PESCaDO

**FP7-248594**

# Personalized Environmental Service Configuration and Delivery Orchestration

# D8.6 Set-up of an operational system platform

**Due date of deliverable: 31.12.2010**
**Actual submission date: 23.12.2010**

Start date of project: 1st January, 2010 Duration: 36 months

Lead contractor for this deliverable: Fraunhofer IOSB Revision V1.0

| D8.6 | Set-up of an operational system platform |
|---|---|
| **Project Acronym :** | PESCaDO |
| **Contract No :** | FP7 - 248594 |
| **Due Date :** | 31.12.2010 |
| **Reply To:** | Jürgen Moßgraber juergen.mossgraber@iosb.fraunhofer.de |
| **Actual date of delivery:** | 23.12.2010 |

**Deliverable Identification Sheet**

| | |
|---|---|
| **Project ref. no.** | FP7-248594 |
| **Project acronym** | PESCaDO |
| **Project full title** | Personalized Environmental Service Configuration and Delivery Orchestration |
| **Security (distribution level)** | PU |
| **Contractual date of delivery** | Month 12, 31.12.2010 |
| **Actual date of delivery** | Month 12, 23.12.2010 |
| **Deliverable number** | D8.6 |
| **Deliverable name** | Set-up of an operational system platform |
| **Type** | Report |
| **Status & version** | Submitted, V1.0 |
| **Number of pages** | 55 |
| **WP / Task responsible** | Fraunhofer IOSB |
| **Other contributors** | All |
| **Author(s)** | All Partners of the Consortium |
| **Internal Reviewer** | Leo Wanner |
| **EC Project Officer** | Manuel Monteiro |
| **Abstract** | The set-up of an operational (i.e., implemented) system platform is a software deliverable which is described in this document. Additionally it contains an updated specification of the architecture for the first SW development cycle which incorporates the new insights gained in the course of the work on both the individual technologies and the integrated system. |

# Table of Contents

# Executive Summary

This document specifies the PESCaDO Architecture (PARCH) and Software Demonstrators of the European STREP FP7-248594 Personalized Environmental Service Configuration and Delivery Orchestration (PESCaDO). The specification of PARCH follows an iterative design approach. Each iteration builds on the results of the former iterations which are described in the deliverables D8.1, D8.3, D8.6, D8.7, D8.8 and D8.9.

The specification follows the Reference Model for Open Distributed Processing (RM-ODP), which defines 5 viewpoints: Enterprise Viewpoint, Computational Viewpoint, Information Viewpoint, Technology Viewpoint and the Engineering Viewpoint. The document is structured in the same order.

In the *Enterprise Viewpoint*, the user requirements are analysed and documented in terms of their functional, informational and qualitative aspects. In PESCaDO, this information is derived from the given PESCaDO Use Cases.

The *Computational Viewpoint* is referred to as *Service Viewpoint* in PESCaDO in order to stress that the focus is not on providing a distributed computing support with tightly-coupled components, but on inter-connecting functionalities and information in terms of services (Service Oriented Architecture - SOA). Thus, the Service Viewpoint classifies and specifies the functional requirements in terms of services.

The *Information Viewpoint* classifies and specifies the informational requirements in terms of an information model. The information model covers, among others, ontologies, user problem descriptions, and information on data nodes with environmental data. The ontologies will be in PARCH one of the main data structures. They will be used by various services of the system in order to guarantee the semantic orchestration of heterogeneous environmental service nodes, user-oriented decision support, and environmental information production and delivery. The description of a problem in terms of user queries will be expressed in formalized statements in terms of PESCaDO's *Problem Description Language*. The information about data nodes which contain required environmental data will be represented in relational structures, which are also discussed under the Information Viewpoint.

The *Technology Viewpoint* specifies the characteristics of the service platform upon which the services and information models are to be mapped for a specific geospatial service network. Under this viewpoint, the focus is thus on the technology used to implement the Knowledge Base.

The *Engineering Viewpoint* presents the realization of the service and information model specifications in the chosen service platform(s). In this deliverable, the possibilities offered by a SOA are outlined by showing how the PESCaDO services can be implemented using a specific computer hardware.

# 1 Introduction

This document specifies the PESCaDO Architecture (PARCH) of the European STREP FP7-248594 Personalized Environmental Service Configuration and Delivery Orchestration (PESCaDO). Additionally it describes the functionality of the Demonstrators D8.6, D8.7, D8.8 and D8.9 when implemented.

The specification of PARCH follows an iterative design approach. Each iteration builds upon the results of the former iterations in that topics described in former iterations are refined and/or new topics are taken up and specified. Furthermore, each iteration of the architecture specification is consistent with a corresponding version of the PESCaDO workbench implementation (D8.6, D8.7, D8.8 and D8.9). Currently, five versions of the PARCH specification are foreseen. The roadmap for the individual versions is continuously adapted according to the analysis of the user requirements and their prioritisation as well as the technological challenges that PESCaDO aims to solve. The focus of the present version 2 is the provision of basic functional and informational building blocks derived from User Case scenario 1 outlined in Section 3 and its first basic implementation.

The objective of this deliverable is to motivate and specify the basic design decisions derived from user requirements and generic architectural principles. It is structured according to the viewpoints of the Reference Model for Open Distributed Processing (RM-ODP) as defined in ISO/IEC 10746-1:1998 (E). The RM-ODP viewpoints are interpreted in the context of a SOA in analogy to the interpretation of the RM-ODP in the Reference Model for the ORCHESTRA Architecture (RM-OA) [Usländer, 2007]. PARCH provides the basic concepts, their interrelationships (conceptual models) and abstract specifications of implementation models, services and interfaces. “Abstract” means that the specification is independent of the specifics of a particular service platform. This methodology has already been used successfully within the European Integrated Projects FP6-511678 ORCHESTRA (Open Architecture and Spatial Data Infrastructure for Risk Management) and FP6-033564 Sensors Anywhere (SANY).

