Tsiros (KARTECO)	April 2017	1.2
Alberto Gotti (AUTH)	April 2017	1.3
Nicos Evmides (UPCOM)	May 2017	1.4
Kostas Flokos (UPCOM)	June 2017	1.5
Theodoros Nikolakopoulos (UPCOM)	June 2017	Final

1 Introduction

This document elaborates the architecture of the ICARUS Decision Support System a web-based, flexible and interactive platform aimed at assisting stakeholders in the selection, application and evaluation of the available datasets and tools for urban impact assessment in support of air quality and climate change governance at different spatial and temporal scales and taking into account the specific regulatory context.

The document is organized in a series of chapters addressing specific needs of the DSS. More in detail, in chapter 2, a short description of the existing DSS taxonomies and employed techniques is provided. Chapter 3 provides a detailed analysis of the modular architecture of ICARUS DSS; this chapter concludes with a validation that all functional requirements are covered by the components of the DSS. In chapter 4 we proceed to a study of the most representative user stories among the ones described in deliverable D7.1, so that the interactions among the components can be displayed and explained. Chapter 5 presents the physical infrastructure, with a special focus to the ICARIUS GeoDatabase, that plays a key role in the operation of the DSS, since it persists and gives access to all the necessary geospatial data. Finally, in chapter 6, we provide a description of the methodological approach that will be used for the development of the DSS.

2 Decision Support Systems

Decision Support Systems (DSSs) are information systems developed to assist their users in the decision-making process. A DSS offers tailored solutions for the needs of the industry sector it has been designed for.

The typical architecture of a DSS system follows the 3-tier paradigm, consisting of:

- The presentation tier, that contains the user interface
- The logic tier, that stores the computational models and handles their execution
- The data tier, that stores all data that are required for the execution of the models and the data produced by it

According to the extent of the assistance that a DSS can provide to its users, it can be classified to one of the following categories[1]:

- Active DSS: systems that guide the user throughout the decision-making process by providing them with suggestions for their decisions
- Passive DSS: systems that allow the users to evaluate the expected results of a perpetual decision

There is also one more category, this of the cooperative DSS systems, that can suggest some basic or predefined solutions to their users, who can in turn modify and parameterize them before having them reevaluated by the system. ICARUS DSS will come under this category, because it allows for flexibility and parameterization of that of an active DSS system, but without expecting form its users such expertise as a passive DSS.

DSSs can be further classified according to the interaction type with the users to the following categories:

- Communication-driven: they support and facilitate collaboration among several users
- Data-driven: they focus on the manipulation of data that can be either native to the system or acquired later by it
- Document-driven: focus is given in supporting processing of data of various formats and datatypes
- Knowledge-driven: based on predefined rules and procedures for resolving relative problems

[1] "Decision support systems for environmental problems: Scientific approach, requirements of structure and data on specific purpose types", Sven Meyer et al., Integration of Environmental Information in Europe, 2010

- Model-driven: they use models in order to allow the user quantify and parameterize the factors that play a key role to the decision they want to make

ICARUS DSS will be a model driven DSS, in order to take advantage of the environmental, emissions, concentrations, and other models that will be developed by the members of the Consortium and that can help predict the results of a policy in a broad range of temporal and spatial scales. Special focus will be given to the exploitation of data of several data types and formats, originating by numerous different sources, such as meteorological and concentration measurement stations, smartphone and wearable sensors.

3 Components of the ICARUS DSS

The design of the ICARUS DSS was driven by non-functional requirements, such as ability to change/add new functionality easily, scalability, etc. as well as the current technological trends in the software development world.

One word that can accurately described the stated non-functional requirements is **Agility** and it is extensively used as the core requirement in all modern systems and applications.

Agility refers to the way pieces of the application interact, as well as the methodology used in order to achieve it. For the methodology, Agile Scrum has been selected and a short description can be found in Chapter 6.

Agility is achieved with loosely-coupled components, components that do not depend on lots of others and that may be easily used by others. When those components are not big as applications, but smaller, with clearly identified functionality and interfaces to other components and the world, the result is increased independence between the components, allowing targeted updates, both when bugs arise or new/updated functionality needs to be developed. Agility is then achieved to a high degree.

This ability to partition an information system in small, clearly identified and self-contained components able to run in their own process space – eventually replicated - is currently described in the literature as a "**micro-services based architecture**". The difference between monolithic vs micro-service based architectures is depicted in The architecture of ICARUS DSS is highly modular, so that it can provide the seamless integration of new functionalities in the future. On the top of the hierarchy lie the User interface module along with the Data collection devices, which.

An effort has been done to identify the core components of the ICARUS DSS so that they can easily be considered micro-services themselves and make use of lower level micro-services as well. The description of those components follows; it occasionally identifies lower level services that make the higher-level component.

There is strong belief that the proposed architecture meets the requirements of the ICARUS DSS, both functional and non-functional, especially the ones that are commonly ignored, such as ability for change, scalability, traceability, etc.

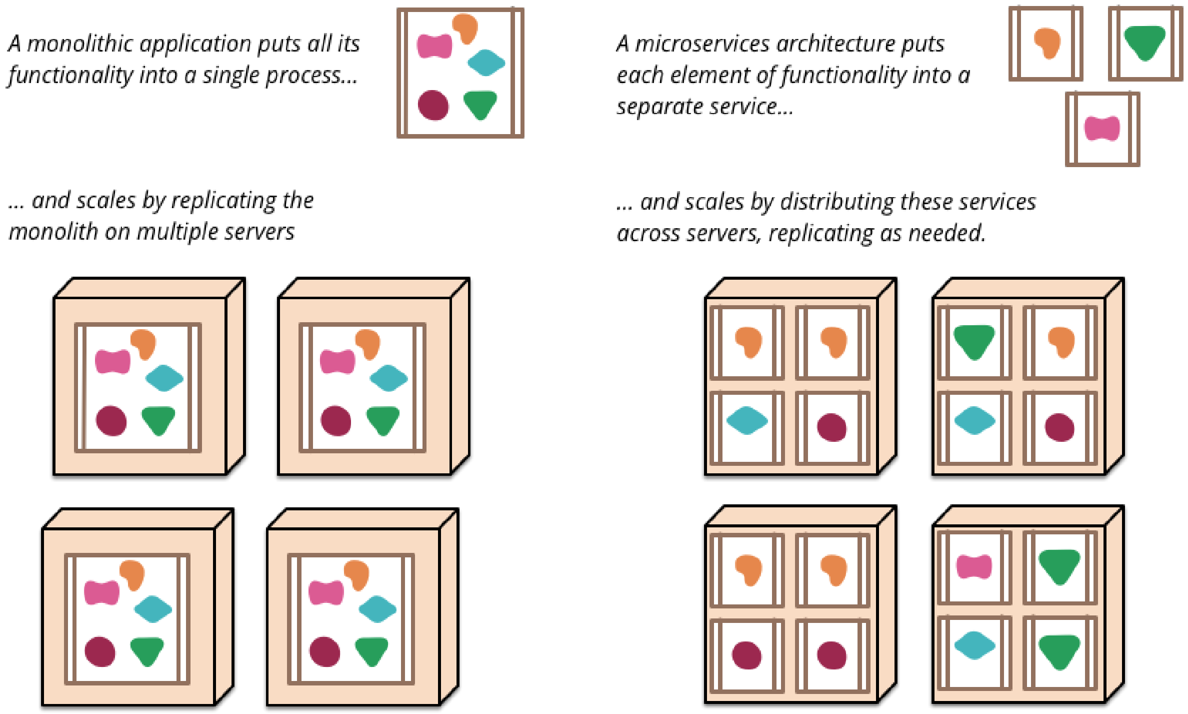
A monolithic application puts all its functionality into a single process...
A microservices architecture puts each element of functionality into a separate service...
... and scales by replicating the monolith on multiple servers
... and scales by distributing these services across servers, replicating as needed.

Figure 1 - Monoliths and micro-services[2]

The architecture of ICARUS DSS is highly modular, so that it can provide the seamless integration of new functionalities in the future. On the top of the hierarchy lie the User interface module along with the Data collection devices, which are the point of interaction of the DSS with its users and data providers. Directly below, there are five modules that are directly controlled by them. The Policy, Raw data and Policy engagement management modules address the needs of the core functionalities of the system. The next two, i.e. the Account management and the Billing modules, serve administrative purposes and business functionalities related to the DSS.

The Task management module is responsible for coordinating all computational tasks done by the DSS and the Model management contains the core computational parts and the model results data. On the bottom of the architecture, the Resource management module allocates resources and enables the Task manager to access them for the execution of the simulations.

In the left column of Figure 2, three components are displayed that can interact directly with most of the modules of the DSS; the Communication services, the Event management module and the Usage monitoring. All external access to the DSS, whether it is done through the User interface or by the Data collection devices, will have to be approved by the Access control module.

[2] Copied from https://martinfowler.com/articles/microservices.html

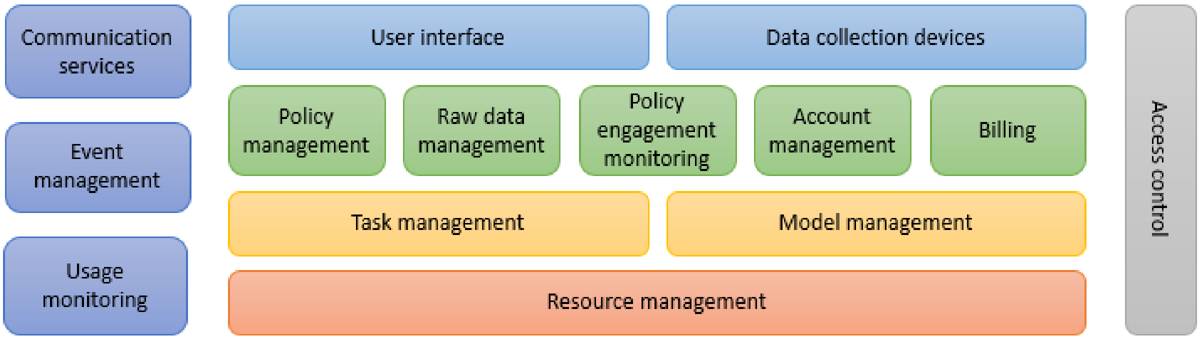

Figure 2 - Brief component diagram of ICARUS DSS

On the next page, you can see the detailed component diagram of ICARUS DSS.

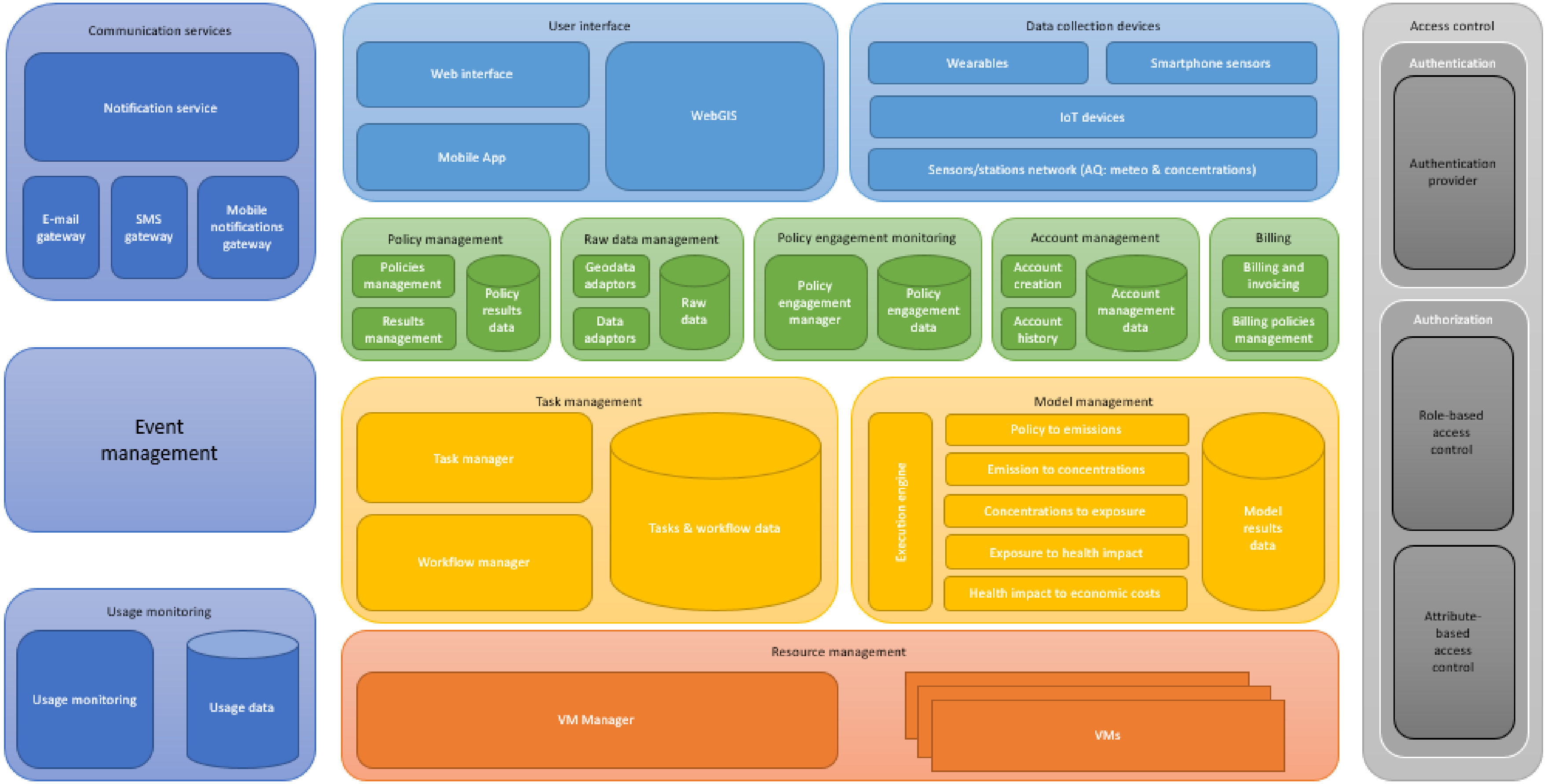

Figure 3 - Detailed component diagram of ICARUS DSS

3.1 User interface

The user interface component is responsible for all interactions with humans, from any channel that may happen, web, mobile, wearables, etc.

In our original analysis, we have identified two channels that need special attention, with the remaining falling under one or the other category: **web interface,** for all interactions of the user via web browsers on the desktop, the mobile or elsewhere and **mobile applications**, for all interactions of the user via mobile devices, exploiting the extra features and advantages of the mobile and wearables. The user of the GIS technology for all geospatial information elevates the WebGIS component as a user interface component; the users will interact directly with it to enrich their interaction with geospatial data, even though this component will be encapsulated in the other two components, the web interface and the mobile applications.

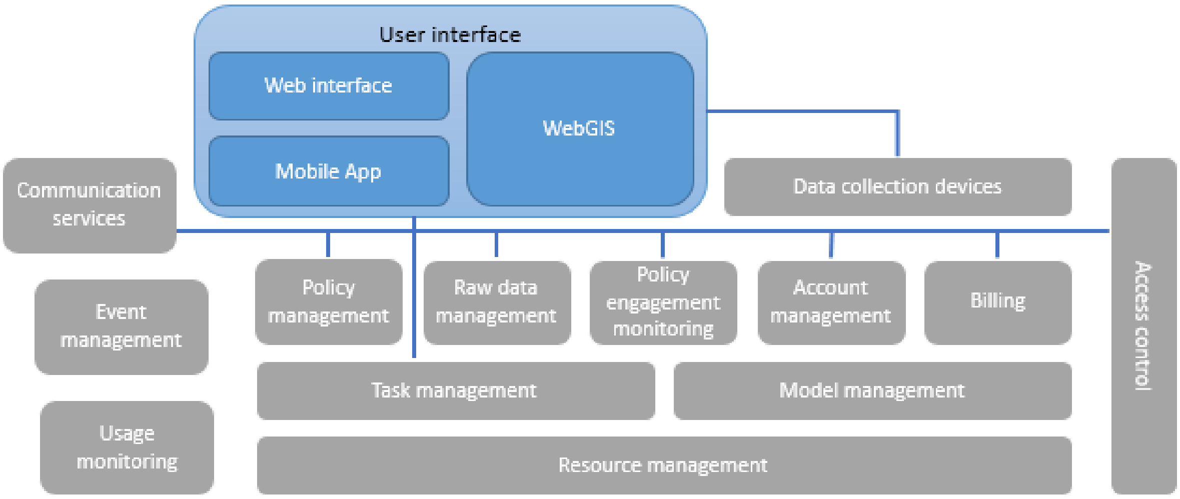

Figure 4 - User interface module

The interaction of the user interface component with the other components is shown in the previous diagram. It is evident that there are a lot of interactions with most of the components. The proposed architecture foresees the execution of the User Interface locally, at the user's browser on the desktop or mobile, on the mobile device as a mobile application. All other components will be centrally located, at the datacenter hosting the ICARUS DSS; the interface will be hosted on the central system as well, but it will be run on the client-side, to off-load processing and offer better responsiveness to the user.

The interactions between the user interface components and the other back-end components will be managed via a single access point, the Access Control[3], which will also play the role of a proxy to the system. This proxy will control the ability of the user to interact with the backend service, identify the correct instance of the backend to invoke, as well as improve the performance of similar requests, by offering an extensive caching mechanism.

The identified sub-components of the user interface are described in more detail below.

3.1.1 Web interface

3.1.1.1 Functionality

The Web Interface is the interface used by end-users when interacting with the ICARUS DSS via their web browsers. It runs on any platform offering a web browser, from desktop, laptop and server-based systems, to mobile devices, tablets, feature-phones, smart TVs and other smart objects.

To cope with the different end-user device used to execute the DSS, the web interface will be highly **responsive**, offering different layouts for different screen sizes.

As you can see below, the layout of the same web page changes depending on the size of the target screen:

[3] Eventually, the Access Control component is composed of lower level components, such as the Proxy, the Authentication Service, the Authorisation Service, etc. Their description is currently outside the scope of this document.

Figure 5 - Responsive design of a web interface

This is exactly what ICARUS DSS will offer for its web interface.

Additionally, depending on the identified end-user device parts of the functionality may be removed or modified to offer a smoother interaction to the user; feature-phones, such as the old Nokia phones, do not have big processing capability; for them, the WebGIS option may be suppressed and replaced with simple maps without interaction capabilities. Even though that functionality removal may seem undesirable by the user, it is mandatory to protect the smooth execution of th remaining parts of the interface. In summary, the web interface will also be **adaptive** to the target devices used to execute it.

The need to cover the biggest-possible number of end-user devices and systems is constrained by the actual number of resources available to support those systems. Architecturally, the user interface will be able to support any browser, on any device and operating system. In practice, a limited list of browsers will be supported, at least during the ICARUS project implementation. This list will be limited to the following browsers[4]:

- Internet Explorer
- Chrome
- Firefox

[4] The minimum version of each browser supported will be identified during the execution of the project.

- Safari

3.1.1.2 Internal architecture

The web interface will be developed using the Model-View-Controller (MVC) pattern to allow maximum agility and ability for quick corrections and updates. The MVC implementation selected to be used for ICARUS DSS is **Angular** 4 (https://angular.io/), using **Typescript** (https://www.typescriptlang.org/) as the main development language. Typescript compiles to JavaScript that is next executed on the client browser.