## 1.1 Changes compared to D8.3

| Chapter | Change |
|---|---|
| 3 | Adjusted the Sequence Diagram and its’ description to suit the modified services. |
| 5 | Removed DataNodeRepositoryService (now integrated inDRS) |
| | Updated Answer Service specification |
| | Updated Problem Description Generation Service specification |
| | Updated Data Retrieval Service specification |
| | Updated Decision Service specification |
| | Updated Related Aspects Computation Service specification |
| 8 | Added description of D8.6 demonstrator |

# 2 Architectural Design Process

## 2.1 Architectural Design Principles

As already mentioned above, PESCaDO uses the same design methodology as used in the European Integrated Project FP6-511678 ORCHESTRA [ORCHESTRA, 2008]. The following architectural design principles apply as well (with some slight adaptations):

- Technology Independence: The PESCADO Architecture shall be independent of specific implementation technologies (e.g. middleware, programming language and operating system), their development cycles and their modifications and shall not be influenced by or deal with technical limitations of specific implementation technologies. It must be possible to accommodate modifications of a technology (e.g. the lifecycle of middleware technology) without changing the PESCADO Architecture itself.
- Evolutionary Development – Design for Change: The PESCADO Architecture shall be designed to evolve, i.e. it shall be possible to develop and deploy the system in an evolutionary way. The PESCADO Architecture shall be able to cope with changes of user requirements, system requirements, organisational structures, information flows and information types in the source systems.
- Component Architecture Independence: The PESCADO Architecture shall be designed such that PESCADO Services and the access to environmental data nodes are architecturally decoupled.
- Generic Infrastructure: The PESCADO Architecture Services shall be independent of the application domain. This means that the PESCADO Architecture Services should be designed in such a flexible and adaptable way that they can be used across different thematic domains and in different organisational contexts, and that the update of integrated components (e.g. applications, systems and ontologies) will cause little or ideally no changes with respect to the use of the PESCADO Architecture Service.
- Self-describing Components: Components of a PESCADO Service Network, such as data elements or services, shall include descriptions of their critical characteristics, including sources, assumptions and the like. The usage of self-describing components that provide context-sensitive formal and semantic descriptions of their interfaces can help to realise semantic interoperability.

## 2.2 Evolution of the PESCaDO Architecture

PESCaDO follows an iterative design approach with the following characteristics (see Figure 1):

The iteration steps are determined by two dimensions: 1) the content dimension that defines the set of topics that are handled in the specification, and 2) the refinement dimension that relates to the level of specification details given in the document.

Each iteration builds on the results of the former iterations, which means that specifications of former iterations are refined and/or new topics are taken up and specified.

Each version of the architecture specification is consistent with the corresponding version of the PESCaDO workbench implementations.

In Figure 1, V1 represents the initial version of the document. The versions V2, V3, V4 and V5 correspond to the implementation phases in the PESCaDO project to follow (D8.6, D8.7, D8.8

and D8.9). The content evolves as described in D8.2 Roadmap for the development of the PESCaDO workbench.

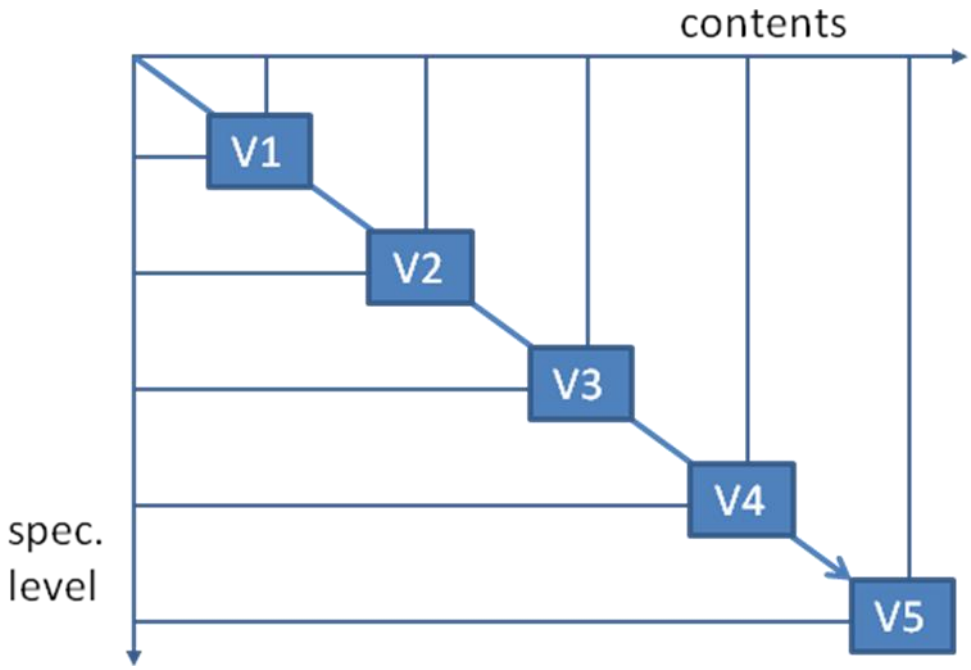


Figure 1: Evolution of the Specification of PARCH

## 2.3 Application of the RM-ODP and the RM-OA

Following the model established by the ORCHESTRA project [Usländer, 2007], an approach based on the ISO Reference Model for Open Distributed Processing (ISO/IEC 10746-1:1998) has been selected for the design of PARCH. The usage of RM-ODP for the PESCaDO Architectural design process is applied on a big scale to the structuring of ideas and the documentation of PARCH itself.

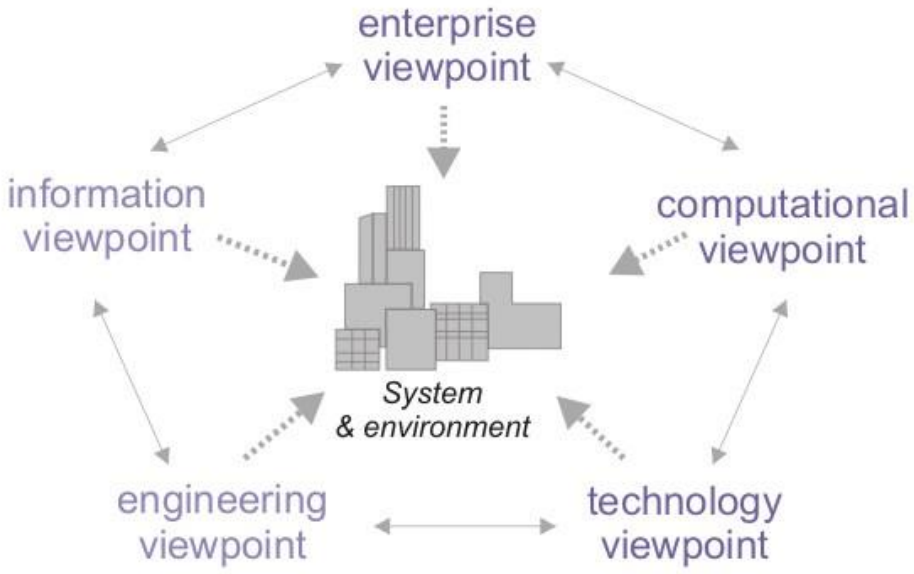


Figure 2: Viewpoints in RM-ODP [Dekker, 2008]

RM-ODP subdivides the specification of a complete system into the so-called viewpoints (see Figure 2). However, as the RM-ODP has originally been conceived in the spirit of distributed object-oriented middleware, the ORCHESTRA process model has adapted the RM-ODP viewpoints to the design of geospatial service oriented architectures and service networks:

In the Enterprise Viewpoint perspective, the user requirements are analysed and documented in terms of their functional, informational and qualitative aspects.

The Computational Viewpoint is referred to as the Service Viewpoint in PESCaDO in order to make explicit that the focus is not on providing a distributed computing support with tightly-coupled components, but, rather, on inter-connecting functionalities and information in terms of services. Thus, the Service Viewpoint classifies and specifies the functional requirements in terms of services. Specific to PESCaDO is the aim of specifying the services first in a platform-neutral manner (e.g. in UML) in order to be able to map them to different service platforms as required.

The Information Viewpoint classifies and specifies the informational requirements in terms of an information model. As for the services, the aim is to do this first in UML to be platform-independent.

The Technology Viewpoint specifies the characteristics of the service platform upon which the services and information models are to be mapped for a specific geospatial service network.

The Engineering Viewpoint specifies the mapping of the service and information model specifications to the chosen service platform(s). Furthermore, the operational policies of the service networks are derived from the qualitative requirements.

# 3 Enterprise Viewpoint

The use cases for PESCaDO are described in detail in the deliverable "D8.5 Specification of the Pilot Use Cases in PESCaDO". To derive requirements for the architecture, we present a first scenario which is part of Use Case 1.

The range of users addressed in the first scenario covers citizens with no professional background on environmental services or clients of the services offered by HSY for the Helsinki Metropolitan Area (HMA). The principal duties of HSY comprise waterworks (production of drinking water and treating the wastewater), waste management and air quality management (AQ) for its four member municipalities (Helsinki, Espoo, Kauniainen and Vantaa). The population of the HMA-area is about 1 million inhabitants. The current services are contracted by a number of different institutions and regularly consulted by citizens. The citizens in Use Case 1 are also health-conscious and outdoor activity engaged general public as well as traffic participants.

## 3.1 Scenario 1

In Use Case 1, the PESCaDO workbench is used to help a citizen with planning a journey (be it on a regular basis such as commuting between home and place of work or a vacation trip) or an excursion. The Use Case comprises issues which may affect the decision concerning a journey – as e.g. weather, air quality, and pollen. The consideration of these issues, combined with traffic situation, is supposed to make the journey fluent and minimize the exposure to pollutants, sun, etc. These issues are useful in routine journey planning and, for example, in air quality episodes when traffic is restricted in a city centre.

Figure 3 illustrates a sample question a user might ask and the necessary steps the system needs to execute to answer this question.

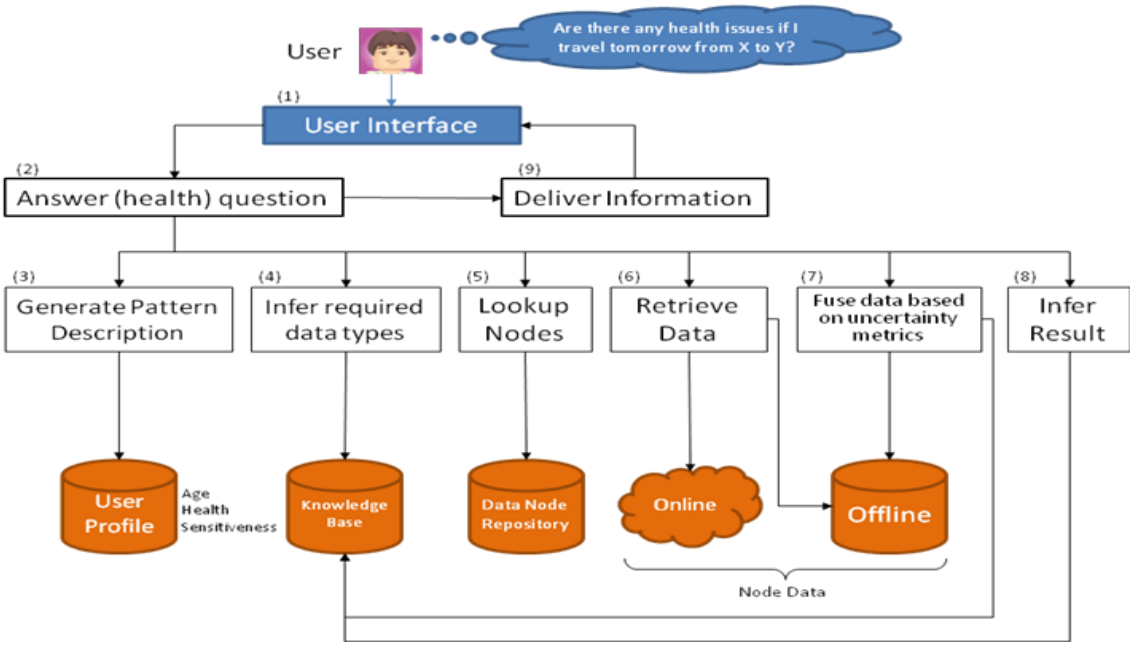


Figure 3: A possible question and the system's reaction in Use Case 1

The user wants to travel from a location X to a location Y in the near future and is interested to know whether any health issues might be provoked by environmental conditions, i.e., weather, air quality, pollen, dense traffic or dangerous roads. They might provide further information such as the transportation means of their preference, diagnosed allergies, etc. Another option is that this information is already stored in a user profile. The steps the system needs to execute are:

1. User Interface: Allows the user to formulate their question.
2. Answer question: Controls the workflow which is necessary to compute an answer.
3. Generate Pattern Description: Generates a formalized description of the user's input.
4. Infer required data types: Infers the aspects which need to be checked to answer the question. For instance, in the context of a question concerning health issues, pollen concentrations, air quality, weather, and road conditions need to be taken into account.
5. Lookup nodes: Retrieve the environmental nodes which contain data for the identified aspects.
6. Retrieve data: Retrieve the required data from the environmental nodes. The data might be already downloaded, retrieved online from a node or calculated by other services (as, e.g., an air quality forecast maybe calculated and provided by a separate service).
7. Fuse data: If there are several data sources for a specific aspect, fuse the values in accordance with the result of the application of uncertainty metrics that assess the trustworthiness and precision of the values from different sources.
8. Infer results: Compute the answer based on the fused data, i.e., select all content that is to be judged relevant to the answer of the user.
9. Deliver information: Cast the selected content into the appropriate mode (text, graphic or table) and deliver the answer to the user.