A general overview of a simple page in Angular follows:

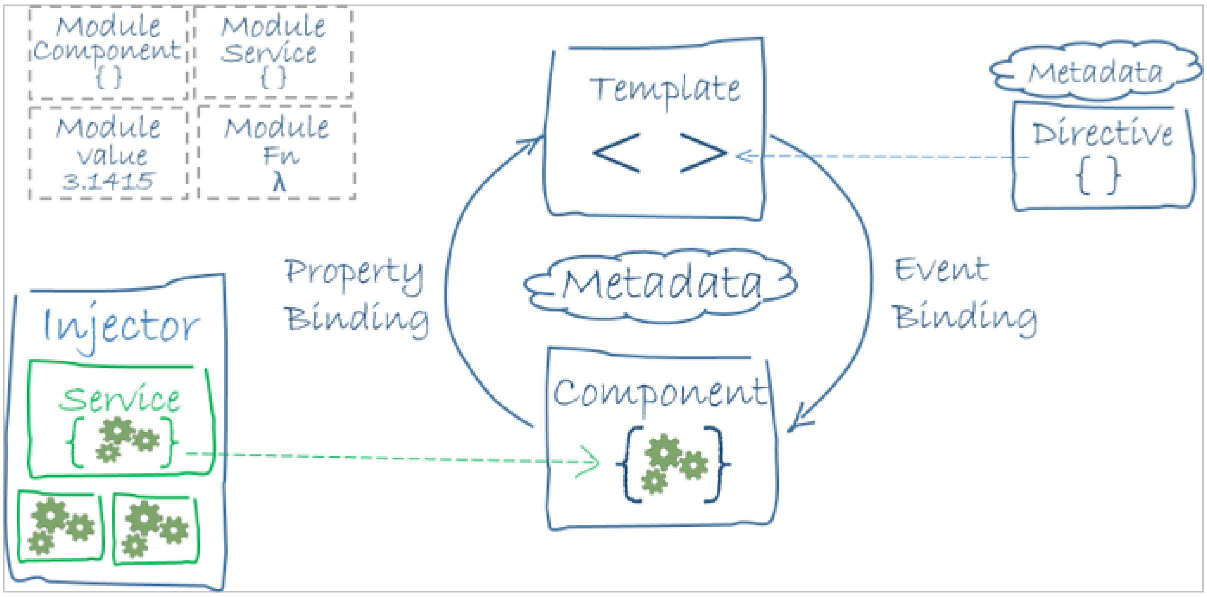

Figure 6 - Angular page scheme

Angular applications are developed by composing HTML *templates* with Angularized markup, writing *component* classes to manage those templates, adding application logic in *services*, and boxing components and services in *modules*.
Then the application is launched by *bootstrapping* the *root module*. Angular takes over, presenting the application content in a browser and responding to user interactions according to the instructions provided.

One can use Angular for all types of web-based projects. In ICARUS, we decided to use single-page application architecture for all developments, including the DSS. **Single-Page Applications** (SPAs) are Web **apps** that load a **single** HTML **page** and dynamically update that **page** as the user interacts with the **app**. SPAs use AJAX and HTML5 to create fluid and responsive Web **apps**, without constant **page** reloads. However, this means much of the work happens on the client side, in JavaScript.

The choice was evident, as this design choice offers improved user experience (the user does not have to wait for page loading from the server, showing blank pages in between), higher responsiveness and overall acceptance by the users. More and more web applications are currently developed and deployed using the SPA paradigm, with one of the first to change the previous tendency being google mail (https://mail.google.com) .

The use of Angular gives the ability to the development team to better structure and organize their work, in components, modules and applications. In Icarus, this is also important as that partitioning of the web user interface will allow development in parallel, by different front-end developers, respecting common guidelines and reusing code developed by their peer colleagues.

Finally, the interaction with WebGIS will happen smoothly, as the Javascript library offered by this last can easily be integrated in the overall Angular-based page and interact smoothly with the GIS backend. If needed, hooks will be developed between the WebGIS JS library and Angular to better control the overall interaction of the user with the web interface.

3.1.2 WebGIS

The Web GIS module design process includes a GIS needs assessment and a system architecture design. The system architecture design is based on user workflow requirements identified in Deliverable "D7.1 Technical report on user requirements and functional specifications of ICARUS DSS". The most effective system design approach considers user needs and system architecture constraints throughout the design process.

The functional user requirements, which the Web GIS is called to address are shown at the following table.

Identifier	Name	Priority
UR_F2	The DSS should store the results of the simulations performed for policy options or technological measures	Should
UR_F3	The DSS must allow viewing the results of simulation for predefined policies and/or technological measures in a region	Must
UR_F4	The DSS user must be able to view the results of the impact in various formats, including maps, tables and/or time series	Must
UR_F5	The DSS user must have the ability to process the results using geospatial tools	Must
UR_F8	The DSS must allow inputting new emission measurements and integrate it with the existing data available to the DSS	Must
UR_F9	The DSS must allow inputting new pollutant concentrations and integrate it with the existing data available to the DSS	Must
UR_F13	The DSS must allow the definition of geographical entities and decentralized management	Must
UR_F15	The DSS must allow viewing and exporting emission measurements for a region over time	Must
UR_F16	The DSS must allow viewing and exporting pollutant concentrations for a region over time	Must
UR_F19	The DSS should support import of spatial databases at the city level	Should

Table 1 - User requirements addressed by the WebGIS submodule

GIS Needs Assessment: The GIS needs assessment includes a review of user workflow requirements and identifies where the Web GIS can improve user productivity. This assessment identifies GIS functionality and data requirements and an implementation strategy for supporting the user needs. It should provide answers to questions on:

- the purpose;
- the goals;
- the target audience;
- the tools and functionality; and
- the content.

System Architecture Design: The system architecture design is based on user requirements identified by the GIS needs assessment. It includes the definition of the system design specifications and the selection of strategies, which are best suited to accomplish the goals set. There should be a clear understanding of the requirements before the development of the system design specifications.

3.1.2.1 Web GIS architecture

Web GIS solutions include integration of a variety of products. Each software, language or library is a part of the total environment that must be integrated together as a whole. Integration is made possible through compliance with generally accepted industry standards.

A general understanding of the fundamental supporting technologies associated with a Web GIS solution provides a solid foundation for the system architecture. Following are the key building blocks supporting a Web GIS environment.

Web server: A web server is a computer system that processes requests via HTTP, the basic network protocol used to distribute information on the World Wide Web. The primary function of a web server is to store, process and deliver web pages to clients in response to their requests. The communication between client and server takes place using the Hypertext Transfer Protocol (HTTP).

The Web GIS will be hosted inside an Apache web server. Apache is the most widely used web server software. It is an open source software available for free, which runs on 67% of all web servers in the world. It is fast, reliable and secure.

GIS server: The GIS server allows sharing, processing and editing of geospatial data and provides great flexibility in map creation and data sharing. The Open Geospatial Consortium (OGC) web services provide a way to make maps and data available in an open, internationally recognized format over the web. OGC has defined the following specification protocols for the transfer of geospatial data from servers to client applications.

- WMS (Web Map Services): for serving collections of layers as map images;
- WFS (Web Feature Services): for serving data as vector features; and
- WCS (Web Coverage Service) services: for serving data as raster coverages

The GIS server that will be used in the development of the Web GIS is GeoServer. GeoServer is an open source, java-based software server. It follows the standards set by the Open Geospatial Consortium.

GeoDatabase: A GeoDatabase or spatial database is a database which is optimized to store and query data that represents objects defined in a geometric space. It allows the representation of geometric objects such as points, line strings, polygons, multipoint, multiline strings, multipolygons and geometry collections. It also provides spatial functions like area, distance, union, difference, buffer, touches, intersects, within, contains, overlaps etc.

The spatial data will be organized into a PostgreSQL / PostGIS database. PostgreSQL is an open source, powerful, mature, reliable and platform independent object-relational database system. PostGIS is a spatial database which is built on top of a standard PostgreSQL database. It adds the necessary support for spatial data types and a multitude of spatial functions.

Client: Client is referring to the user's computer, specifically their web browser.

Mapping application: The mapping application is used for loading, displaying, and rendering interactive maps. It incorporates regular mapping functions, allows for the selection of elements, data editing, theme-based dynamic tasks and other.

The open source, high-performance, JavaScript library of Open Layers will be used for the development of a custom mapping application, which will provide the user of the Web GIS with all the necessary mapping tools and functionalities.

The tools and functionalities, which will be included in the Web GIS are shown in the following table.

Mapping	• zoom in / out / pan • show selected layers • show / hide legend • extract information from selected features ○ in tabular form ○ in chart / diagram form • print map • share map with social media (e.g. Facebook, twitter) • mapping animation (to visualize map changes through time)
Geo-spatial tools	• Extraction tools • Statistical analysis and measurement tools • Conversion tools
Querying	Search functions according to • geographical area • attributes • time
Editing	Add/remove/modify selected features
Downloading	Retrieve information in several available file formats, e.g. kml, ESRI shape file, text files, CSV file, tiff files etc.

Table 2 - WebGIS functionalities and tools

The following figure illustrates how these components are working together to create a web map GIS application.

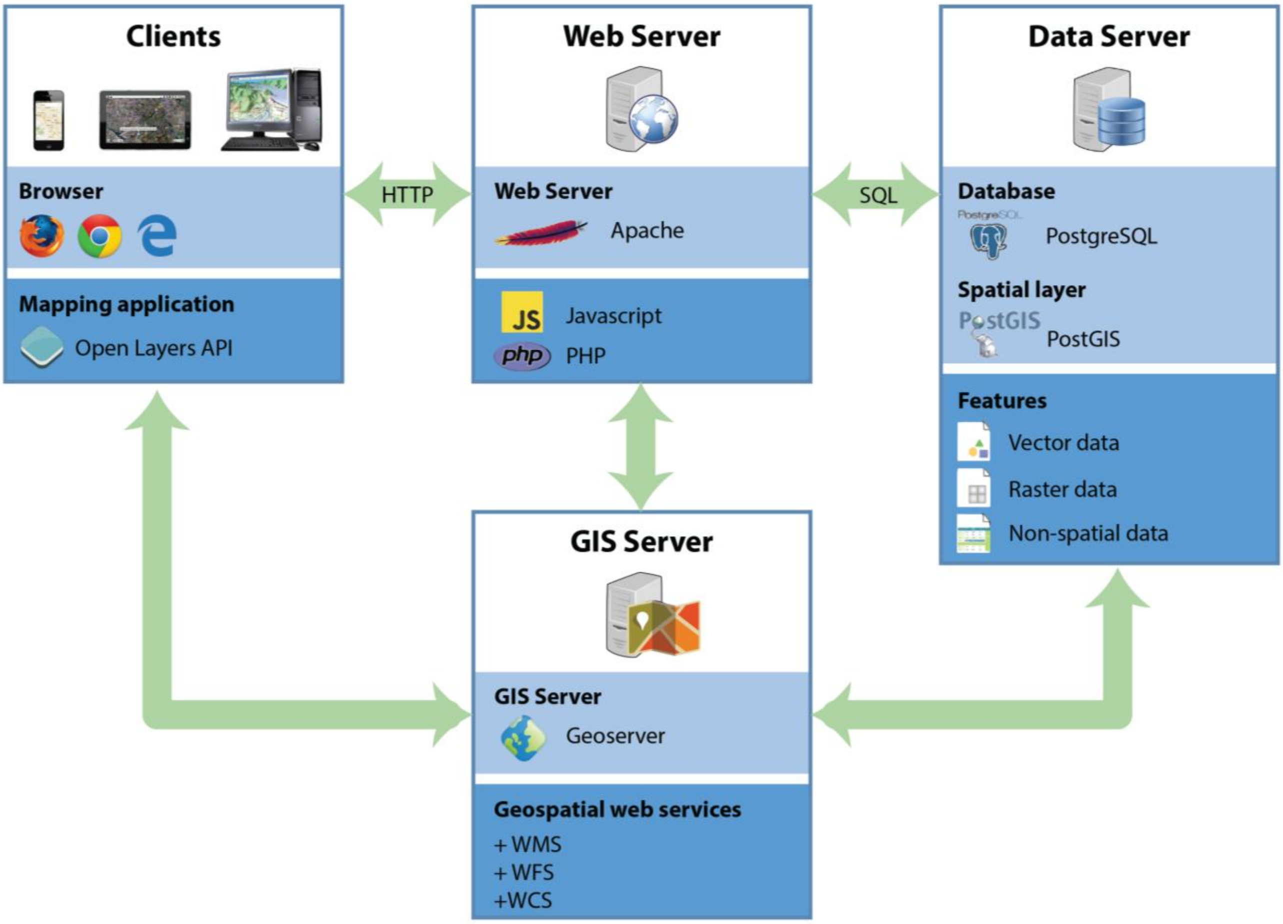

Figure 7 - WebGIS interactions

The procedure that takes place after user's request is as follows:

1. The user through the browser and the mapping application sends a request to the Web server
2. According to the request, the Web server finds out the relevant attribute data from the Database and passes the corresponding parameters to the GIS server.
3. The GIS server requests from the database specific spatial data and process them. It calls the appropriate method, accesses data and conveys the data obtained to the Web server.
4. Web server package contains the data obtained in the query as a JSON object or a web page. The former is displayed on the map as the form of WMS and the latter is displayed in the form of a property window so that the user can see the results in the browser

3.1.3 Mobile Applications

3.1.3.1 Functionality

The mobile applications developed will offer the complete functionality foreseen in Icarus, and interact with the DSS and the other developed services without the user being aware of the responsible backend.

The applications will be mainly used for end-users/citizens, who have been sensitive about environmental issues, such as global warming, impact of pollutants in humans' health, etc. Their access to the DSS will be relatively limited, with the following functionality offered to them:

- View current status of pollutant concentrations in specific geographical area; typically the system will pick-up the current location of the user, using the integrated GPS module, and show the measurements in the area; the user would be able to use a map to select another location to view relevant metrics. Along with concentrations, the health impact may have been computed and made available to the user, from DSS data

- View policy engagement by the local authorities on future/current environmental friendly actions; the citizen would be able to view which actions the authorities have committed and view the results of the simulations performed and analysed by the DSS

- Provide environmental information back to the DSS, collected either by the applications themselves or wearables connected to the mobile application

The functionality related to the engagement of the user is outside the scope of the DSS and as such will be described in deliverable D7.3, due in month 24.

3.1.3.2 Internal architecture

The mobile applications will support the dominant mobile operating systems nowadays: Android and iOS.

An extensive discussion has taken place between partners on the best platform to use for the implementation of the mobile applications.

- The first group suggested the use of **HTML5/JS cross-platform development**, which could be considered a small extension to the Web Interface, described previously.

  The advantage of the solution was two-fold:

  - avoiding replicating code for similar functionality, between the web interface and the mobile applications, and

  - writing code just once for both mobile operating systems.

  The disadvantages were:

  - Slower reactivity of the applications; native code performs better than the interpreted Javascript and this is noticeable by experienced users
  - Big parts of the code would have been – in any case – in native code, as the interaction with hardware options and external devices can only be coded natively

- The second group suggested the use of **native mobile application development**, in Java for android devices and Objective C for iOS.

  The advantages of the suggestion were:

  - Faster code execution and better, more natural feeling for the end-user
  - Parts of the code interacting with hardware options/devices would be developed natively, in any case
  - Use of specific functionality and user experience offered by the native operating system can be exploited and offered to the user; both OS producers try to offer features that differentiate one from the other. Those features are more easily exploited in native development, improving the overall appreciation of the user for the developed application

  The disadvantages were:

  - Double code base, one for Android and another one for iOS
  - Increased development time and use of resources

The team finally decided to select the second option and opt for native mobile application development. All believed that the user experience to the users is more important than the time and the resources needed for the implementation. Previous efforts in user engagement efforts have failed because the user experience was not properly evaluated.

Additionally, as some parts of the delivered interface is provided by third parties, such as the WebGIS component, the team decided to use the WebView feature of both mobile operating systems to integrate this and other components in the native interfaces.

Coding standards and agreed guidelines across the team will help achieve a uniform source code base, easy to maintain for the years to come.

The detailed architecture of each mobile application developed will be part of internal to the project documents. The high-level architecture of those applications will be included in a revised version of this deliverable in due time – when the development activities would start.

3.2 Data collection devices

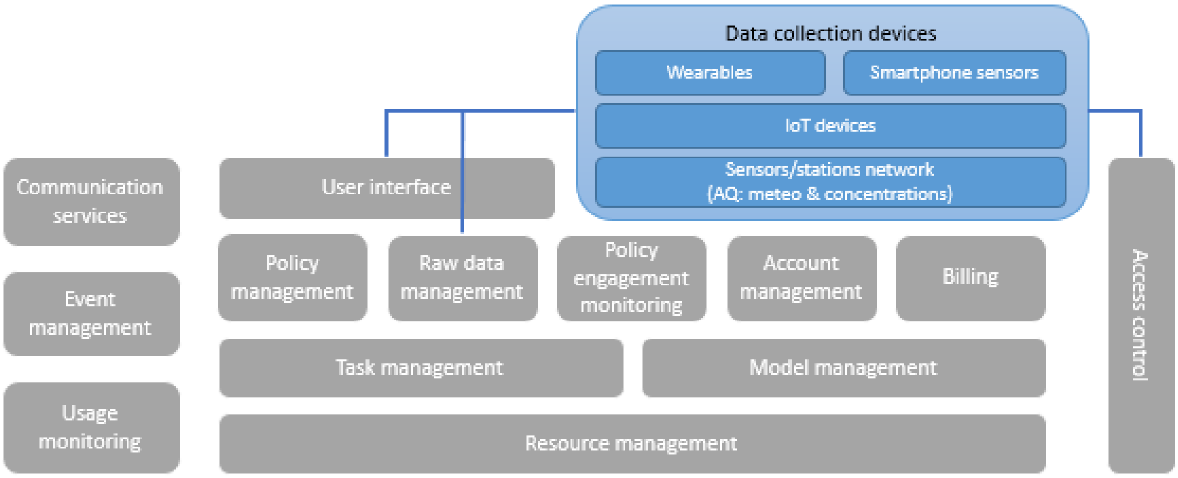

Figure 8 - Data collection devices component diagram

In this section, a brief overview of the data collection devices follows. A thorough published methodological review of "The use of sensor technologies in defining external exposure at individual level" has already been conducted at the Deliverable D1.3 with the latest advances in external sensor technologies and geo-referenced systems used. Nowadays, as a consequence of technological advances in sensing, wireless communication and devices miniaturization, associated cost downsizing, and especially the widespread use of smartphones, it is possible to assess external exposure of individuals to variety of environmental stressors in real-time and with very high spatial and temporal resolution. Sensor technology and associated IT infrastructure are developing very fast. Innovations in sensors technology create possibilities to collect environmental data / exposure-related data at a great depth, more easily and frequently, enhancing understanding of the linkages between environment and health.

Data collection on external environmental exposure and exposure determinants is performed through a wide range of devices by combining location and activity data in different microenvironments. In general, two types of sensors are used: air quality sensors (AQ), and physical activity sensors (PA) including location (GPS) tracking sensors. Within the first group are considered sensors measuring presence/concentrations of gases, particulate matter as well as specific environmental (e.g., meteorological) parameters. Second group comprise various physical activity sensors: accelerometers, respiration, heart rate, R-R interval, breathing rate, etc. and environmental parameters like temperature, relative humidity and visible light. To this end, a multi-platform data collection tool should be developed in order to store, manage and process all data coming from different devices.

Apart from providing air pollution and climate change information of the European cities to stakeholders, the ICARUS DSS has the ultimate purpose to be networked with data collection devices and then share feedback at individual level for human heath impact assessment and other environmental parameters of interest in urban systems where high spatiotemporal variations exist. As outlined in Deliverable DE1.3, It is anticipated that ICARUS will combine data collected by sensor technologies with the agent-based modeling as to capture individual spatio-temporal behaviors and to model individual exposure. Considering the outcome of the ELS method, it is referred that for ICARUS, Ideally, multi-sensor setup should comprise environmental, location and personal movement/activity data, combined in user friendly, easy to use "package". The ICARUS data collection devices that will be finally used are under overview, since not only new sensor and state of the art mobile technologies in collecting exposure data emerge but also many of them are at testing phase only and without clearly demonstrated fitness of purpose. However, an initial consideration must be done for the device types of indoor and outdoor exposure assessment that the ICARUS DSS could support by now. These devises are distinguished in: Wearables, Smartphone sensors, IoT devices, and Sensors/stations network. Details for the aforementioned devices are provided at the following table.

Data collection device	Description
Wearables	Wearable sensors are portable devices and are used for personal exposure assessment. Such devices are: -Wristbands that track the chemicals a person encounters every day such as Silicon wristbands that detect PAHs - Polycyclic Aromatic Hydrocarbons -Wristbands that calculate steps, location, intensity of activity, heart rate -Wristbands for feelings -Waistband with sensors for location and speed -Backpack with AQ sensors for PM, VOCs, UV - Personal Sensors to Assess the Exposure and Acute Health Effects
Smartphone sensors	GPS-enabled smartphones that give information about spatiotemporal movements of people involved in studies and additional information. Smartphone sensors provide map visualization of location, activities, noise levels, and customized mobile applications for human health such as UV exposure, eFood patterns etc.
IoT (Internet of things)	IoT devices are used for home, environmental and personal exposure assessment. IoT devices include all smart and state of the art devises oriented to exposure assessment. Such devices are the Air quality Egg… Smart devices for more reliable "time - geography of exposure" data.
Sensors/stations network (AQ, meteorological & concentrations)	At this category are included: -the Fixed ground-based monitoring networks which are mostly used for city exposure assessment through the processes of atmospheric dispersion modeling, remote sensing and data fusion. The ground station network of participating cities

	that has been determined in WP3 can provide regularly environmental data to the ICARUS DSS. - all sensors of AQ monitoring from conventional meteorological and concentrations measurements to the latest technologies of AQ micro-sensors such as low-cost AQ monitoring technologies and other.

Table 3 - Data collection device types

Personal Exposure Assessment is performed either with only access to AQ information, or with combination of the individual exposure profile, or, more thoroughly, through the combination of air concentrations and physical activity data, measured at personal level, with additional external information or services (e.g. modeled AQ values, on-line health information services and similar applications). The latter case blends objective data (PM, hearth rate and cadence measurements) with subjective data using questionnaires to record participants perception of air quality, as well as reported health symptoms. “Questionnaires” and “Time Activity Diaries” can provide important feedback for individual exposure. In detail, “Questionnaires” preferably in tablets- online into a web form for later upload into a database-, demonstrate information about the housing characteristics and household (building materials, insulation, cleaning products used etc.), demographics and socio-economics. A time-microenvironment-activity diary combined with concentration measurements can lead to the estimation of a personal exposure profile taking into account different levels of exposure in different microenvironments. Moreover, “Time Activity Diaries” can be correlated with the various portable positioning system receivers, usually GPS-enabled smartphones that give information about spatiotemporal movements of people involved in studies.

The aim of collecting data from the aforementioned devices is to merge air pollution and climate measurements with spatiotemporal movements of individuals. It is crucial when AQ measurements are captured to remark when and where this measurement was made. Thus, the devices should be synchronized. To this purpose, a multi-platform data collection tool is suggested, a portal on which data from multiple devices can be uploaded or synchronized. The portal is hosted on a constantly monitored server with the benefits of open APIs and the “internet-of-things” concept. It is of great importance to point out the necessity of Anonymized accounts/data and Participants to see all their data in real time and Researchers to have direct access to all data for further analysis. Participants also should be willing to have sampling equipment in home and accept fieldworkers to survey their home, and place data collection devices in the home, they should be willing to carry around personal monitors and perform any functions required to maintain or record data from them and finally, willing to use the ICARUS portal for data gathering and syncing. Through the collaboration of these sensors advanced analysis of exposure data can be conducted.

3.3 Policy management

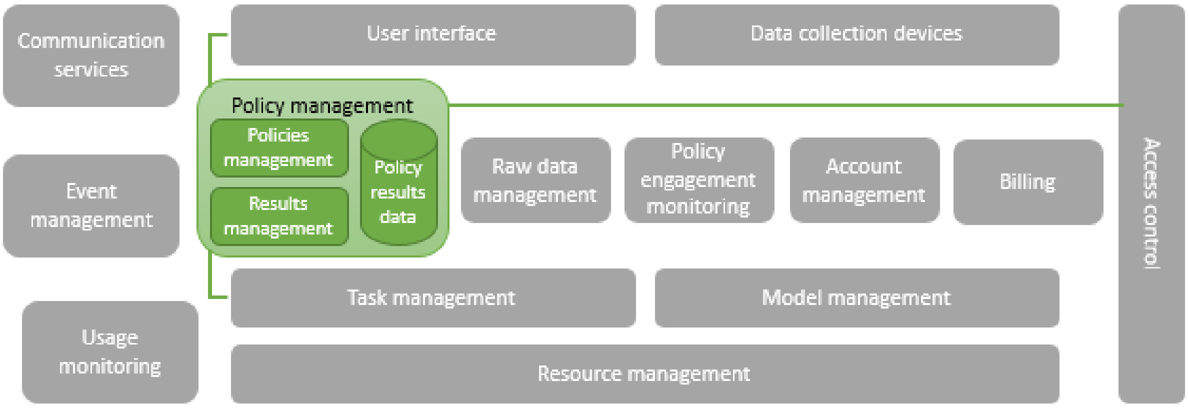

Figure 9 - Policy management component diagram

The Policy management module is responsible for the definition, management and storing of the policy scenarios created by the users, as well as persisting the results of their execution. It interacts with the User interface via the Access control module and with the Task manager. Since the geographical entities

The **Policies management** submodule handles the execution of the predefined scenarios and the definition of new ones. In the former case, the user will be able to parameterize them so that they better correspond to the status of their geographical jurisdiction. In the latter case, the user can create a new policy from scratch, by defining measures and other parameters, using the WebGIS component. Each policy that is created will be persisted so that it can be later executed and so that the user can keep track of its application in the area.

The **Results management** submodule undertakes the task of receiving, storing and giving access to the geodata produced by the execution of the model simulations.

3.4 Raw data management

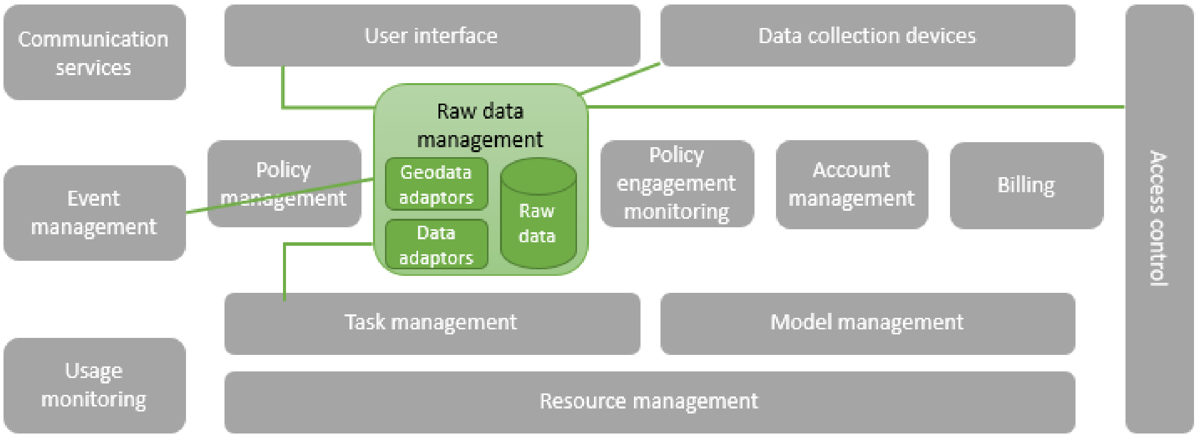

Figure 10 - Raw data management component diagram

Raw data is data[5] that has not been processed for use. Decision making for air quality and climate change adaptation policy-making, planning and management in urban systems relies largely on processed information, not raw data. Data have to be interpreted before they can be utilized. The volume of raw primary data is often very large, and so can only be used effectively if held and managed in a dedicated component of the DSS. Furthermore, the raw data management component covers several user requirements. Its importance lies in addressing the need of stakeholders to import their own raw data for their geographical region (partly covering the user requirements "UR_F8: The DSS must allow inputting new emission measurements and integrate it with the existing data available to the DSS" and "UR_F9: The DSS must allow inputting new pollutant concentrations and integrate it with the existing data available to the DSS") and the need of real-time or near real-time data collection from devices (covering the user requirement UR_F11: The DSS must allow collecting data from the user-centric tools)

The functions of the component for the management of raw data can be summarized as follows:

- store and secure the raw data;
- allow easy access to raw data;
- ensure the validity of the data;
- ensure data conform to EU and international standards;
- ensure data integrity and internal consistency;

[5] http://searchdatamanagement.techtarget.com/definition/data

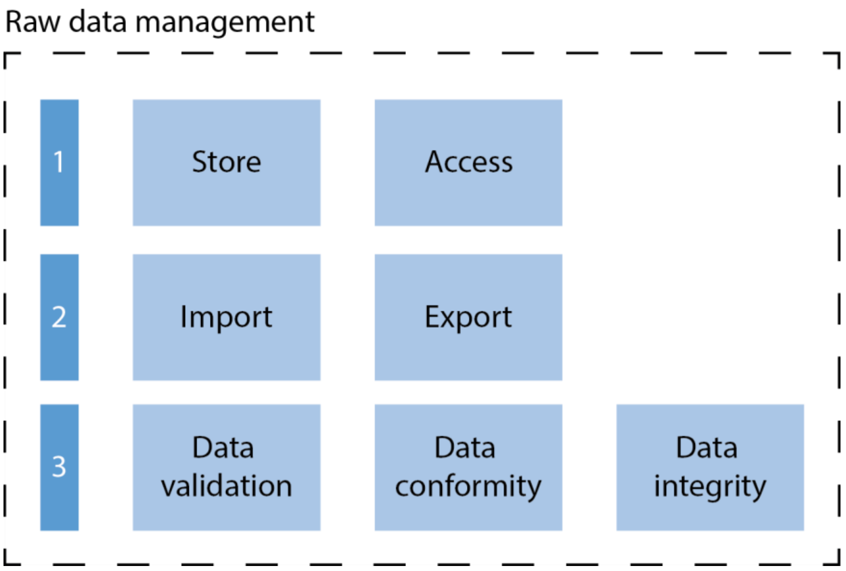

Figure 11 - Functions of Raw data management module

The basic principle is to store all data in their primary form. This allows flexibility in processing and ensures that computations are reproduced from source data incorporating all stages of analysis. Taking into account the vast investment in data collection in contrast to the low costs of storage utilities, it is considered essential to hold data in its primary form.

Raw data must be stored securely, but made easily available for analysis. The GeoDatabase will store the original raw data. The raw data management component will be integrated with the data collection devices. As mentioned in the beginning of the subchapter, the component will not only provide the possibility to store and archive raw data in numerous formats in the GeoDatabase and allow users to import their own raw data for their geographical region, but also to save and archive raw data from collection devices (wearables, smartphones and IoT devices, sensor/station networks) over a long period of time, including the possibility to open and visualize raw data and metadata. Raw data will be visualized and reviewed in detail in a built-in viewer without having the need to load these data into other software.

Data entry will integrate import functions and validation controls. An Import function should be available to incorporate data commonly held in various formats (e.g. excel files, csv, spreadsheets, shp, ASCII files, txt, etc.). This function will ensure data integrity and quality is maintained. The system will be able to incorporate historical data using methods of bulk data conversion. Also, retrieval of data from collection devices will be facilitated by developing structures or links. Again, special attention will be given to maintaining data integrity and data validation.

Data validation will be implemented at two levels, namely data collection and data entry/import to the raw data management component. Data entry user-interfaces will be structured to enforce sets of rules applied to validate inputs.

Following the search of raw data, an export function will also be provided. Features of this function include:

- Selection of file format to be exported
- Detailed summary of exported data and types

Last but not least, access will be controlled to ensure database integrity and confidentiality).

3.5 Policy engagement monitoring

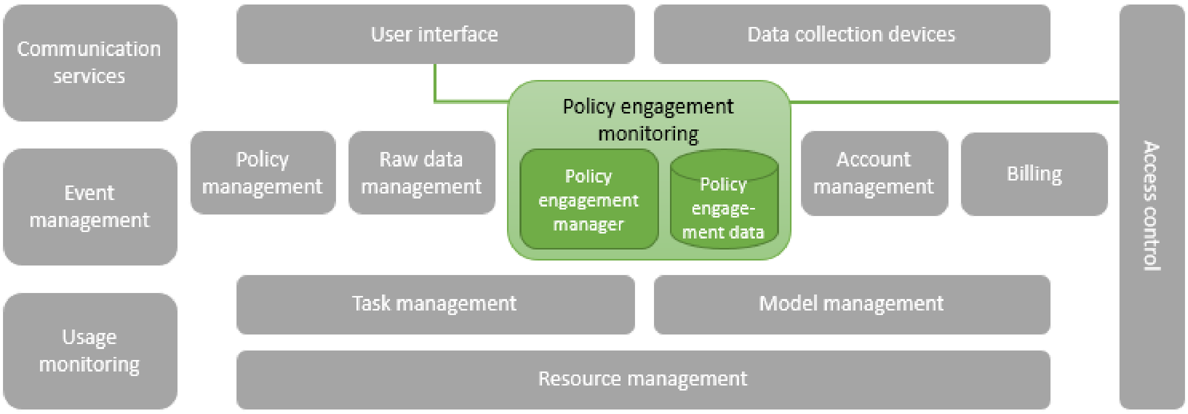

Figure 12 - Policy engagement monitoring component diagram

The policy engagement monitoring module is responsible for enabling the user keep track of the policy implementation status. I.e. it can persist information about which measures of a policy already created and simulated in the DSS have been already implemented and to what extent. It enables the user to update the implementation status from time to time. By storing policy engagement data, the user can have a clear view of the efficiency of the measures taken by the authorities with higher accuracy, since the rate of materialization is taken into consideration.

The policy engagement monitoring module interacts with the user through the User interface component and access to it is supervised by the Access controller.

3.6 Account management

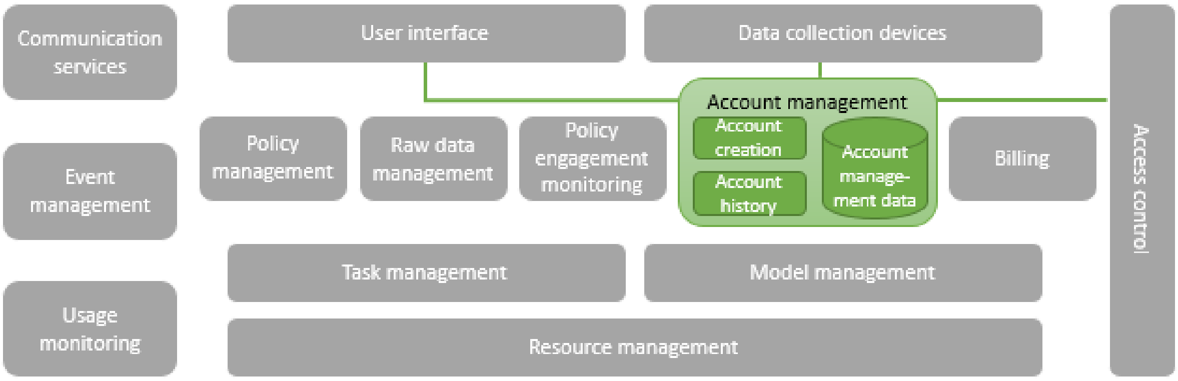

Figure 13 - Account management component diagram

The Account management module manages all the information and tasks related to the accounts of the DSS users, the affiliated data collection devices and the geographical entities. It has two submodules, the Account creation and the Account history, as well as a database to store the relative data. It interacts regularly with the Access control component, to provide information about account privileges and log-in status, and with the User interface, in order to allow management of accounts and geographical entities.

The **Account creation** submodule provides functionality for creating, modifying and deleting accounts for users and for data collection devices. It also enables the master administrator to define and manage geographical entities and assign users to them. It stores all necessary information in the Account management database.

The **Account history** submodule is responsible for saving and retrieving all information relative to account activity, such as log-in attempts and authentication tokens.

3.7 Billing

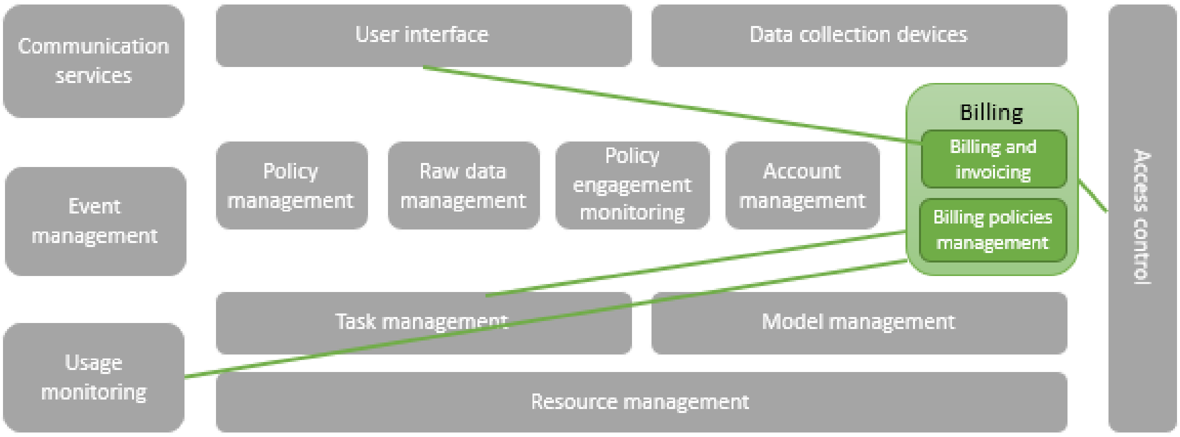

Figure 14 - Billing component diagram

The Billing component is responsible of managing the billing policies, creating invoices based on the usage and check whether the billing policy of a client allows the provision of sufficient resources in order to execute a policy simulation. The Billing component has two submodules, the Billing and invoicing and the Billing policies management. Many different billing policies can be created, based either on a fixed price limited or unlimited plan, as well as pay-as-you-go plans.

The **Billing policies management** submodule allows the master administrator and the accountant roles to create, modify billing policies and assign them to geographical entities. It also retrieves which billing policy corresponds to each geographical entity.

The **Billing and invoicing module**, on the other hand, allows the master and local administrators as well as the accountants to view billing information for geographical entities and to create invoices. Invoices might be calculated in collaboration with the Usage monitoring component, which keeps track of the resources used by each client, if the last has opted for a pay-as-you-go billing policy.

3.8 Task management

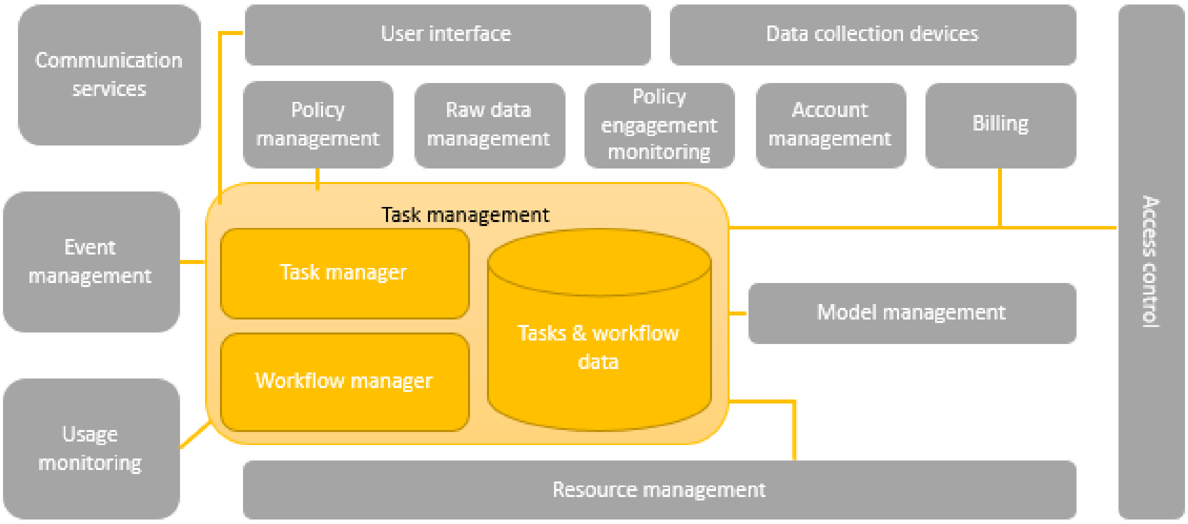

Figure 15 - Task management component diagram

The Task management module is the component that coordinates all computational activity in the DSS. Each time a simulation needs to be run, it is the **Task manager** submodule that will be invoked by the User interface. It will retrieve the information about the policy scenario that will be simulated (by the Policy management component), communicate with the Model management module and the Raw data management module so that it can collect the required algorithms and data that need to be executed and processed for the simulation. The Task manager will also “dry run” the simulation, in order to estimate the resources that need be allocated. Before execution, it must also check whether the client that requested the simulation has access to the needed resources for the execution; this is done by querying the Billing module. Finally, the Task manager will split the simulation into tasks, which will be passed to the Workflow manager submodule for execution. The Task manager keeps track of the progress of each task and when all tasks of a simulation have been executed, it will notify the user and store the data produced in the policy management geodatabase.