## 3.2 Requirements of the User Interface

The complexity and expressiveness of the user interface depends on the requirements that arise from the different user scenarios and from the different components of the workbench. For the first sample scenario in Use Case 1 sketched above, user interaction comprises mainly query generation and answer presentation. The user interface must be flexible enough to support these tasks by providing (e.g., wizard-based) rich interaction with free querying, reformulation and extendable system answer representation. Furthermore, backend task-like administration and quality assurance are important aspects.

Figure 4 shows a mock-up example of how the user can state a request of the type of the first use case scenario. A summarizing form (top left) provides access to the wizards for specialized input of different aspects. Among these wizards, there is a map component for geo references, a component to state the time frame of the request and a suggestion-based selection of information aspects that should be covered by the response of the system. The navigation in the supported aspects and tasks is supposed to be easily accessible and without a steep learning curve.

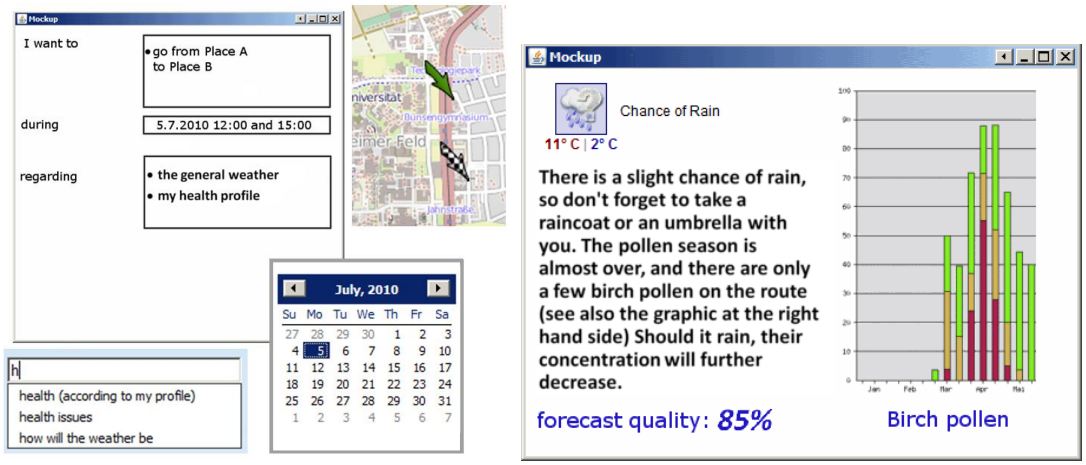


Figure 4: Wizard based request formulation and basic answer of the system

The representation of the answer of the system should be as concise as possible and as extensive as needed in order to be comprehensive. The selection of the content to be communicated in the answer is based on the user profile and system-internal content relevance criteria. To ensure that the user trusts the answer and to support a learning effect, the user must be able to navigate to additional information which provides a more detailed justification for the answer.

For administrators, advanced end users and potential service providers, a more flexible and sophisticated decision support interaction is required. Figure 5 depicts a mock-up of a possible user interface implementation for this interaction. The visual query editor allows free use of available parameters for the problem description. Visual aids like the structured query layout or colouring according to areas of interest (environmental information parameters, geographical parameters, user profile parameters, etc.) can help to keep the increased complexity manageable.

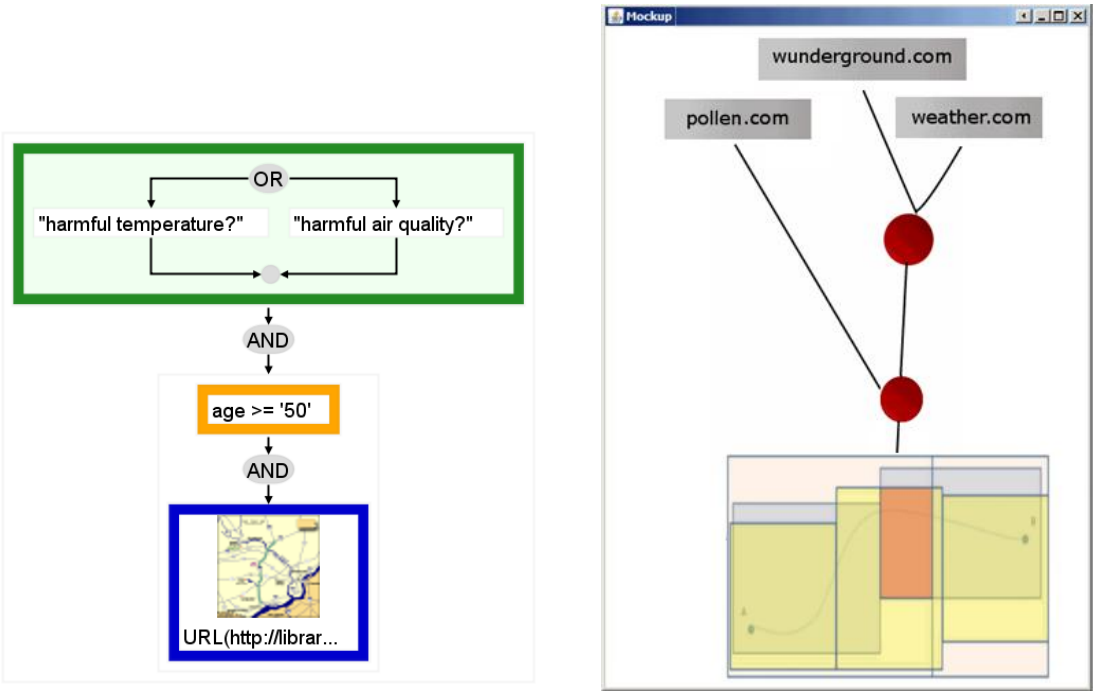


Figure 5: More complex query generation and data provenance

On the answer representation side, additional information about data provenance, orchestration and result explanation is available for exploration and in-depth analysis. This information helps administrators to assess quality aspects, track how uncertainty has been introduced by orchestration, and optimise the reasoning for an improved end user experience of future requests.