The **Workflow manager** submodule maintains the execution queue and asks the resource management for available resources. It sends the code to be executed and the data to be processed to the respective VM, gets notified each time a task completes execution and also notifies the Task manager on the event.

As presented in the diagram, the Task management module interacts with following modules:

- the User interface,
- the Policy management,
- the Model management,

- the Event management,
- the Usage monitoring,
- the Billing and
- the Resource management.

Each call to the Task manager by the User interface has to be authorized by the Access controller.

3.9 Model management

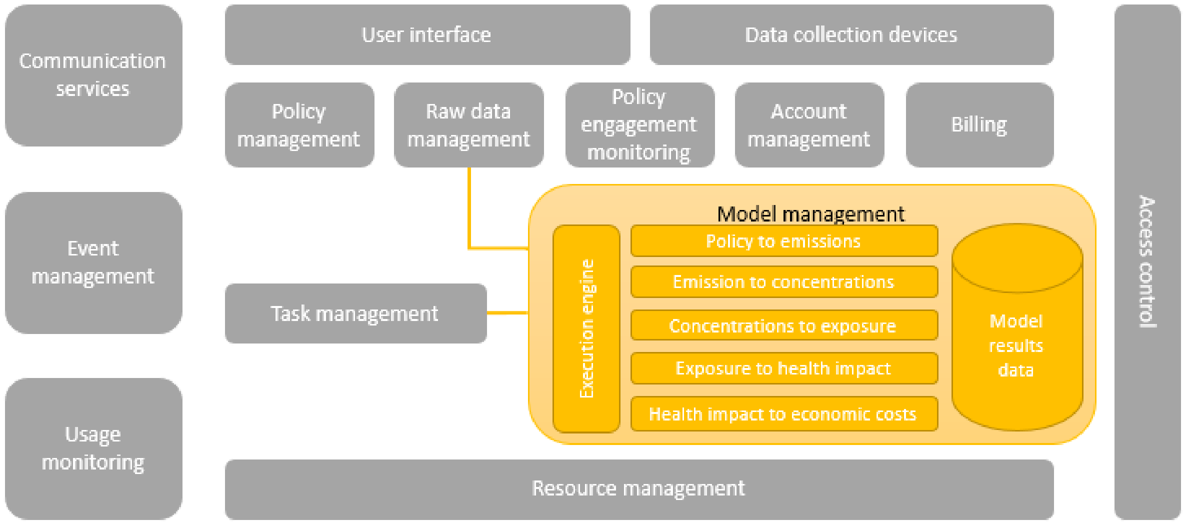

Figure 16 - Model management component diagram

In the Model management module lies the logic core of the DSS. This module will select and parameterize the appropriate models that have to be invoked for the execution of a policy simulation and feed them with the necessary data. The Model management module is also responsible for storing and giving access to the geodata on which the models rely for execution. It interacts with the Raw data management and the Task management modules.

The **execution engine** submodule is responsible of analyzing the parameters of the policy scenario that will be simulated and invoke the correct models and pass them for execution to the Task manager in the correct order. It also undertakes the task of retrieving the necessary data for the simulation, whether these data are model results data or raw data that have been collected for the specific geographical entity.

The **Model results data** are stored in the geodatabase of ICARUS DSS. With this term we refer to the data that will be produced by the initial execution of the climatic and other models for 2020, 2030 and 2050.

3.9.1 ICARUS DSS Computational system

The ICARUS computational system is the computation core of the ICARUS DSS. It encompasses a series of sub modules along the full chain from policy to economic evaluation.

The workflow of the ICARUS DSS computation system module is reported in the following figure:

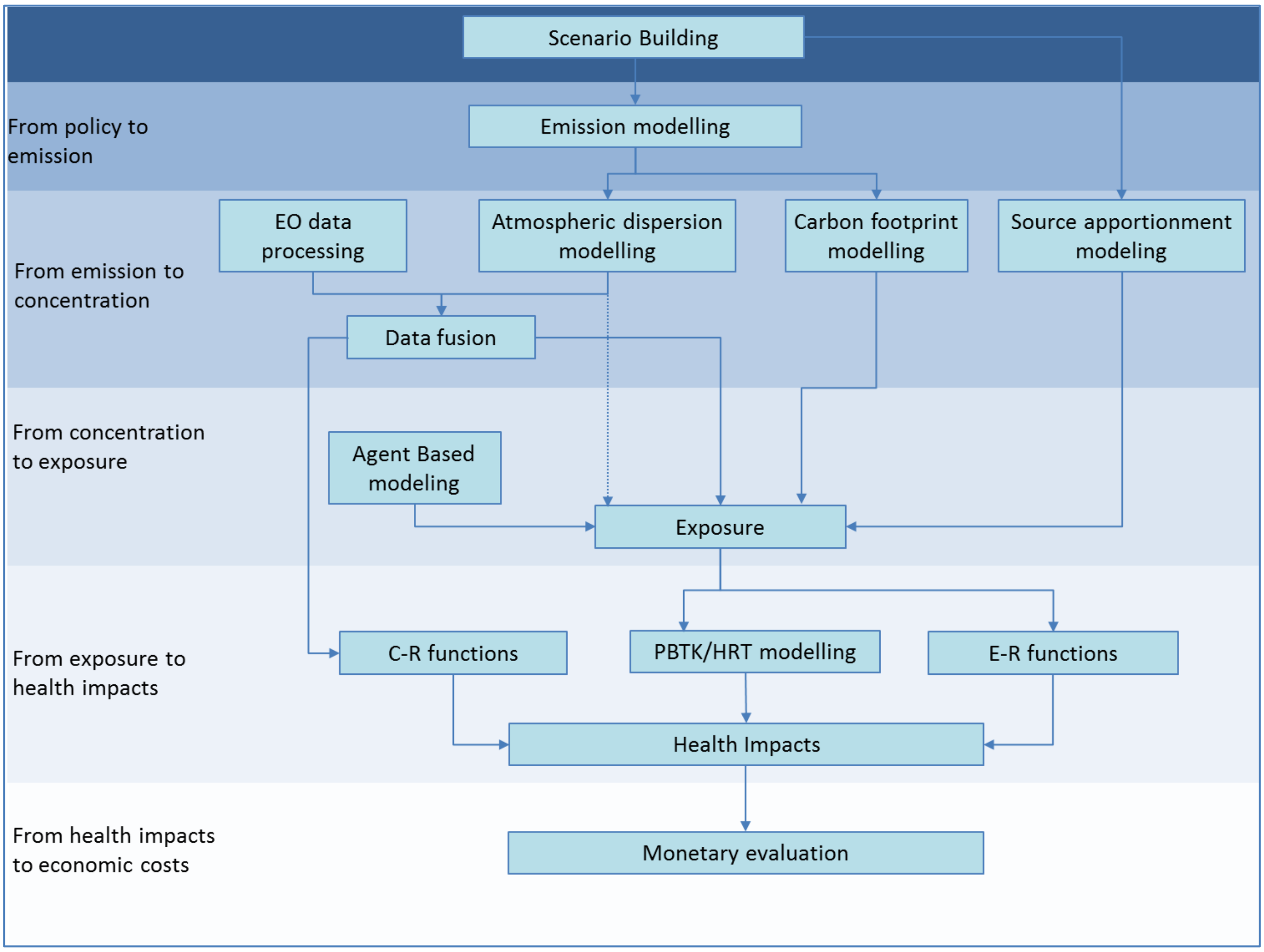
Scenario Building
From policy to emission
Emission modelling
From emission to concentration
EO data processing
Atmospheric dispersion modelling
Carbon footprint modelling
Source apportionment modeling
Data fusion
From concentration to exposure
Agent Based modeling
Exposure
From exposure to health impacts
C-R functions
PBTK/HRT modelling
E-R functions
Health Impacts
From health impacts to economic costs
Monetary evaluation

Figure 17 - Schematic representation of the workflow of the ICARUS DSS computational module

3.9.1.1 Scenario building

The scenario building module is the entry point of the ICARUS DSS used to define or select an initial policy and/or technological scenario which will be finally assessed throughout the full chain in terms of impacts onto receptors like human health, ecosystems and economic costs. It represents the very first step of the modeling chain where the users can select an already pre-defined scenario (e.g. Business As Usual- BAU) or build a new one. Some pre-defined scenarios will be already loaded giving the possibility to the user to modify them.

Since one of the major objective of ICARUS is the assessment of environment and health impacts as well as the associated cost benefit analysis of policy measures this module provides the user the possibility to select or to create a specific policy scenario from which the next module may calculate the relevant emission values.

Main input: users have to define policy options or technological measures leading to a change of the emission of pollutants into the atmosphere (e.g. change in vehicle fleet composition)

Main output: " a user defined" scenario to be used in the next step to estimate the relevant missions.

3.9.1.2 Emission modeling module

The final goal of this module is to generate emission scenario to be used as input in the successive atmospheric dispersion modeling module.

It uses all the emission relevant data stored in the ICARUS GeoDb (i.e. activity-emission factor matrices) to produce a comprehensive emission scenario for the selected pollutant.

Since the models in the atmospheric dispersion modeling module usually need emission data at high spatial and temporal resolution, the emission data generated by the application of specific policy measures has to be spatially disaggregated to satisfy the spatial format. To this objective this module will also include spatial disaggregation tools in order to resolve spatially the emission data on a requested grid.

The module offers the following functionalities to the user:

- Select the stressor for which the user wants to calculate the emission scenario
- Select the scenario: baseline or specific policy scenarios (as defined in the previous module)
- Select the spatial scale for the spatial disaggregation of emission scenario.

Main input: " a user defined" scenario representing some policy and/or technological measures

Main output: an emission inventory for selected air pollutants which incorporates the policy options and/or technological measures defined in the previous module.

3.9.1.3 Atmospheric dispersion modeling module

This module allows the user to perform the calculations necessary for the estimations of primary and secondary pollutants concentrations in the domain of interest both horizontally and vertically. The necessary input for these calculations is taken from the geo-referenced database of ICARUS DSS. Such input includes meteorological data (including horizontal and vertical wind profiles), digital elevation models of the modelling terrain, emissions inventory (in grid format) derived from the Emission modeling module and (if needed) initial concentrations of primary pollutants. This module encompasses atmospheric dispersion model such as WRF-Chem and/or CMAQ as well as other less sophisticated publicly available Gaussian plume models used widely by regulatory agencies such as CTDM+ or AERMOD. Other atmospheric modelling software, which may be available at the end-user site, can be readily incorporated to the SMAQ platform through an open-ended I/O architecture.

The module offers the following functionalities to the user:

- Select the stressor for which the user wants to calculate the atmospheric concentration levels.
- Select the emission scenario (obtained from emission modeling module)
- Select the spatial domain for the model simulation.

Main input: emission scenario associated to policy and/or technological measures as defined in scenario building module; meteorological/climate scenario; digital elevation model; land cover - land use

Main output: a set of estimated geo-referenced atmospheric data including the horizontal (and vertical if needed) concentrations of primary and secondary pollutants.

3.9.1.4 Earth Observation (EO) data processing

This module enables the calculation of the Aerosol Optical Depth (AOD) values by means of processing high and high or moderate resolution satellite images (e.g. Landsat, Spot, MODIS).

This module offers the following functionalities to the user:

a) ***Radiometric Correction****:* Performs the radiometric correction of the satellite data by reading all the required parameters i.e. gain and offset values, sun elevation angle, day of the year etc. from the leader files of the images.

b) ***AOD Calculation****:* Performs the AOD calculation by applying the appropriate algorithm to the radiometrically corrected satellite images. This process is taking place after the radiometric correction of a new image, without the interaction of the end-user.

Main input: Raw satellite images

Main output: Aerosol Optical Depth (AOD) maps derived from the satellite image selected

3.9.1.5 Data Fusion

This module ***enables*** the integration and fusion of the information coming by ground measurements, air quality models and satellite images in order to calculate: a) scattering coefficient of tropospheric aerosol; b) pollutant concentrations based on assimilation of scattering coefficient estimated from satellite imagery and AQ ground stations measurements; and c) pollutant concentrations based on the “fusion” of Earth Observation (EO)data with atmospheric dispersion modelling results.

This module offers to the user the following functionalities:

a) ***Scattering coefficient calculation:*** Performs the calculation of the scattering coefficient using the AOD values from the AOD calculator and the mixing layer height generated via meteorological models and/ or measurements.

b) ***Pollutant estimation based on data fusion:*** performs the calculation of pollutant concentrations based on the scattering coefficient derived by the previous process.

c) ***Pollutant estimation based on model fusion:*** performs the calculation of pollutant concentrations fusing the results of the data fusion model and the results of atmospheric pollution transport and transformation models.

Main input: AQ data from i) ground measurements, ii) air quality models results and iii) Aerosol Optical Depth (AOD)

Main output: concentration field at the ground level of the pollutant selected derived from the fusion of data from monitoring network stations, AQ modeling results and Earth Observation information.

3.9.1.6 Carbon footprint model module

This module allows the calculation of the total amount of GHGs emission directly caused by an activity which may be associated to policy measures or to human behavior. Refined footprint estimation will be carried out through the latest edition of the CoolClimate Calculator developed by the Institute of the Environment (BIE) and the Renewable and Appropriate Energy Laboratory (RAEL) of the University of California, Berkeley. The calculator shows how different lifestyles contribute to global warming and allows individuals, households, small businesses and even cities to estimate the amount of CO2 they generate annually based on their mode of transportation and their food, housing and lifestyle choices. Using CoolClimate we shall identify areas where the C footprint can be reduced significantly and compare the results to the footprints of similar households in different large urban areas. In ICARUS the CoolClimate Calculator will be adapted to reflect the European dimension to provide residents with more precise information that is based, for example, on available energy sources and modes of public transportation, and local food and energy prices. The module will initially estimate carbon footprint in three areas: home energy, transportation and waste and will be expanded to include other sectors.

Main input: policy measures identified in the scenario building module; human behaviors daily activities.

Main output: carbon footprint estimation associated to the policy measures identified in the scenario building module and/or to change in human behavior.

3.9.1.7 Source apportionment modeling module

This module provides different modeling tools to estimate the contribution of an emission source or emission source type to the overall atmospheric pollution burden. Source apportionment (receptor) models are mathematical procedures for identifying and quantifying the sources of ambient air pollutants and their effects at a site (the receptor), primarily on the basis of species concentration measurements at the receptor, and generally without need of emissions and meteorological data.

In the ICARUS DSS we will make use of a tiered modeling approach based on the use of models with different levels of complexity. The tiered use of different models, starting from analyses of routinely available data and gaining higher complexity with increasing data availability and information, will deliver a complex data base with far reaching information about the sources causing ambient air pollution.

This will eventually lead to the development relationships between changes in emissions of certain emission sources and the change in concentrations based on the source apportionment results. These relationships will be implemented in the ICARUS DSS so as the policy makers can estimate concentration changes caused by policy/measure combinations without having to rely on atmospheric modeling.

Main input: concentration field at the ground level of the pollutant selected, chemical fingerprints of the sources (source profiles) [a database of atmospheric particulate matter emission source profiles in Europe (SPECIEUROPE) developed in the framework of FAIRMODE, Working Group 3 is accessible at the website http://source-apportionment.jrc.ec.europa.eu/Specieurope/index.aspx).

Main output: Contribution of different emissions sources types to the overall atmospheric pollution burden.

3.9.1.8 Agent Based Modeling module

The main functionality of the Agent Based Modelling (ABM) module is to estimate time-activity patterns of representative subgroups of population, taking into account societal dynamics factors modulating the overall results of different policy actions. The module will be based on stochastic, a city level ABM that can feed into a population-based assessment without imposing prior bias, but rather basing its estimations onto emerging properties of the behavior of the agents, computerized autonomous decision makers that compose the modelled city.

The main inputs of this module are consisted of the city's population and sociodemographic data, as well as the road and buildings networks of the region under investigation. Data is transformed into human, road and building agents respectively that act and react in a city-environment, by following rules based on their individual attributes. Survey outputs and literature findings with data on lifestyle/behavioral patterns are associated with human agent behavioral rules, aiming to model representative to real world conditions. By modelling agents individually, the full effects of the diversity that exists among their attributes and behaviors can be observed as it gives rise to the behavior of the system as a whole. Different human agents based on different age, gender or SES status, will express different behaviors. At the end of a model's run, spatiotemporal behavior of individual or groups of human agents is captured, in a GIS format, as an outcome of the prevalence of specific preferences and decision-making.

This approach enables a better understanding of the behavior of individuals and populations in social and evolutionary settings and permits the computationally cost-effective identification of refined time-activity diaries throughout the day

Main input: Population and sociodemographic tables, road and building networks in GIS format, utility functions that define human spatiotemporal movement.

Main output: The ABM derived spatio-temporal behaviors at an individual or population level will be then used in the Exposure modelling module to derive exposure profiles. ABM can be used as a means for estimating and comparing the probable effects of different public health strategies prior to implementation.

3.9.1.9 Exposure modeling module

The main functionality of the exposure module is to quantify the population exposure levels for a selected chemical starting from the concentration levels calculated in the "Atmospheric dispersion modeling" module.

This module includes a further sub-module "human respiratory tract deposition mode (HRT)l" to estimate the internal doses of chemicals in in the respiratory tract

The main inputs of this module are:

- Concentration data (derived from the "atmospheric dispersion modeling" module).
- The population target group (e.g. gender, age and socio-economic categories)

- Time activity pattern for the population target group selected
- Behavioral information (e.g. activities during the day) for the population target group selected derived from the ABM module

The module offers the following functionalities to the user:

- Select the stressor(s) for which the user wants to perform a simulation
- Select the spatial scale
- Select the specific population target group
- Select the exposure model or human respiratory tract deposition (HRT) model.

Depending on the model selected the system will send the input file(s) to the exposure model(s) in a format readable by that model and the user will be directed to its execution phase.

Main input: concentration field at the ground level of the pollutant selected

Main output: Exposure estimates (external or internal) for selected population groups

3.9.1.10 Health impact assessment modeling module

The main functionality of the health impact assessment module is to quantify the impact on human health for different target groups due to the exposure levels calculated in the “Exposure” module and/or the concentration level deriving from the “Atmospheric dispersion modeling”” module.

The impact on human health will be quantified integrating population density data with exposure-response functions and/or concentration - response functions (both derived from epidemiological studies and stored in the ICARUS database).

For some specific chemicals that will be flagged in the database, the platform should allow the application of with Biology-based Dose Response (BBDR) models in order to derive quantitative estimate of the impact on human health.

The input of this module will be represented by the exposure levels and/or concentration values and the output will be the risk posed by them to human health.

The main inputs to the models implemented in this module are:

- Concentration levels (from atmospheric dispersion modeling module) or
- Exposure levels (from exposure modeling module)
- Population target groups (e.g. gender, age and socio-economic categories)
- Concentration or exposure-response functions (e.g. the ones established by WHO in the HRAPIE project)

This module offers the following functionalities to the user:

- Select the stressor(s) for which the user wants to perform a simulation
- Select the spatial scale
- Select the specific target groups

- Select the model (i.e. exposure-response or concentration-response or Biology-based Dose Response (BBDR) model).

Depending on the model selected the system will send the input file(s) to the model in a format readable by that model and the user will be directed to its execution phase.

Main input: Exposure estimates for selected population groups

Main output: Quantitative estimates of health impacts for selected population groups in term of mortality and/or morbidity

3.9.1.11 Monetary evaluation modeling module

This module represents the end-point of the ICARUS computational system module. Its main functionality is to quantify economic cost or benefit associated to specific policy scenarios trough a Cost Benefit Analysis approach

For the cost-benefit analysis all benefits, damages and costs will be calculated and the non-monetary (intangible) items will be transformed into monetary values, where possible including:

- Monetary values for all processed health endpoints (derived from literature) using appropriate "value transfer" methods from contingent valuation and/or cost of illness studies. The input will be the quantitative assessment of the specific health endpoint considered (e.g. lung cancer, leukemia, neurodevelopmental disorders, etc.) calculated by the "Health Impact assessment" module in terms of change of disease prevalence due to the policy measure considered.
- Societal costs, e.g. costs for the source operator and the state, minus income and savings (e.g. state income from taxation).
- Monetized contributions to climate change through the evaluation of the marginal avoidance costs (MAC) and marginal costs per t of $CO_{2,eq}$.
- Utility gains and losses

Main input: Quantitative estimates of health impacts for selected population groups in term of mortality and/or morbidity

Main output: Quantitative estimates of economic costs of the policy options and/or technological measures selected at the beginning of the process

3.10 Resource management

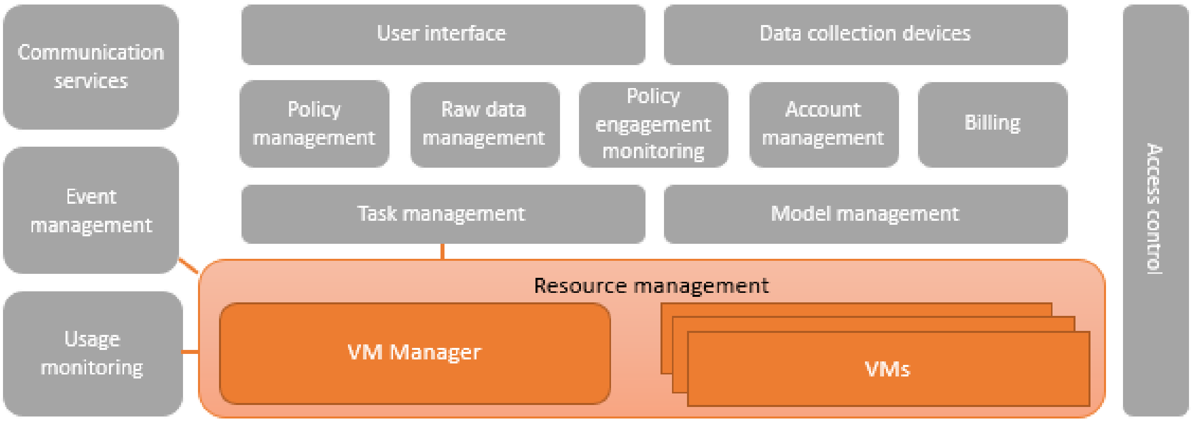

Figure 18 - Resource management component diagram

The Resource management module is responsible for allocating computational and memory resources to the Task management. The **VM Manager** receives the tasks that have to be executed and the amount of resources that can be dedicated to them. It subsequently allocates the appropriate **Virtual machines**, passes the code to be run and the data to be processed to them and monitors the execution process. Upon completion, it sends information about the resources used by the task to the Usage monitoring module and an event to the Event management module, so that the workflow manager can be notified about the completion of the task and receive a reference to the results data.

The resource management module interacts with the Event manager, the Task manager and the Usage monitoring module.

3.11 Communication services

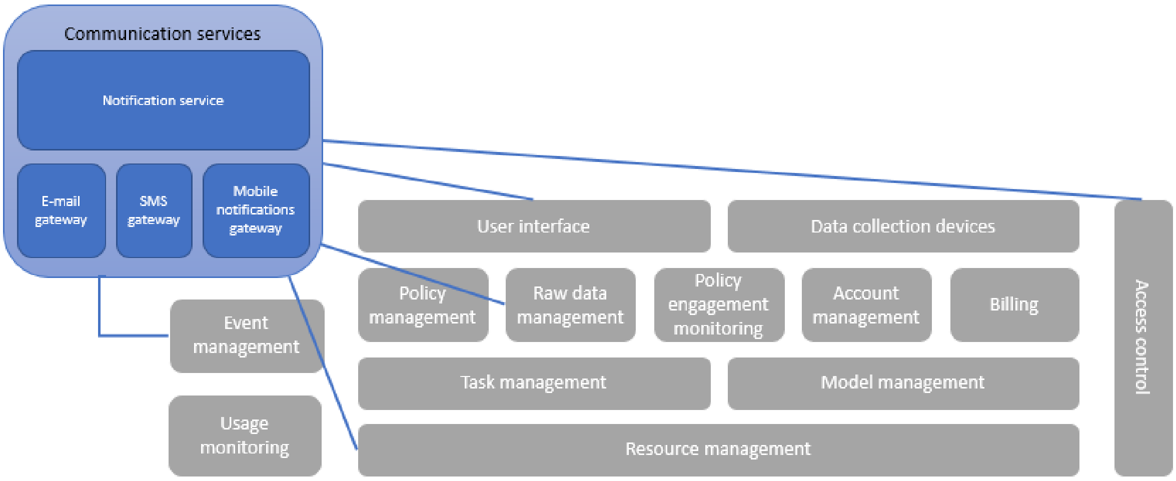

Figure 19 - Communication services component diagram

The communication service component is responsible for transmitting to the user any type of notifications and other information needed. This includes:

- Connection tokens
- Reset password links
- Account activation confirmation
- Invoicing
- Notifications when a requested simulation has ended and the results are available
- Notifications about account activity (log-in events, password resets etc.)
- Notifications to the mobile application about air quality limits

The communication services can reach users via e-mail, SMS and mobile application notifications, by using the respective gateways, as seen in the diagram. The notification service submodule receives the messages that need to be forwarded along with the account information and determines which communication channel must be used, based on the user's preferences.

This component interacts with the Event management (to send notifications on air quality and simulation completion), the Access control (for communicating account events and connection tokens) and directly with the User interface.

3.12 Event management

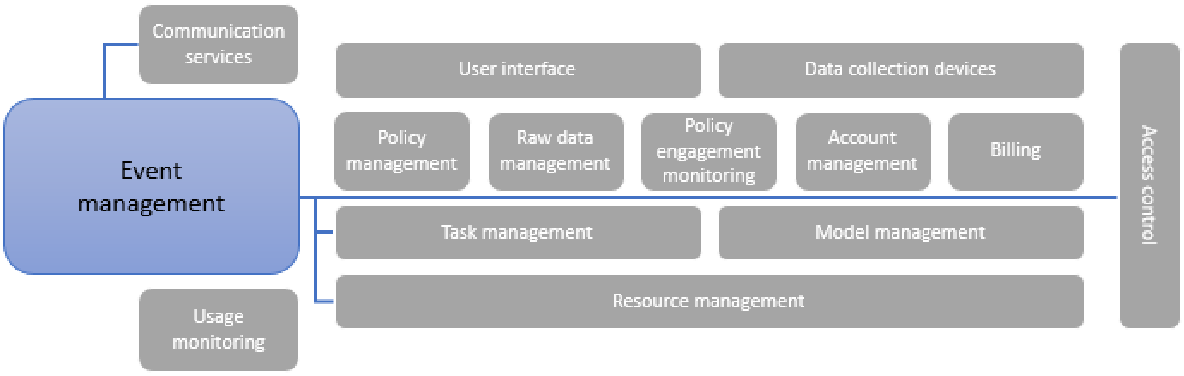

Figure 20 - Event management component diagram

The modular nature of the DSS architecture requires a robust mechanism of synchronization and information exchange among its numerous modules. Events will be created in several different circumstances, such as when a task completes its execution, when the results of a simulation become available to the user, when an account is accessed through the user interface or by the data collection devices and each time measurements are received from the last.

To address this need, a dedicated Event management module is foreseen. It will interact with the following modules:

- Task management,
- Access control
- Resource management
- Communication services

Interaction with the communication services takes place each time information on the event has to be forwarded to the user(s).

3.13 Usage monitoring

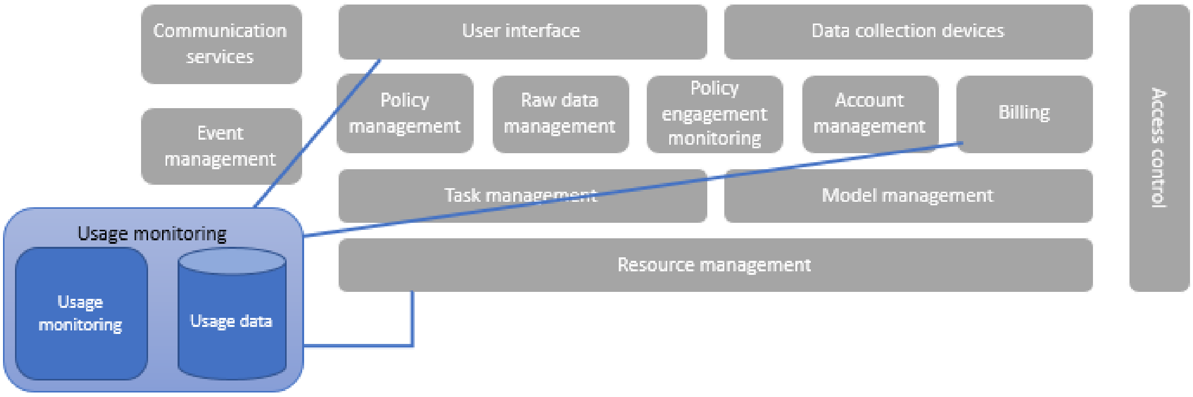

Figure 21 - Usage monitoring component diagram

Since usage of ICARUS DSS will be offered as a paid service, a usage monitoring module is indispensable, so that billing can be based on it. The Usage monitoring module receives and persists usage information from the resource management module. It provides usage data to the User interface module when the master administrator wants to access them. It also provides usage data to the Billing module on two different occasions: when an invoice must be issued (or billing information must be presented to an Accountant or a Local administrator role) and when the Task management needs to determine whether the billing policy of a client provides them with enough resources to run a pending simulation.

3.14 Access control

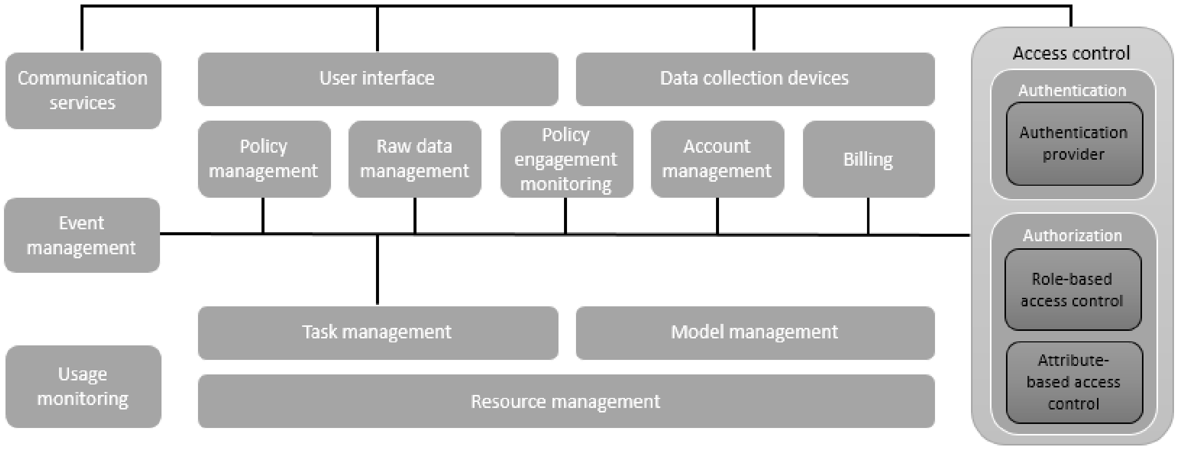

Figure 22 - Access control component diagram

The access control component is responsible for providing a connection mechanism for the users and for safeguarding the ICARUS DSS from unauthorized access. It consists of two submodules, one for each of the aforementioned tasks. It interacts with almost all the other components of the DSS, since each request originated by the user interface or the data collection devices will pass through it.

3.14.1 Authentication

The authentication submodule undertakes the task of authenticating each user or data collection device that attempts to connect to the DSS. To do so, it sends the log-in credentials (which are collected either by the user interface or directly sent by a data collection device) to the account management and retrieves the respective account information. Not all accounts have the same authentication schema, since some of them (administrators, accountants) require higher security levels. The authentication provider will proceed to the predefined authentication method, based on the information received. Two authentication methods are currently foreseen for the DSS:

- Authentication based on username and password
- Two-factor authentication

In the former case, the authentication provider will only create an authentication token. The authentication token will be communicated to the user/data collection device, so that it can be sent together with any subsequent request and provide seamless authentication, and to the account management component, so that it can be stored in the accounts active tokens.

In the latter case, one more step is required for authenticated the user, which will be executed through a communication channel other than this that the user interaction takes place, i.e. whether by SMS or through a smartphone application, provided that the user is not trying to log-in from this application. In this case, the authentication provider will also retrieve account information from the account manager, but will also send a connection token (such as a short alphanumeric sequence) using the communication channel already selected by the user. The connection token is forwarded to the communication services, which send it to the user. If the connection token is sent by SMS, the user will have to copy it to the UI. If on the other hand the connection token is sent to the smartphone application, it is enough for the user to click on a button to confirm that they are indeed trying to log in, and the connection token will be retransmitted back to the authentication provider directly by the application. Upon reception of the connection token, the authentication provider verifies it and if it is correct, it proceeds in the creation, storage and communication of an authentication token, as described in the previous case.

3.14.2 Authorization

The authorization submodule undertakes the task of determining whether a user or data collection device has the privileges to perform an action. Privileges are defined by taking into consideration:

- The role the user has
- The geographical entity they belong to, if applicable

Supposing that the user (or a collection device) has already been authenticated, they submit a request to perform a specific action, accompanied by their authentication token. The authorization submodule will receive the request and send the authentication token to the account manager in order to retrieve the account information. The account information contains the privileges the owner of the account has and the geographical entity they belong to, if applicable. Based on those, the role-based access controller will determine if the user/device has the right to perform this type of action and the attribute-based access control will determine if the user/device can perform this action on the data they are trying to access/modify. E.g., the local administrator of geographical entity A wants to see billing information for their geographical entity. They send a request to view billing info for geographical entity A, the role-based access control determines that they have the privilege to see billing info for their own geographical entity, since their role is local administrator, and invoked the attribute-based access control, which in turn verifies that they are indeed local administrator of geographical entity A, for which they requested the billing information.

3.15 Traceability matrix

In the following table all components are matched with the user requirements they address.

UR_FCD	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23
3.1		X	X	X	X			X	X				X		X	X			X				
3.2											X												X
3.3	X	X					X											X					
3.4								X	X		X												
3.5						X																	
3.6													X										
3.7														X									
3.8														X									
3.9								X	X											X			
3.10														X									
3.11												X											
3.12										X										X	X	X	X
3.13														X									
3.14																	X						

Table 4 - Traceability matrix

4 Collaboration diagrams

In this chapter, the interactions and the information flow among the different components of the DSS is analyzed. To this end, we examine several scenarios of users' and other actors' interaction with the DSS, that cover the most representative user requirements.

4.1 Two-factor authentication

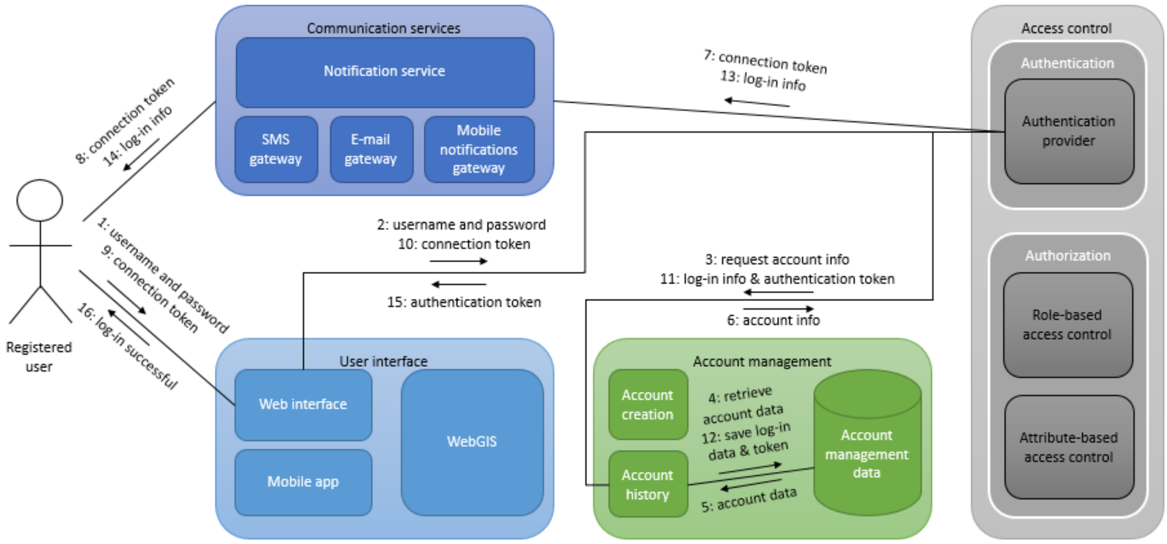

Figure 23 - Two-factor authentication collaboration diagram

In order to ensure the security of the most important account types, such as those of the master and the local administrators, a two-factor authentication protocol will be implemented. In the current scenario, we examine the case where a registered user initiates the procedure using the web interface. They will introduce their username and password to the web interface and submit the log-in form. The DSS will then forward a connection token through the user's preferred channel (SMS, mobile application). The user will type this token in the web interface and then log in to the system.

Prerequisites/assumptions:

- The user is already registered and the two-factor authentication functionality is available for their profile type.
- The user has already defined what is the preferred channel for the two-factor authentication functionality (SMS, mobile application)

As seen in the diagram, the user has first to use the web interface in order to type in their credentials (username and password) [1]. Then the web interface will send them using SSL encryption to the authentication sub-component of the access control [2]. The authentication providers will request the account information from the account management module [3]. The account history submodule will query the account management database using the username and the hashed value of the password [4] and receive the necessary data that are related to the account of the user [5], which will pass to the authentication providers [6]. The latter, based on the account information, will determine that this user account needs to be authenticated using the two-factor authentication protocol and the selected channel (SMS, mobile application) to be used for this purpose. A connection token, such as a short string containing letters and numbers will be forwarded to the communication services [7], along with information on the communication channel to be used. Depending on that, the connection token will be forwarded to the user either by SMS or to their mobile application through the respective gateway [8]. The user will then input the connection token to the web interface [9], which will be forwarded to the authentication provider for validation [10]. Assuming the connection token is the correct one, the authentication provider will create an authentication token.

The authentication token will be forwarded to the account management component, together with information on the current log-in event (e.g. time, device, possibly IP address) [11]. The account history will save these data in the account management database [12]. The information on the log-in event can be also forwarded to the communication services [13], so that the user gets informed about the log-in event [14] and can take action in case this was not done by themselves. Finally, the authentication token is also transmitted to the web interface [15], so that it can be used along with any other request to the system's components.

4.2 View predefined policy results

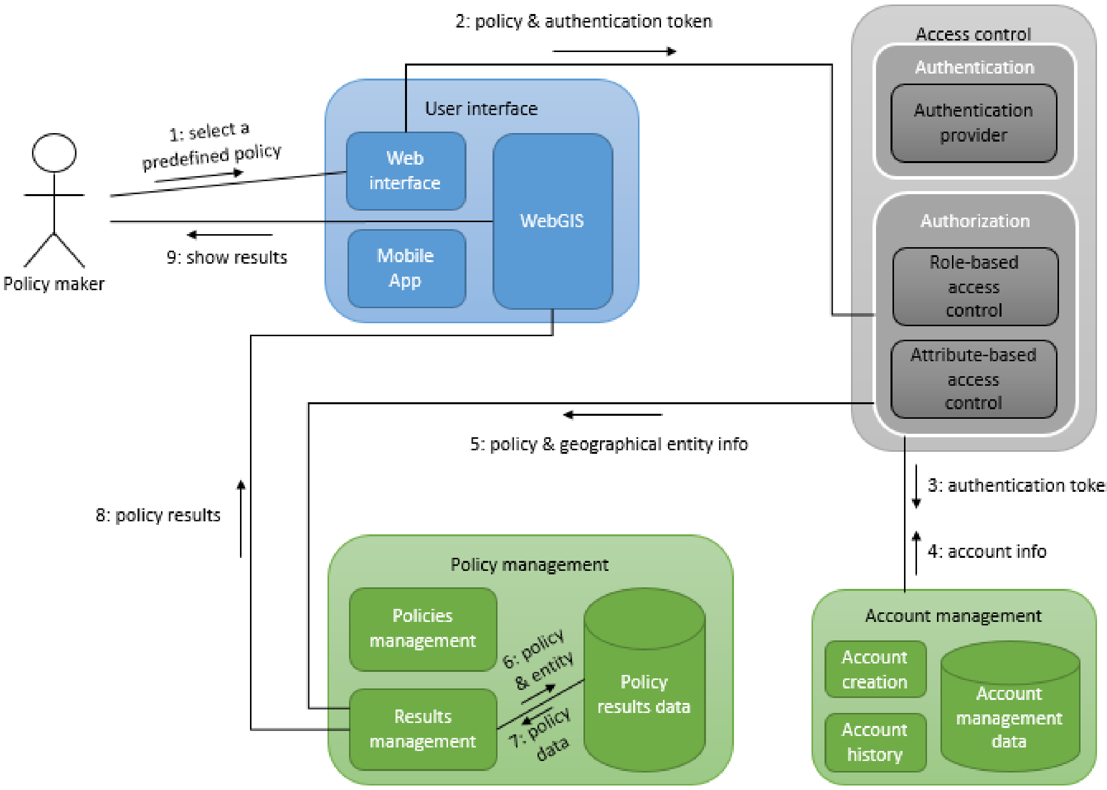

Figure 24 - View predefined policies collaboration diagram

Policy makers need to see the expected results of predefined policies and measures for their region.