## 3.3 Sequence Diagram for Scenario 1

The following two sequence diagrams display an example workflow of the services which are required to answer the user's question in the context described in Section 3.1. The services are described in detail in Section 5. The Knowledge Base Access Service (KBAS) and the User Profile Management Service (UPMS) are left out of the picture in order to keep it as simple as possible (nearly all of the services need to query those two).

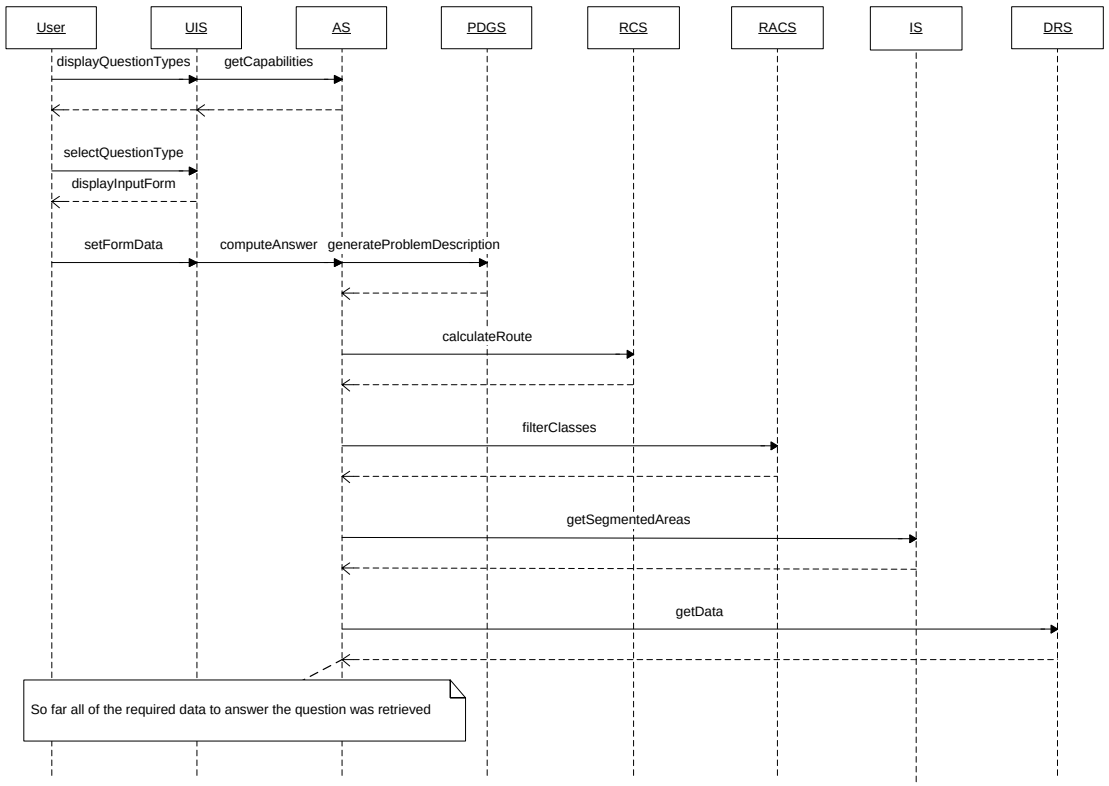


Figure 6: Sequence Diagram for Scenario 1 – part 1

In the first step, the user asks the system about the possibilities to formulate his question (displayQuestionTypes). This information is delivered by the Answer Service (AS) (getCapabilities), which offers information about the problem types that can be formulated, the input parameters required for a specific problem, and the kind of output a specific problem returns. With this information at hand, the user formulates his request by filling out a form (setFormData) and the AS is instructed to compute an answer. The AS controls the workflow which is necessary to compute the answer.

The AS requests from the Problem Description Generation Service (PDGS) a structured description (as a document formulated in PESCaDO's Problem Description Language (PDL)) of the problem submitted by the user (generateProblemDescription). It is checked that all the

required data to compile a complete problem description are available among the input parameters.

In the next steps, the route is calculated (calculateRoute) and those aspects which need to be checked to answer the question (as pointed out above, when the user asks about health issues, air quality, pollen concentrations, weather and road conditions are potentially of relevance) are to retrieved. The Related Aspects Computation Service (RACS) assesses, consulting the user profile, which of the retrieved aspects are actually relevant to the user and returns those that are relevant (filterClasses). The RCS and RACS are independent and can be executed in parallel.

The Intersection Service (IS) segments the calculated route into areas or smaller routes through the operation getSegmentedAreas. Then, the Data Retrieval Service (DRS) is requested to return the aspect measurements for each geographical area or route part (getData). For each specific service node which provides related data, the DRS returns the latest stored measurement with the geographical area covered by the node in question, the time at which the latest measurement was obtained and the data that reflect the reliability of the service node.

At this stage, all required information needed to compute an answer to the user's request has been collected.

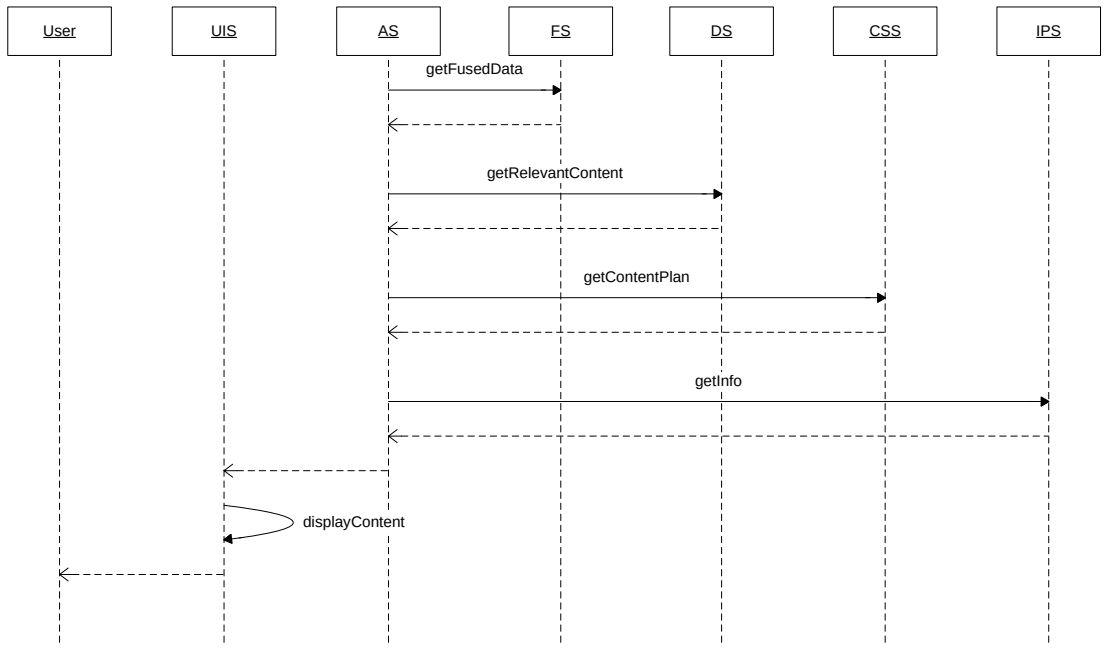


Figure 7: Sequence Diagram for Scenario 1 – part 2

In the next step, it has to be checked whether several data sources for a specific aspect are available. In this case, the values have to be fused in accordance with the result of the application of uncertainty metrics that assess the trustworthiness and precision of the values from different sources. The AS triggers the Fusion Service (FS) to construct a fused data set (getFusedData).

Then, the AS instructs the Decision Service (DS) to retrieve the content that is relevant to the solution of the user's problem (getRelevantContent). However, not all of this content should

actually be communicated to the user – be it because the content is too detailed, because it contains redundancies introduced during the reasoning activities of the DS, or because the user is already aware of some aspects. The Content Selection Service is called by the AS to select the appropriate content. The CSS delivers a content plan, i.e., a set of content elements interlinked by content relations, to be communicated to the user (getContentPlan).

The Information Production Service (IPS) is then instructed to produce user problem targeted multimodal information in one of three modes (text, graphic, table) – or in a combination of thereof – in one of the languages of PESCaDO (Finnish, English or Swedish). The generated information is displayed by the UIS to the user.

# 4 Information Viewpoint

Ontologies are foreseen to be one of the main data structures of the PESCaDO architecture. They will be used by various services of the system in order to guarantee the semantics-driven orchestration of heterogeneous environmental service nodes, user-oriented decision support, and environmental information production and delivery.

The description of a problem of the user encoded in terms of queries will be expressed in terms of the *Problem Description Language* (and, for processing purposes stored in a separate document).

The information about data nodes which contain required environmental data are represented in relational structures.

## 4.1 Data node Description

A data node structure captures the information that describes efficiently the environmental services nodes. More specifically, a data node will include information at least on the following basic aspects:

- Unique Node Identification (e.g. Uniform Resource Locator (URL))
- Environmental aspect type (e.g. air quality, temperature, etc.)
- Aspect measurement unit (e.g. °C)
- Absolute value of aspect measurement
- Time and date of the measurement
- Location in the format of geographical names and/or coordinates
- Language
- Node structure information if required (i.e. information that will facilitate future parsing of the information provided by the service node)
- Functional information of the discovery service (e.g. history of automatic queries submitted, domain specific keywords, query parameters etc.)

This information will be stored in the data node repository in order to be accessible by the data node service. The repository will be supported by a relational database implementation which will allow multiple table construction, fast indexing and SQL-based query submission.

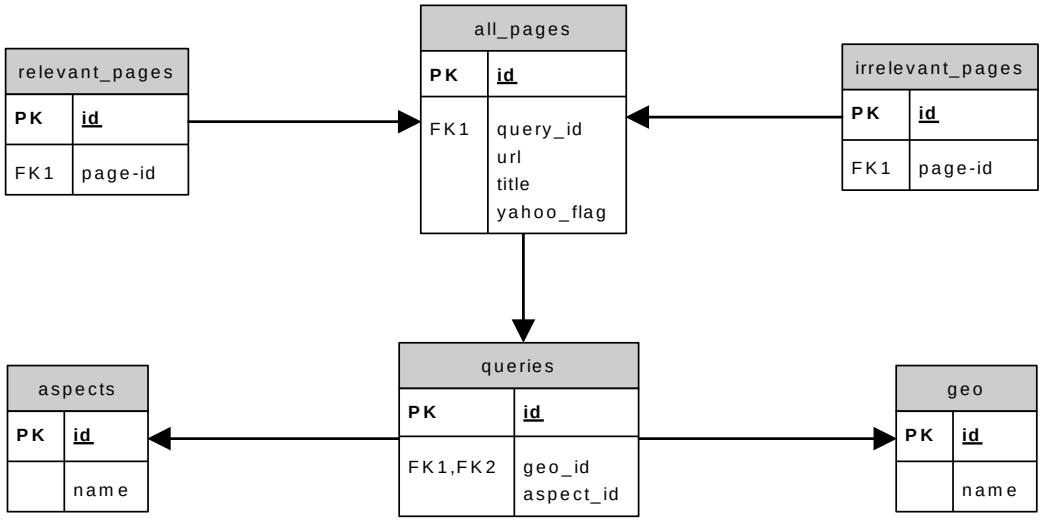


Figure 8: Example of a relational database table structure

An example of how this information is organized in a database repository is given in Figure 8. Figure 8 illustrates different tables for relevant and non-relevant pages (i.e. nodes). Additional tables store the aspect and the geographical information, the query types. The field "id" (i.e. the unique identification for each node) constitutes the connection point in this relational database structure. The description in Figure 8 is further detailed Figure 9, in which example data is presented for each of the aforementioned tables.