Prerequisites/assumptions:

- The policy maker is already logged in to the system
- The policy maker has already been presented with a list of policies and measures

The policy maker selects one of the predefined policies and possibly parameterizes some of its measures, before submitting it [1]. The policy parameters are sent to the authorization submodule along with the authentication token [2]. The latter forwards the authentication token to the account management module [3] which retrieves the account info, including privileges, of the user account and sends it back to the authorization submodule [4]. In this case, the authorization procedure is more complicated, as the role-based and attribute-based access control need to collaborate. The former determines that the user, having the role “policy maker”, has the right to access predefined policy results. The latter then has to deduce the geographical entity for which this particular policy maker is entitled to see the results. After doing so, the authorization submodule sends the selected policy parameters together with the geographical entity info to the results management submodule [5].

The results management submodule queries the policy results database for data corresponding to the specified policy and possibly region [6] and retrieves them [7]. The policy results are then sent to the WebGIS interface [8], so that they can be presented to the user in a easy to perceive and human-friendly format [9].

4.3 Estimate policy results in emissions

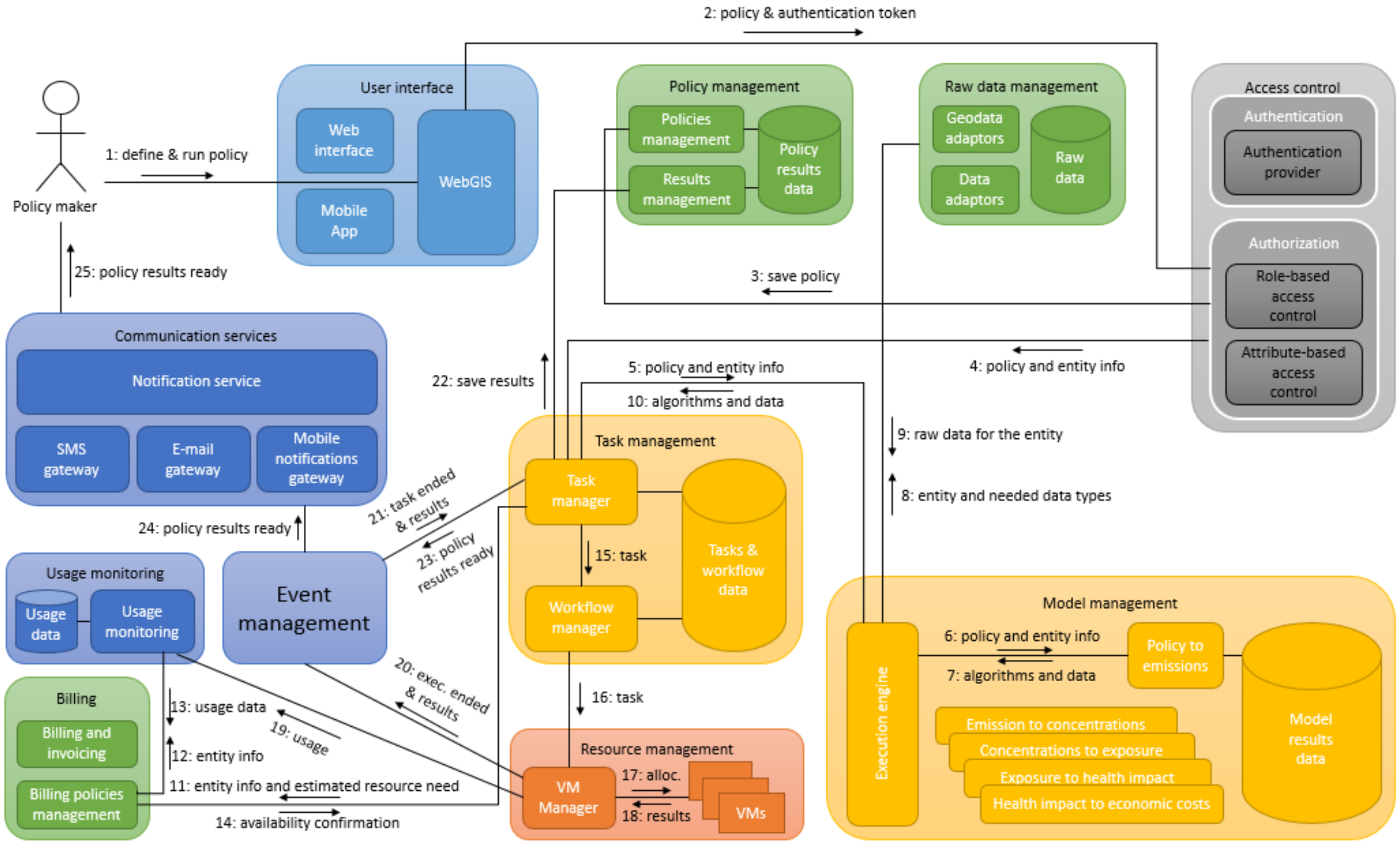

Figure 25 - Estimate policy results collaboration diagram

In this collaboration diagram is shown the way that most of the modules of the DSS work together in order to allow the user to define the parameters of and execute a computationally demanding policy simulation for their region and see its results. To do so, the DSS may have to combine existing data from the models and raw data contributed by the authorities or the user-centric tools from the geographical entity for which the simulation will be run. Model simulations like this can last from minutes to hours, or possibly days, so adequate resource management is necessary, along with an event management mechanism that will notify the user once the execution is completed and the results are available.

Prerequisites/assumptions:

- The user has the role policy maker
- The user is already logged in
- The user has already defined a policy using the WebGIS interface and wants to see its impact to the emissions of pollutants in their geographical entity
- The user has possibly chosen areas of focus, where the simulations will be of higher resolution
- There are already raw data for this area, either imported by the client (geographical entity's local administrator), or collected by the stations or the user-centric tools
- The client has enough computational resources available to perform this simulation acc's billing policy and used resources do not limit its access to computational resources

The policy maker defines in detail the policy simulation they want to run, using the WebGIS interface [1]. All the parameters and the authentication token are sent to the authorization submodule of the access controller [2], where the privileges of the user regarding this geographical entity and the attempted operation are checked, according to the process described in 4.2. Then, the policy configuration is sent to the policies management submodule [3], in order to be persisted. Then, the policy and geographical entity parameters are sent to the task manager [4], which sends them to the execution engine [5]. The execution engine will invoke the model(s) of "policy to emissions" [6], which will return the algorithms that have to be run and the data that have to be analyzed in order to simulate the policy under examination [7]. Based on the information received by the models, the execution engine will look for relevant raw data available for the specific geographical region by querying the raw data management module [8]. The latter will gather these data, and, if needed, will convert them to the appropriate format, before sending them back to the execution engine [9]. After receiving those data, the execution engine will send to the task manager the algorithms and data sets that have to be processed for the simulation.

Before proceeding to running the simulation, the task manager has to check if the client (geographical entity) is entitled to sufficient resources. To this end, it "dry runs" the simulation and determines the resources needed. It sends the resource estimation and the geographical entity info to the billing policies management submodule [11], so that it can determine if there are enough resources still available. The billing policies management requests the usage data for the specific entity [12] and receives them by the usage monitoring submodule [13]. Then, it proceeds to adding the estimated resources to the consumed and checks if the sum is over the limit defined by the billing policy. Assuming it is not, it confirms the availability of resources to the task manager [14].

The task manager can create the tasks for the simulation and pass them to the workflow management [15], so that it can proceed to assigning them to the available resources. For simplicity of the diagram, we assume that all the simulation process will be run in a single task. The workflow manager sends the task to the VM manager [16], which will allocate the necessary VMs [17] for its execution. Once the execution is completed, the VMs return the results to the VM manager [18]. Th VM manager in turn records the time and amount of resources used for the task in the usage monitoring component [19]. It then notifies the event management module that the execution has ended, along with a reference to the location of the results [20]. The event management notifies the task manager about the completion of the task [21], which in turn retrieves and persists the results data, forwarding them to the results management submodule [22].

Having assume that the whole simulation was executing in a single task, the task manager knows that the simulation is completed and all its results available in the policy management component. It can now send a notification to the event manager, stating that the simulation has been executed and the results are available for the user [23]. The event management forwards the notification to the communication services [24], which will in turn forward it to the user [25] through its preferred communication channel.

4.4 Record policy engagement

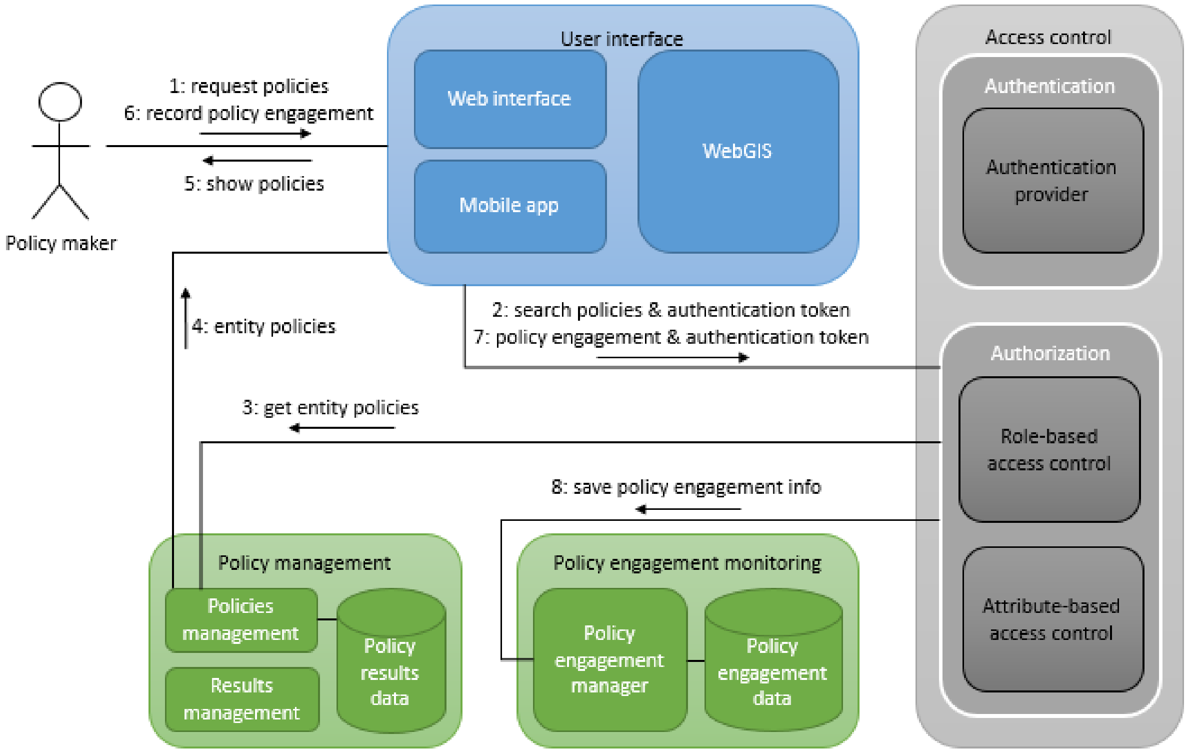

Figure 26 - Record policy engagement collaboration diagram

In this scenario, the user can select among the policies that they have already defined and keep track of the measures that have already been applied.

Prerequisites/assumptions:

- The user has the role policy maker
- The user is already logged in to the system
- There is at least one policy already defined for the geographical entity of the user

The user first wishes to see all the policies that have been created for their geographical entity [1]. The user interface sends this request along with the authentication token to the authorization submodule [2], which verifies that the user has the privilege to access the requested information as described in 4.2. The request is then forwarded to the policies management submodule [3], which retrieves and returns all the policies defined for the specific geographical entity to the user interface [4]. The user then is presented with the policies available for their entity and selects one for which they will record the policy engagement [6]. The policy engagement info is sent with the authentication token to the authorization submodule [7], which, after verifying the user has the privileges to execute this action, sends the policy engagement info to be saved by the policy engagement manager [8].

4.5 Compare policy estimates to actual data

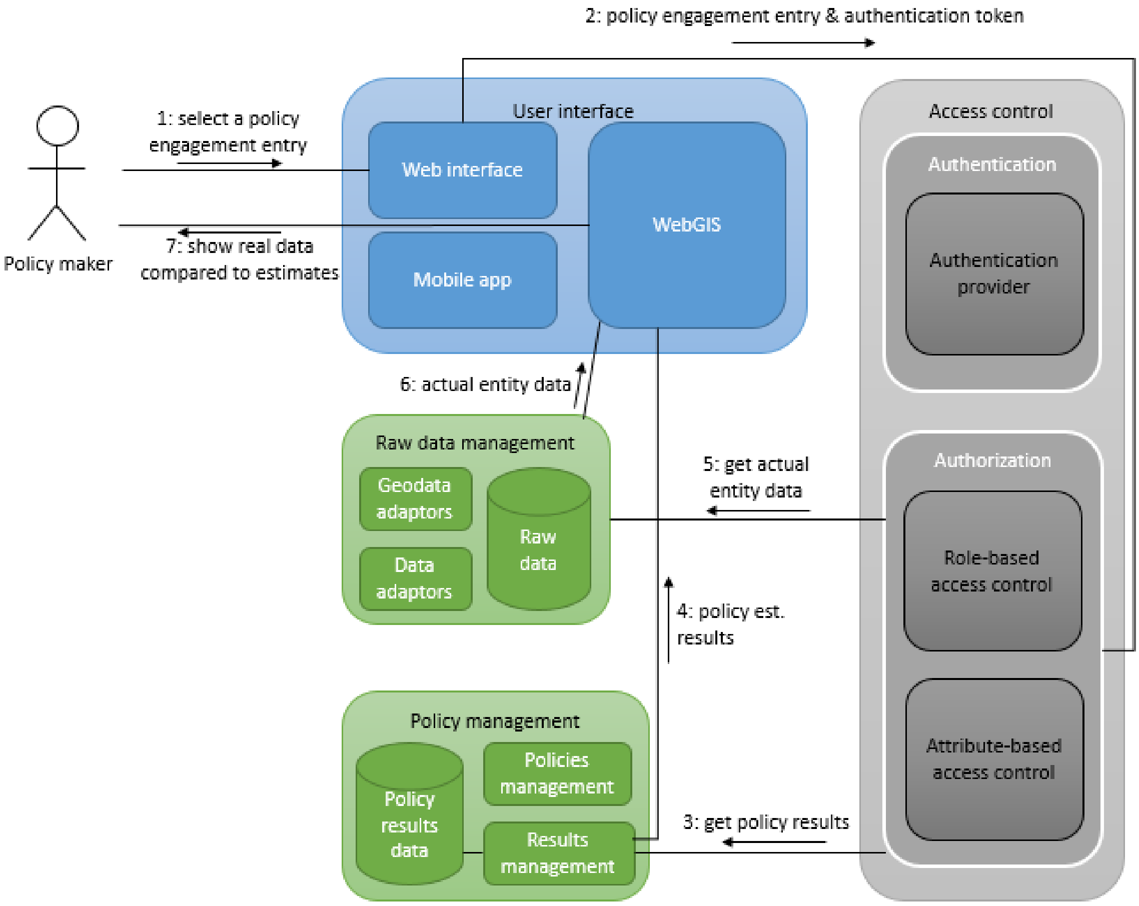

Figure 27 - Compare policy estimates to actual data collaboration diagram

The user must be able to compare the estimated results of the policies they have created to the actual data collected for their geographical entity. This comparison can also take into consideration the policy engagement records. This way, the user can compare the actual data with the estimations that correspond to the already implemented measures only.

Prerequisites/assumptions:

- The user has already logged in to the system
- The user has the role policy maker
- The user has already created policy engagement records – this is not obligatory in order to proceed to the results, it is only an assumption that makes the scenario more inclusive
- The user has already been presented with a list of the policy engagement entries they have created

The user selects the policy engagement entry of which the results they want to compare with the actual data [1]. The web interface sends it to the authorization submodule together with the authentication token [2], where the user's privileges are verified (as described in 4.2). Then, the policy results for the measures that have already been implemented are requested by the results management submodule of the policy management component [3] and once retrieved, they are sent to the WebGIS interface [4]. The real values for the same parameters are also requested for the geographical entity [5] and sent to the WebGIS interface as well [6].

The user can now use the WebGIS interface features, so that they can visualize the differences in actual and estimated values in human-friendly ways, such as by projecting them on maps using layers of several colors.

4.6 Raw data collection and notifications to user-centric tools

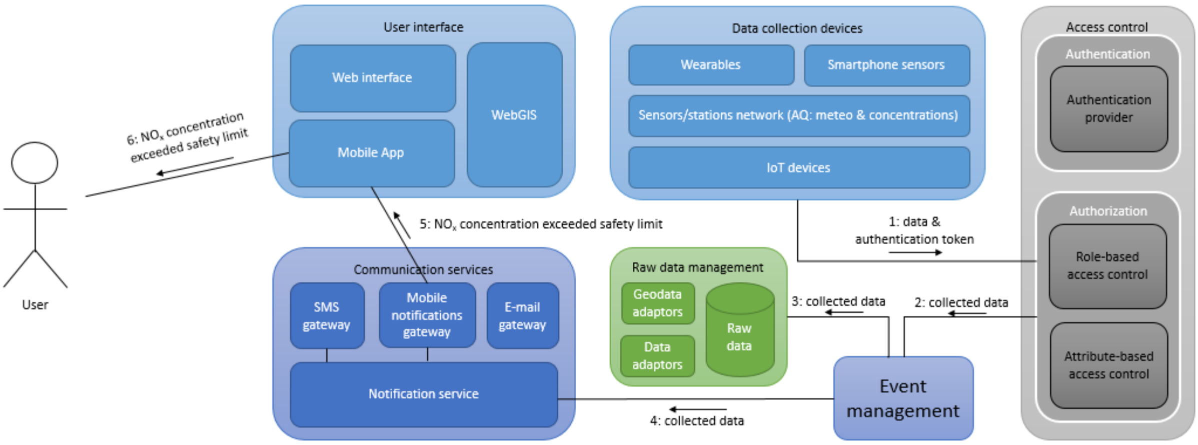

Figure 28 - Raw data collection from user-centric tools collaboration diagram

This collaboration diagram covers several user requirements simultaneously, since it analyses the data collection from wearable devices, sensors networks etc. and combines it with notifications sent to the user-centric tools in case of concentration limits exceedance. We assume that a station sends NO$_x$ measurements to the DSS, which happen to be over the security limit. The data will be stored in the DSS and at the same time users will be notified about the risk.

Prerequisites/assumptions:

- The user belongs to the same geographical entity as the measurement station
- The user has installed the mobile application in their mobile phone
- The notification service has been configured to send notifications to the users when the NO$_x$ concentrations exceed certain limits
- NO$_x$ concentrations exceed those limits

The measurement station sends several data along with its authentication token [1]. The authorization submodule in collaboration with the account management module verifies the station's privileges (as described in 4.2) and creates an event by forwarding the data to the event manager [2]. The event manager will then send the collected data to the raw data management module [3], so that they can be stored in the geodatabase, and also to the notification center [4], so that it can check whether any notification should be sent to the users. The notification service compares the values with the limits and, since they exceed the limit, sends la notification through the mobile notifications gateway to the mobile applications about the risk [5]. Likewise, the user is notified by their mobile application [6].

4.7 Create a geographical entity

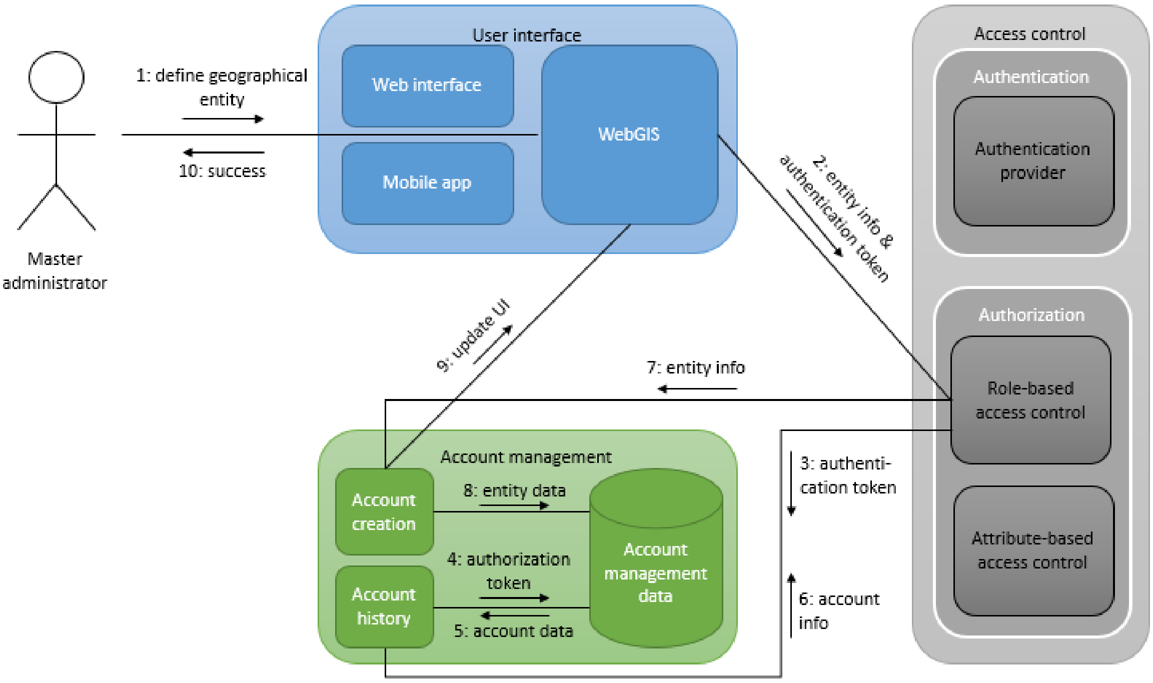

Figure 29 - Create a geographical entity collaboration diagram

The DSS needs to support the definition of geographical entities both for operational and for administrational reasons. The master administrator has the necessary privileges to manage them. In this scenario, we analyze the interactions among the DSS components for the creation of a new geographical entity. To this end, they can input its name, select its type (city, region, country), define the area of its jurisdiction etc.

Prerequisites/assumptions:

- The master administrator is already logged in to the system

- The master administrator has already navigated to the part of the interface that allows them to create a new geographical entity

The master administrator can then use the WebGIS features to select all the parameters regarding the geographical entity under creation [1]. When finished, the WebGIS submodule will send all the required parameters along with the authentication token to the role-based access control of the authorization submodule of the access control component [2]. The role-based access controller will send the authentication token to the account history submodule [3], which will query the database [4] and retrieve the account data to which this authorization token corresponds [5]. Then, the account history submodule forwards the account info back to the role-based access controller [6].

The latter, based on the account type and the account privileges, will assert that the user has the right to perform this action and will forward all the information for the creation of the new entity to the account creation submodule of the account management component [7]. The account management component will check the integrity and the validity of the data to make sure that there are not any name, area or other conflicts and will then store the new entity data to the account management database [8].

Finally, the account management module will update the WebGIS [9] with the new geographical entity that has been created. Likewise, the master administrator is informed about the success of its action and can see the new status quo on the map [10].

4.8 View billing information and send invoice

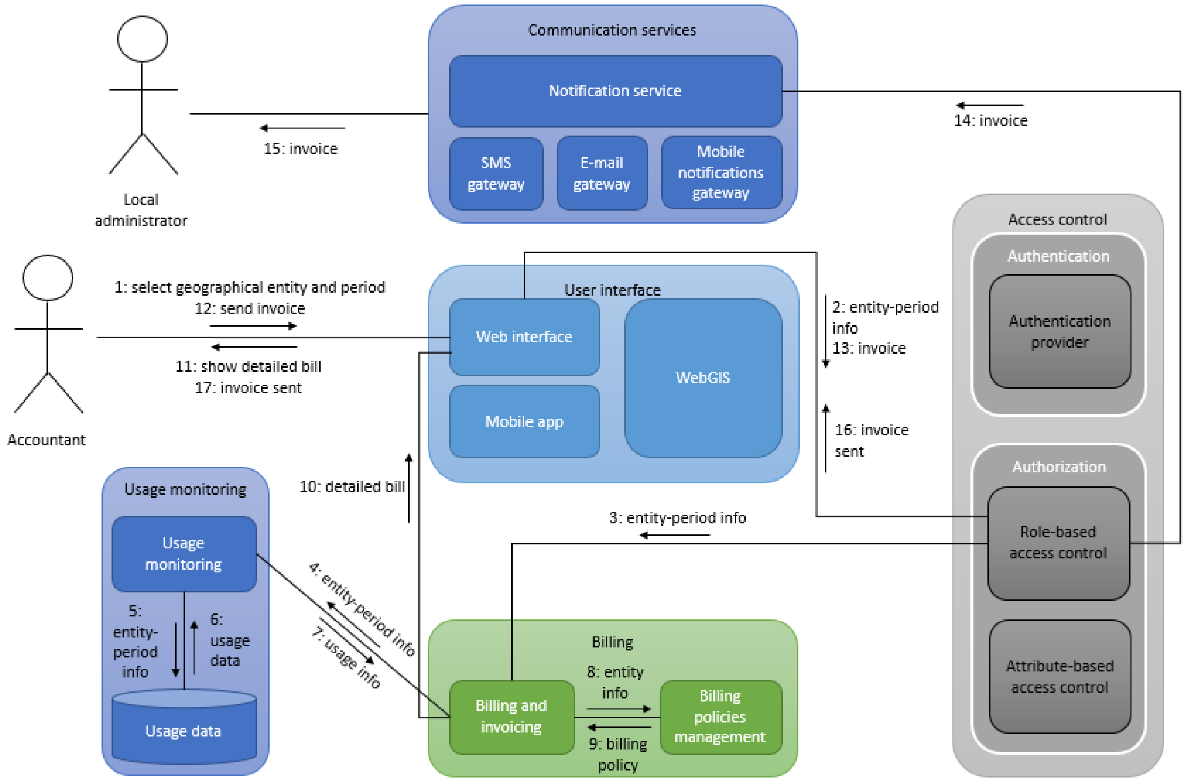

Figure 30 - View billing information and send invoice collaboration diagram

The DSS will be provided as a paid service and as such a billing mechanism must be foreseen. In the current scenario, a user having the accountant role wants to see billing information of a client (corresponding to the administration of a geographical entity). We assume that the client has opted for a pay-as-you-go billing policy.

Prerequisites/assumptions:

- A billing policy has been chosen for the specific geographical entity
- The local administrator of the geographical entity has configured the way they want to receive their invoices
- The accountant user has recently logged in to the system
- The accountant has already navigated to the billing page and is already presented with the list of the clients-geographical entities

The accountant selects the geographical entity for which they wish to see the billing information and the reference period [1]. This information along with the authentication token are transmitted to the authorization submodule of the access controller [2]. The authentication token and the part that the role-based access controller communicates with the account management module in order to request the account information and privileges of the user is omitted for simplicity; it is identical to the authorization control previously described in. Assuming that the user indeed has the permission to see the billing information, the role-based access controller passes the request information to the billing and invoicing submodule [3].

The billing and invoicing submodule needs both the usage data and the billing policy, so that it can calculate the amount due and create the invoice. To do so, it sends the geographical entity and the reference period to the usage monitoring submodule [4], which queries the usage database [5] and gets the usage data [6]. Then, the usage information is sent back to the billing and invoicing submodule [7]. The latter requests [8] and receives [9] the billing policy that applies to the geographical entity during the invoicing period. Having the usage information and the billing policy to be applied, the billing and invoicing submodule can now calculate the amount due and create the invoice, which it sends back to the web interface [10]. The accountant sees the billing information [11] and supposedly decides to send the invoice to the client [12].

The web interface sends the invoice along with the contact communication channel preferences of the geographical entity's local administrator to the access control [13], where the privileges of the user are checked once again, as previously described. Then, this information is passed to the notification service [14], which will sent the invoice to the local administrator by e-mail, through the preferred channel [15] (e-mail or mobile application gateway). The local administrator can also be notified for the invoice by SMS, if they have opted for this choice.

It is worth noting here that the invoicing can be also done in an automatic way, which means that the accountant role need not initiate the process, which will be managed by the event manager component.

4.9 Correspondence to functional requirements

Hereby there is an overview of the functional user requirements that are partially or totally examined in the collaboration diagrams previously presented.

Two-factor authentication

- UR_F17: The DSS must secure access to the system, with various levels of security depending on the user

View predefined policy results

- UR_F3: The DSS must allow viewing the results of simulation for predefined policies and/or technological measures in a region
- UR_F18: The DSS must provide a guidebook on best practices to the public, based on the experience gained through the system

Estimate policy results in emissions

- UR_F1: The DSS must estimate the expected environmental and health impact of a policy option or a technological measure in a region
- UR_F2: The DSS should store the results of the simulations performed for policy options or technological measures

Record policy engagement

- UR_F6: The DSS should provide tools to record policy or measures engagement.

Compare policy estimate to actual data

- UR_F7: The DSS could compare simulation results of the past with actual status, based on realized policies or measures
- UR_F6: The DSS should provide tools to record policy or measures engagement.

Raw data collection and notifications to user-centric tools

- UR_F8: The DSS must allow inputting new emission measurements and integrate it with the existing data available to the DSS
- UR_F9: The DSS must allow inputting new pollutant concentrations and integrate it with the existing data available to the DSS
- UR_F11: The DSS must allow collecting data from the user-centric tools
- UR_F12: The DSS must provide information back to the user-centric tools.

Create a geographical entity

- UR_F13: The DSS must allow the definition of geographical entities and decentralized management

View billing info and send invoice

- UR_F14: The DSS must keep track and/or regulate usage of resources for billing purposes.

5 Physical infrastructure

The infrastructure that will be used to host the developed DSS will be similar to the one presented below:

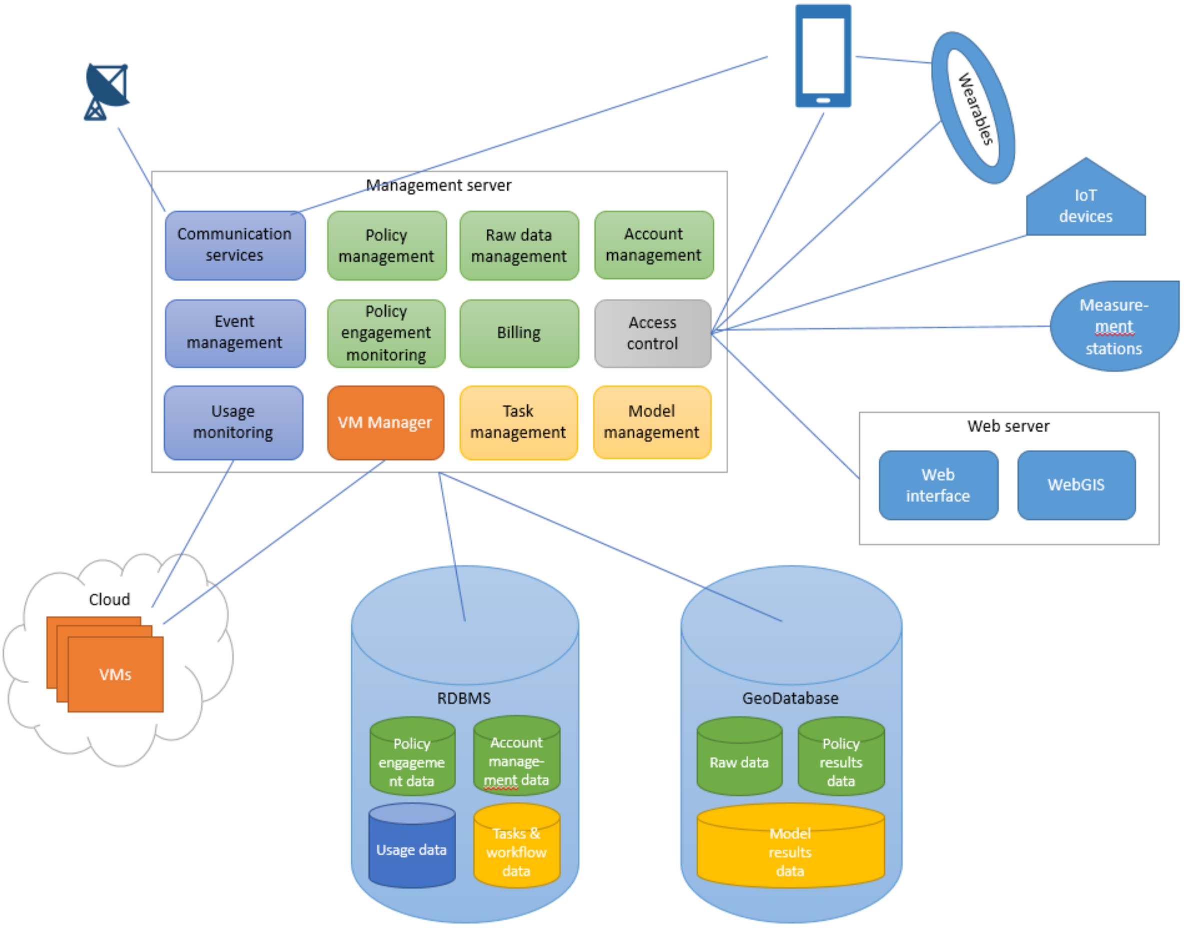

Figure 31 - ICARUS DSS Deployment diagram

Depending on the scalability needs of the solution, more resources will be made available in different areas to respond to increased demand. Most physical nodes may be replicated and load-balanced several times, without impacting the offered functionality.

5.1 Web server

The web server is playing the role of the gateway between the end-user and the applications developed in the context of the DSS.

It will mainly host an apache or ngix server, able to serve thousands of requests per second. It will mainly redirect the request to the management server, but also serve all static content, such as the libraries used for the Javascript based web applications.

If needed, the web server can scale to several physical of virtual hosts without affecting the functionality offered. In that case, a shared file system will be needed to reduce the maintenance time of the static files served by the web servers.

5.2 Management server

The Management Server will initially host all backend services – apart from the ones identified clearly in the previous diagram.

Nevertheless, as the ICARUS DSS will have a clear micro-services architecture, all components will be deployed in separate containers (in the same physical/virtual machine). The advantage of this deployment strategy is that it allows easy and scalable re-deployment of the micro-services to other hosts with simple reconfiguration of the connection points (URIs).

The deployment follows the "Service Instance per Container Pattern" as described in the "Choosing a Microservices Deployment Strategy" article (https://www.nginx.com/blog/deploying-microservices/).

The following diagram indicates how services will be deployed in the same physical host.

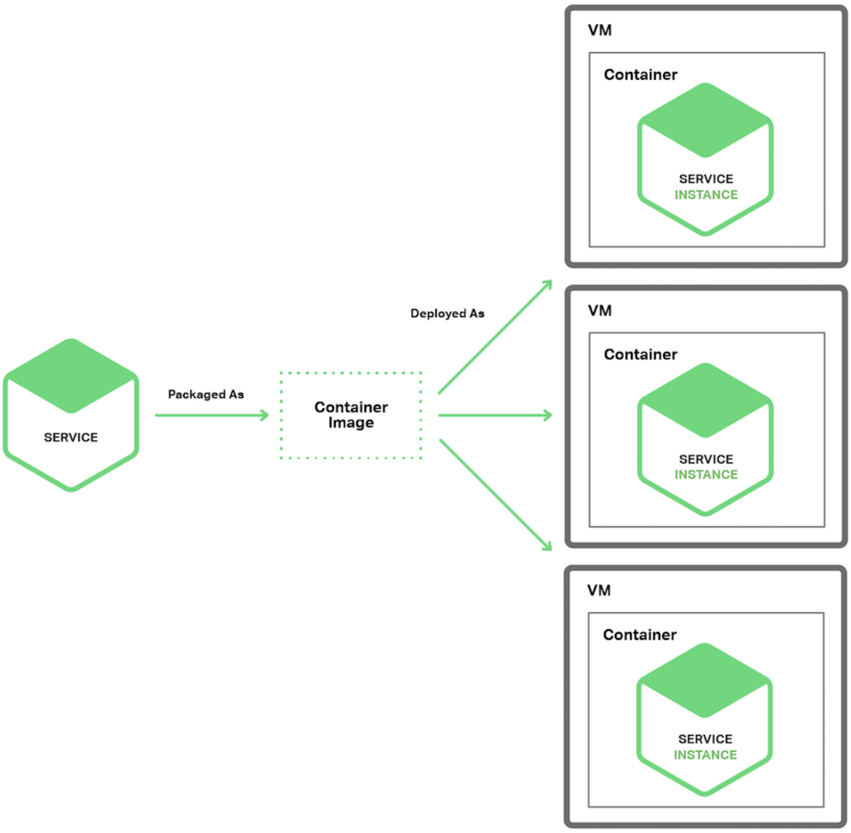

Figure 32 - Deployment diagram of the services

The result of such deployment strategy is that more physical/virtual nodes can be added to the system and containers deployed to them. Requests are directed to the micro-service instances either directly or through a stateless load-balancer that distributed the load between instances of the same micro-service.

We can easily assume then that the entire Management Server is fully scalable, able to respond to thousands of users and/or external applications.

5.3 GeoDatabase

ICARUS DSS will rely on and produce significant amount of geodata. To cover the data persistence and retrieval needs of its several components, a geodatabase will be used. The organization of data in a spatial database provides significant advantages when compared with the utilization of isolated spatial files. Spatial databases are ideal for storing and managing geospatial data. They offer structural, performance, and data management advantages. Data manipulation in databases improves versatility and usability, optimizes performance and simplifies data migration. It also offers tools for data management and maintenance, such as the data integrity, security, spatial indexing, spatial querying etc. There are generally six stages in the database lifecycle (Figure 3).

Analysis: The analysis phase is where the stakeholders are interviewed and any existing system is examined to identify problems, possibilities and constraints.

Design: The design phase is where a conceptual design is created from the previously determined requirements, and a logical and physical design are created that will ready the database for implementation.

Implementation: The implementation phase is where the database management system (DBMS) is installed, the databases are created, and the data are loaded or imported.

Testing: The testing phase is where the database is tested and fine-tuned, usually in conjunction with the associated applications.

Operation: The operation phase is where the database is working normally, producing information for its users.

Maintenance: The maintenance phase is where changes are made to the database in response to new requirements or changed operating conditions, such as heavier load.

5.3.1 GeoDatabase Design

The database design is characterized by the following phases:

i) conceptual design,
ii) logical design and
iii) physical design

The conceptual design stage requires decisions about how real-world dimensionality and relationships will be represented in the database. This phase is independent from software and hardware. The included entities must be defined and how entities are represented in the database must be identified.

Following the identification of the key thematic categories and classes, the database structure will be defined. Subtypes will be identified and relationships will be created with rules for association, as well as classifications for complex code domains. Subtypes and relationships will preserve coarse-grained classes in the data model, and improve display performance. All data fields of all features will be specified. For each data field, the valid values and ranges for all domains will be identified. Primary keys and types of indexes will also be specified.

In contrary with the conceptual and logical design phases which are software independent, the physical design phase depends on the appropriate software that will be utilized for the database development.