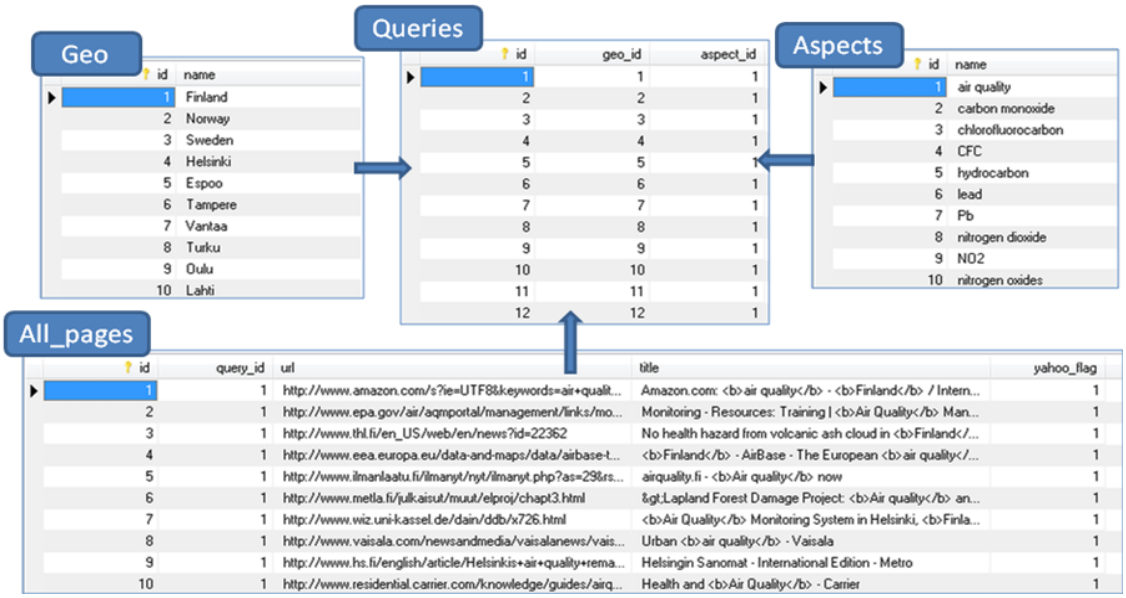


Figure 9: Tables of the relational database with example data

This structure supports fast and efficient data retrieval based on the SQL query language such that, e.g., queries like "Return the environmental nodes that provide information about the air quality of Helsinki" can be easily handled.

## 4.2 Ontologies

The PESCaDO ontology repository will comprise several ontology modules, covering different PESCaDO relevant knowledge domains. In what follows, we briefly summarize the first analysis of the PESCaDO-relevant knowledge domains (given the preliminary definition of the pilot use cases, environmental data typically available at environmental related sites, and the basic environmental related terminology/concepts). An extended and more detailed version of this section is contained in D4.1 – "Inventory of environmental ontologies and corpora from the environmental domain". At the end of the section, we provide some details of the current working version of the PESCaDO ontology.

### *4.2.1 Meteorological Conditions and Phenomena*

The meteorological conditions and phenomena domain covers concepts and attributes commonly found in weather reports and weather forecasts. Among these relevant concepts that are characteristic for this domain are: temperature, types of precipitation, intensity and types of wind, sky conditions, sun exposure and UV index, atmospheric pressure, etc.

### *4.2.2 Air Quality*

This domain covers typical concepts and attributes dealing with the quality of air. Particularly relevant are concepts related to air quality terminology, air quality indexes, air pollutants (e.g. Particulate Matter 10 ($PM_{10}$), Ozone ($O_3$), etc.), etc.

*4.2.3 Pollen*

This domain covers the typical concepts and attributes dealing with pollen and pollen counts, and, in particular, with concepts describing the main commonly monitored pollen types (e.g. grass pollens or birch pollen), and units of measurement of concentrations of pollen in the air.

*4.2.4 Travel and traffic information*

This domain deals with travel and traffic information relevant from an environmental point of view: in particular, means of transportation (e.g. car, bus or train), weather-related road conditions, traffic situation, also in connection with on-going road works, etc.

*4.2.5 Human Health*

The knowledge of this domain concerns human diseases and symptoms. In particular, environment affected and caused diseases, and environment triggered health symptoms.

Another important aspect concerns the relations between the different domains considered in PESCaDO. For instance, certain meteorological/air quality conditions may be risky or cause some symptoms in people suffering from some disease. In this case, in order to provide adequate user decision support, the PESCaDO application will also have to take into account this kind of information.

*4.2.6 Working version of the PESCaDO ontology repository*

The working version of the PESCaDO ontology repository (the Knowledge Base, KB), which is codified in the standard Semantic Web Ontology Language OWL (W3C, 2007), covers at the time of writing this deliverable environmental content such as meteorological conditions and phenomena, air quality, and pollen, as well as other relevant environment-related content essential for the targeted user-tailored service: travel and traffic information, human diseases, geographical data, monitoring station details, user profile details, etc. In addition, the KB is also capable of formally representing the description of the user inquiry (see Section 4.3)
The current version of the KB contains around 1300 classes, 115 attributes and properties, and 350 individuals. The KB has been obtained by (i) including customized version of currently available domain ontologies (e.g. SWEET Ontology[1]), (ii) automatically extracting key concepts from domain relevant text resources, and (iii) manually adding additional properties and attributes.

## 4.3 Problem Description Language (PDL)

The PESCaDO problem description language (PDL) allows a user to express a request that is to be submitted to the system. In what follows, we briefly present a first draft description of the PESCaDO PDL. The PDL is based on OWL, and its first version is fully documented in Deliverable D5.1 – “Decision support request and problem specification language”.

[1] http://sweet.jpl.nasa.gov/2.0/

The current structuring of a problem description is the result of the first analysis of the user needs for decision support as deduced from the preliminary description of the available pilot use case. Examples of user requests in this pilot use case are:

- "Is there any health issue for me to travel now by car from Helsinki-Puotila to Helsinki-Katajanokka?"
- "Do I get symptoms from pollen/air quality today?"
- "I want to go for a bike tour this Sunday in Helsinki. How does the weather forecast look like?"

A problem description that captures a user request comprises three main components:

- The activity that the user wants to perform (e.g. travel, attending an outdoor event, etc).
- The type of request: answer to a question (e.g. "Any symptoms from …"), decision support (e.g. "Is there anything I can do to…."), request for warning in case specific conditions are met (e.g. "Any warnings regarding…").
- The user who submitted the request (a reference either to the profile of the user stored in the user profile repository, or to some default profile which captures a specific category of user, e.g. asthmatic teenager).