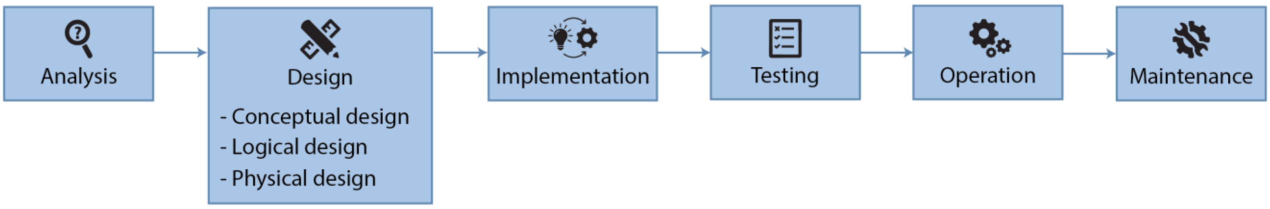

Figure 33 - GeoDatabase lifecycle

A well-designed database is comprised of tables that:

- seek to minimize redundant data
- represent a single subject
- have a primary key
- do not contain multi-part fields
- do not contain multi-valued fields
- do not contain unnecessary duplicate fields
- do not contain fields that rely on other fields

5.3.2 Structure

The structure of a database is the representation of different entities in tables and relationships, which comprise the building blocks of the database. An important part of designing a relational database is dividing the data elements into related tables. The GeoDatabase will contain tables, fields and relationships for at least the following categories:

- Policy results data
- Raw data
- Model results data

An attribute or attributes will serve as the primary key for each table, if any. A primary key (PK) is a unique identifier for a given entity, meaning that a specific record can be selected by only knowing the value of the primary key. Attributes chosen as primary keys will be unique, unchanging, and always present (never NULL or empty).

5.3.3 Relationships

The relationships between tables can pull data together in meaningful ways. They will be established after the fields and types of each table are finalized, when associations between the tables will be more obvious. The database system relies on matching values found in both tables to form relationships. When a match is found, the system pulls the data from both tables to create a virtual record. Cardinality refers to the quantity of elements that interact between two related tables. Identifying the cardinality helps make sure the data are divided into tables most efficiently. Each entity can potentially have a relationship with every other one, but those relationships are typically one of three types:

One-to-one relationships: When there's only one instance of Entity A for every instance of Entity B, they have a one-to-one (1:1) relationship. Unless there is a good reason not to, a one-to-one relationship usually indicates that it is better to combine the two tables' data into a single table.

One-to-many relationships: One-to-many relationships (1:M) occur when a record in one table is associated with multiple entries in another.

Many-to-many relationships: When multiple entities from a table can be associated with multiple entities in another table, they are said to have a many-to-many (M:N) relationship.

5.3.4 Normalization

Normalization is the process of organizing the columns (attributes) and tables (relations) of a relational database to reduce data redundancy and improve data integrity.

- First normal form (1NF) describes a database whose tables represent distinct entities, have no duplicative columns and have a column or columns that uniquely identify each row (i.e., a primary key).
- Second normal form (2NF) describes a database that is in 1NF and also has had duplicative data eliminated.
- Third normal form (3NF) describes a database that is in 2NF and also avoids having columns that derive their values from columns other than the primary key.

The GeoDatabase will be in the third normal form.

5.3.5 Data types

The GeoDatabase will contain tables and fields with spatial and non-spatial information, of different types. Data type is a label and a guideline, in order to identify what type of data is expected inside of each column and also what interactions can be made with the stored data.

The GeoDatabase will support the following main data types:

- Boolean
- Character types such as char, varchar, and text.
- Number such as numeric, integer, etc.
- Temporal i.e., date, time, timestamp, and interval
- Array

Except from the afore-mentioned non-spatial data types, the GeoDatabase will also support geometry and geography data types. The geometry and geography data types include sixteen spatial data objects, or instance types. However, only eleven of these instance types are instantiable - can be created and worked with in a database. These instances derive certain properties from theirparent data types that distinguish them as Points, LineStrings, CircularStrings, CompoundCurves, Polygons, CurvePolygons or as multiple geometry or geography instances in a GeometryCollection.

The figure below depicts the geometry hierarchy upon which the geometry and geography data types are based. The instantiable types of geometry and geography are indicated in blue.

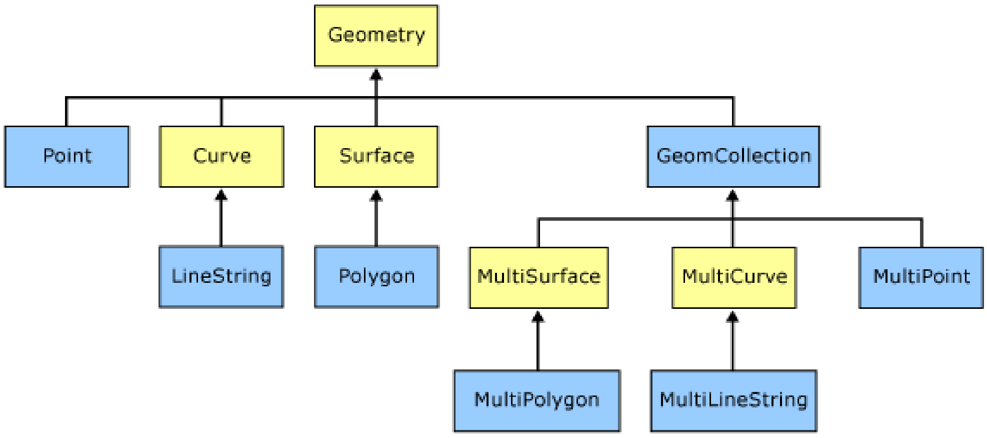

Figure 34 - Geometry hierarchy of data types

5.3.6 Data integrity rules

The GeoDatabase will also be configured to validate the data according to the appropriate rules.

The entity integrity rule says that the primary key can never be NULL. If the key is made up of multiple columns, none of them can be NULL. Otherwise, it could fail to uniquely identify the record.

The referential integrity rule requires each foreign key listed in one table to be matched with one primary key in the table it references. If the primary key changes or is deleted, those changes will need to be implemented wherever that key is referenced throughout the database.

Business logic integrity rules make sure that the data fits within certain logical parameters. For instance, a temperature cannot be lower than -273.15^{o}C.

5.3.7 Queries

In addition to typical SQL queries such as SELECT statements, spatial databases can perform a wide variety of spatial operations. The following operations and many more are specified by the Open Geospatial Consortium standard:

- Spatial Measurements: Computes line length, polygon area, the distance between geometries, etc.

- Spatial Functions: Modify existing features to create new ones, for example by providing a buffer around them, intersecting features, etc.

- Spatial Predicates: Allows true/false queries about spatial relationships between geometries. Examples include "do two polygons overlap" or 'is there a polygon located within 100 meters of another polygon

- Geometry Constructors: Creates new geometries, usually by specifying the vertices (points or nodes) which define the shape.

- Observer Functions: Queries which return specific information about a feature such as the location of the center of a circle

5.3.8 Security

Database security refers to the collective measures used to protect the database, including the data, the database applications or stored functions, the database systems, servers and associated network links, from intrusions, improper modifications, theft, and unauthorized disclosure of private information.

Some of the security measures, which will be enforced to the database are the following:

1. Password Protection: All accounts and resources must be protected by passwords which meet the following requirements, which must be automatically enforced by the system:

2. Must be at least eight characters long;

3. Must include at least three of the following four characteristics in any order: upper case letters, lower case letters, numbers, and special characters, such as !@#$%^&*; and

4. Must be changed at least once per year.

5. Software Updates: Systems must be configured to automatically update operating system software, server applications (webserver, mailserver, database server, etc), client software (web-browsers, mail-clients, office suites, etc), and malware protection software (anti-virus, anti-spyware, etc);

6. Firewall: Systems must be protected by a firewall which allows only those incoming connections necessary to fulfill the business need of that system;

7. Throttle repeated unsuccessful login-attempts: A maximum rate for unsuccessful login attempts must be enforced. Account lockout is not required, but the rate of unsuccessful logins must be limited; and

8. Enable session timeout: Sessions must be locked or closed after some reasonable period.

5.3.9 Backup and recovery

The purpose of the backup and recovery strategy is to protect the database against data loss and reconstruct the database after data loss. Backup administration tasks will include the following:

- Planning and testing responses to different kinds of failures;

- Configuring the database environment for backup and recovery;
- Setting up a backup schedule;
- Monitoring the backup and recovery environment;
- Troubleshooting backup problems; and
- Recovering from data loss if the need arises.

5.3.10 Software

Taking the user requirements under consideration, the best choice for the Database Management System (DBMS) to be used in the development of the DSS is the free and open source PostgreSQL object-relational system, as well as the PostGIS spatial database extender for PostgreSQL.

PostgreSQL is an open source, powerful, mature, reliable and platform independent object-relational database system. It has more than 15 years of active development and an architecture that has earned a strong reputation for reliability, data integrity, and correctness. PostgreSQL provides transaction management, disk storage routines, SQL processing and planning and other tasks of standard database. It runs on all major operating systems, such as Linux, UNIX and Windows.

PostGIS is a spatial database which is built on top of a standard PostgreSQL database. PostGIS provides corresponding spatial types, functions and indexes. It adds spatial data types like points, line strings, polygons, multipoint, multiline strings, multipolygons and geometry collections. It supports both geometry and geography types. It also adds spatial functions like area, distance, union, difference, buffer, touches, intersects, within, contains, overlaps etc.

5.4 RDBMS

The need of a database to store all other information, different from the geospatial, made the team select a relational database. To be in line with the geospatial data and ease future management of the two databases, Postgresql has also been selected as the core RDMBS.

Discussions are on-going related to the use of NoSQL databases for the big amount of raw data collected and analysed by the system. There are advantages and disadvantages in this solution that prohibits the team from making a clear choice at this stage of the project.

Most probably, the final solution will be a mixture of the two technologies, one RDBMS and a NoSQL database. The RDBMS will continue being used for what we can refer as "structured data", while the NoSQL for all data introduced in the system as raw data that needs further processing.

5.5 Cloud platform

The processing of the various models needed to convert raw data to meaningful information requires heavy processing power and – occasionally – clusters of powerful computers.

Originally, partners thought of using a privately owned dedicated infrastructure to perform all those calculations. This solution had though the following problems:

- The dedicated infrastructure has a very high initial cost, which may not be justified by the real usage of the system
- The dedicated infrastructure, even very powerful, would have limitations on the number of models it can serve and the time it takes to complete the execution of one

Upcom has good experience in cloud computing and the use of cloud resources in the execution of High Performance Computing algorithms. They proposed the use of cloud resources for the execution of the model. The advantages were clear:

- The scalability of the system is only limited by the actual models and the way they are developed; it is no longer a problem of available resources, but rather a problem of how to best organize those resources
- The initial cost may be very low, as even demanding models may be executed with a small number of nodes in a few hours time; renting those resources from public data centers would cost a fraction of the cost of a respective proprietary system.

All partners agreed that it was a much better solution than the currently available. The use of cloud resources would solve the problem of processing power needed to execute difficult models.

Initially, the effort will be devoted to migrating all model code to run on a cloud resource. When successful and if needed (depending on the actual execution time of a model), model code will have to be modified to support parallel execution in multiple nodes.

More difficult models may start using instead of normal Central Processing Units (CPUs), Graphical Processing Units (GPUs) offering at least 100 times more processors than a system with parallel CPUs. The end results could be exceptional, especially if communication between nodes is reduced to the absolute minimum.

To improve the management of those cloud resources, the team made a choice of using virtualized resources running on physical nodes with minimal overhead. That implies deploying a single virtual machine or container on a physical node, but exploiting all the advantages of managing a virtual node instead of a physical node. Most big data centers offer that kind of virtual machines/containers nowadays; some of them even offer virtual machines using the GPU(s) of the physical host, allowing the execution of highly parallel algorithms on those systems with excellent results.

5.6 Data collection devices

The data collection devices are external to the system, as they are not managed and supported by the team. They are represented in the diagram to indicate the complexity of this interaction.

The data collection devices we foresee are:

- the local station used by authorities to measure concentrations in the city
- devices used ad-hoc by the research partners to measure emissions and/or concentrations in the field
- wearables that are presented to the users by the ICARUS team, but they interact with the DSS via the ICARUS mobile applications

In all cases, the DSS should store the data collected by those devices and make it available to the system components that need them – such as the modelling engine.

6 Development methodology

The ICARUS DSS, as most software development efforts in this project, will be implemented using the Agile software development methodology SCRUM. The core components of the methodology are described below.

Scrum focuses on the business-critical processes and disciplines needed to deliver services around development. These procedures provide the structure behind our delivery.

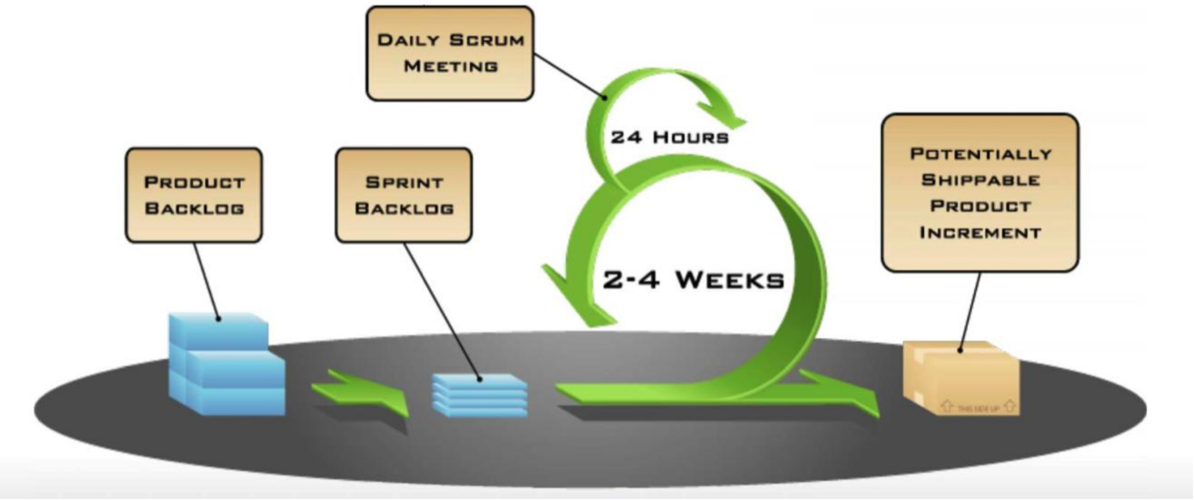

Figure 35 - Scrum development cycle

6.1 Characteristics

Scrum is a "process skeleton," which contains sets of practices and predefined roles. The main roles in Scrum are:

- the "Scrum Master," who maintains the processes
- the "Product Owner," who represents the stakeholders
- the "Development Team", a cross-functional group of about 7 people who do the actual analysis, design, implementation, testing, etc.

During each "sprint", typically a two to four-week period (with the length being decided by the team), the team creates a potentially shippable product increment (for example, working and tested software).

The set of features that go into a sprint come from the product "backlog", which is a prioritized set of high level requirements of work to be done. Which backlog items go into the sprint is determined during the sprint planning meeting. During this meeting, the Product Owner informs the team of the items in the product backlog that he or she wants completed. The team then determines how much of this they can commit to complete during the next sprint. During a sprint, no one is allowed to change the sprint backlog, which means that the requirements are frozen for that sprint. After a sprint is completed, the team demonstrates the use of the software, during the sprint review meeting.

A key principle of Scrum is its recognition that during a project the stakeholders can change their minds about what they want and need (often called requirements churn), and that unpredicted challenges couldn’t be easily addressed in a traditional predictive or planned manner. As such, Scrum adopts an empirical approach—accepting that the problem cannot be fully understood or defined, focusing instead on maximizing the team's ability to deliver quickly and respond to emerging requirements.

6.2 Documents

The product backlog is a high-level document for the entire project. It contains backlog items: broad descriptions of all required features, wish-list items, etc. prioritized by business value. It is the "What" that will be built. It is open and editable by anyone and contains rough estimates of both business value and development effort. Those estimates help the Product Owner to gauge the timeline and, to a limited extent, priority. For example, if the "add spellcheck" and "add table support" features have the same business value, the one with the smallest development effort will probably have higher priority, because the ROI is higher.

The product backlog is property of the Product Owner. Business value is set by the Product Owner. Development effort is set by the Development Team.

6.3 Implementation for ICARUS

- The traditional product owner role will be replaced by a “**Proxy Product Owner** (PPO)”, member of the consortium team, who will be responsible to interact with the stakeholders whenever needed. The “Proxy Product Owner” will have the empowerment (inside the team) to refine specifications (User Stories), define priorities, and clarify business issues.

- During the Sprint Planning meeting, an evaluation of the weight of each User Story and the capacity of the team to complete them in the Sprint is done with Story Points, using a modified version of the Fibonacci sequence, with numbers such as 1,2,3,5,13,40,100.

- Sprints will usually take 2 weeks, although the Sprints duration may change, if needed. Nevertheless, to avoid repetitive acceptance tests from the stakeholders, the maximum number of intermediate deliveries of a project will not exceed 3; the first after the first Sprint to make sure initial specifications are well understood, a second half way and the last upon project completion. The stakeholders will be able to participate in all Sprint Review meetings to closely monitor the evolution of the project.

- After the identification of potential issues by the stakeholders, those will be introduced in the Jira tool and re-introduced in the backlog of the project. The team, in agreement with the PPO, will re-plan them in the following Sprints based on the usual planning methods. If blocking issues are discovered during the acceptance tests, the team will devote time to resolve them immediately, so that tests can proceed.

- Each User Story in the Sprint will be split in 4-6 separate tasks, namely:

  - Analysis and graphical layout (if needed and agreed during the Sprint Planning), realized in collaboration of the PPO and the graphic designer;
  - Software design and development, realized by the developer responsible for the User Story; unit testing will be integral part of this task and it will only be marked as DONE, if unit tests run successfully;
  - Code and Design Review, realized by a second developer of the team;
  - Testing, realized by the Tester of the team; the Tester might update the acceptance criteria of the User Story to include more tests, as needed;
  - Documentation, realized by all members participating in the User Story implementation; it includes technical documentation, both in code and in the Software Architecture Document, as well as user-related (updates to the User Manual);
  - Security Assessment (if needed and agreed during the Sprint Planning), performed jointly by all team members involved in the User Story; the Assessment will be based on a well-defined questionnaire that requires all potential threats are properly handled

- A User Story will be marked as DONE, only if all sub-tasks are marked as DONE in the system.
- For every Sprint, a standardized User Story will be added for the deployment of the application; the Story will be realized by a member of the Maintenance Team and will contain the following tasks:
  - Automated deployment script implementation, following a DevOps methodology;
  - Review Script, performed by another Maintenance Team member;
  - Testing, performed by the team tester;
  - Documentation, performed by a member of the Maintenance Team;
  - Security Assessment of the script, performed by all team members involved in this task.

6.4 Testing Approach

The approach to testing is composed of two parts: methodology and culture! The methodology specifies the steps each member of the team should follow to guarantee quality results of his work. Culture is inherent and assumes responsibility at all levels.

6.4.1 Testing Methodology

All artefacts of the Sprint are gone through a rigorous testing (quality assurance) methodology, with clearly identified steps for each phase of the development:

- **Requirements specifications, analysis, wireframes, User eXperience (UX) recommendations, graphical design, etc.**: Irrespective if the document is drafted by the stakeholders or the consortium, the final document is reviewed by a second team member; all versions are stored in the document management system, clearly identified by their version.

- **Test scenarios**: They are normally embedded in the acceptance criteria of every User Story and reviewed by the developer implementing the Story.

- **Source code**: The majority of the backend source code developed will be unit-tested by each developer and tests run in the Continuous Integration environment (based on Hudson); junit will be the core library used to automate the unit tests, without excluding the use of other, most specific tools – if needed. To check good unit test coverage, cobertura will be used at the CI environment and results will be send back to the team – at least – on a daily basis. SonarCube will be also be part of the CI environment and the set of rules to check code quality will be proposed by the consortium and agreed with the client. Finally, a code review step has been added to all User Stories, to achieve maximum quality of the developed code. This step MUST be executed by a second team member and this will be monitored and controlled continuously.

- **Executable**: Based on the delivered source code and the acceptance criteria of each User Story, automated functional tests will be created using celenium and executed during the Continuous Integration daily step. All load and performance tests required to guarantee the expected response time under heavy load will be developed using jMeter. An effort will be done to automate this process in the long run and integrate it in the CI environment, but initial Sprints may be completed with a manual execution of those tests. Manual tests are also foreseen for most of the applications, to capture nuances that are difficult to test and validate automatically, such as correct layout on screen, colors, etc.

- **Deployment scripts**: The deployment of the applications will be automated using DevOps tools (such as Ansible). Deployments will be tested in various environments. Again, deployment will be triggered by an administrator, but will run automated until the successful deployment of the application in the respective environment. The testing will be based on its successful execution in different environments.