Each of these components is described by an ontology module in the ontology repository. Figure 10 shows, as an example, an excerpt of a preliminary ontology describing some user activities:


Figure 10: Ontology excerpt showing the hierarchy of user activities

A user problem description is represented as a set of OWL ontology individuals extended by some facts about these individuals. It will contain, for instance, an individual belonging to an activity class, extended by a date/time data type property value which states when the activity is/will be performed. Below, we show an excerpt of the process description that reflects the

user question “Is it safe for me to travel now by car from Helsinki-Puotila to Helsinki-Katajanokka?”

```
<ProblemDescriptionOntology:Problem rdf:about="#problemABC">
    <ProblemDescriptionOntology:hasProblemActivity rdf:resource="#activityABC"/>
    <ProblemDescriptionOntology:hasProblemOutputFormat rdf:resource="#outputFormatABC"/>
    <ProblemDescriptionOntology:hasProblemTypeOfRequest rdf:resource="#typeOfRequestABC"/>
    <ProblemDescriptionOntology:hasProblemUser rdf:resource="#userABC"/>
</ProblemDescriptionOntology:Problem>

<ProblemDescriptionOntology:Travel rdf:about="#activityABC">
    <ProblemDescriptionOntology:hasActivityDateTime rdf:datatype="&xsd;dateTime"
        >2010-05-04T09:30:10</ProblemDescriptionOntology:hasActivityDateTime>
    <ProblemDescriptionOntology:hasActivityToLocation rdf:datatype="&xsd;string"
        >Helsinki-Katajanokka</ProblemDescriptionOntology:hasActivityToLocation>
    <ProblemDescriptionOntology:hasActivityFromLocation rdf:datatype="&xsd;string"
        >Helsinki-Puotila</ProblemDescriptionOntology:hasActivityFromLocation>
    <ProblemDescriptionOntology:hasActivityTransportation rdf:resource="#transportationABC"/>
</ProblemDescriptionOntology:Travel>

<ProblemDescriptionOntology:AnyHealthIssueFixedTransportation rdf:about="#typeOfRequestABC"/>
<ProblemDescriptionOntology:ExtendedText rdf:about="#outputFormatABC"/>
<ProblemDescriptionOntology:User rdf:about="#userABC"/>
```

# 5 Service Viewpoint

The following table shows the required services of each step in Scenario 1 (see Figure 3).

<table>
<tr><th>Step in Scenario 1</th><th>Required Services</th></tr>
<tr><td>(1) User Interface</td><td>
<h2>5.2 Content Selection Service (CSS)</h2>
<table>
<tr><td>Standard Specifications</td><td>PESCaDO's Content Selection Service uses the following standards for communication purposes with other services and modules:<ul><li>XML (Extensible Markup Language)</li><li>W3C SWRL</li><li>HTTP (Hypertext Transfer Protocol)</li><li>W3C SPARQL Query Language for RDF</li></ul></td></tr>
<tr><td>Description</td><td>The Content Selection Service selects the content elements that are to be communicated to the user from the content set provided by the Decision Service in accordance with the profile of the user and other relevance criteria. The expert user is involved in the process of content selection.<br>The Content Selection Service provides the functionality through the following interfaces:<ul><li>RetrieveContentPlan</li></ul>The Content Selection Service interacts with the following services:<ul><li>Decision Service</li><li>User Profile Management Service</li><li>Answer Service</li><li>User Interaction Service</li></ul></td></tr>
<tr><td colspan="2">Interface RetrieveContentPlan</td></tr>
</table>
</td></tr>
</table>

| | |
|---|---|
| *getContentPlan* | Provides to the Answer Service the content, i.e., a set of content elements interlinked by content relations that are considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The content provided by the Decision Service.<br>• The contextual settings, including the profile of the user.<br>The output of the function is:<br>• A content plan, i.e., a set of content elements interlinked by content relations, to be communicated to the user. |
| Comments | None at the moment. |

## 5.3 Data Retrieval Service (DRS)

*Note: The* Data Node Repository Service *(DNRS) was removed due to the fact that its functionality was moved to the* Data Retrieval Service *(DRS). The relevant operations are* getCapabilities *and* describeSensor*.*

| | |
|---|---|
| Standard Specifications | The Data Retrieval Service uses the following standards for communication purposes with other services and modules:<br>• The XML (Extensible Markup Language), which is a subset of SGML (Standard Generalized Markup Language). SGML is a widely used international text processing standard that is standardised with ISO 8879:1986(E). XML's goal is to enable generic SGML to be served, received, and processed on the Web. (http://www.w3.org/TR/2008/REC-xml-20081126/)<br>• SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment and it is based on XML. SOAP can potentially be used in combination with a variety of other protocols such as HTTP. (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)<br>• The HTTP (Hypertext Transfer Protocol) is an application-level protocol. HTTP allows an open-ended set of methods to be used to indicate the purpose of a request. It builds upon the notion of reference provided by the Uniform Resource Identifier (URI) as a location (URL) for indicating the resource on which a method is to be applied. |
| Description | The Data Retrieval Service is responsible for the retrieval of the requested information (i.e. measurements of specific aspects) for the geographical area of interest. The DRS, as shown in Figure 12, plays also the role of managing the available environmental data nodes and the Intersection |

| | | | |
|---|---|---|---|
| | | | Service in the sense of processing data requests. If the results that are obtained from the Web do not include updated measurements for the involved aspects, the service connects directly to the nodes and downloads the latest measurements.<br>DRS takes as input:<br>• Area of interest, which is described either as a single route (i.e. trajectory of movement) or as a geographical area. It is defined by a set of points described by their geographical coordinates. This information is received from the Route Calculation Service.<br>• List of types of relevant aspects (e.g. air quality, temperature, pollen). This information is retrieved by the Related Aspects Computation Service.<br>The output of the service is:<br>• Measurements of several aspects in different (probably overlapping) areas with their specific time and space values.<br>The Data Node Repository Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *DataAccess*: Enables the client to submit requests and retrieve information. |
| | | Interface *ServiceCapabilities* | |
| | | *getCapabilities* | Informs the client about the capabilities of a Data Retrieval Service instance. The capabilities include the functionalities of the service as well as the type of data that can be retrieved, which is a list of the available sensors, covered areas, time ranges and quality information. Note: The term sensor is meant in a wider sense, e.g. an "environmental data node" in the PESCaDO terms. |
| | | *describeSensor* | Get the metadata description of a sensor (the term sensor is meant in a wider sense, e.g. an "environmental data node" in the PESCaDO terms). The metadata include information like the covered area, time range for which data is available, quality and reliability of the data, keywords that describe the service according to the node provider, etc. |
| | | Interface *DataAccess* | |
| | | *getData* | Get data from sensors. Data can be filtered by aspect, attribute, time and location (area). |
| | | Example usage | In the reference use scenario, the user has the intention of travelling from A to B (Figure 3). The Route Calculation Service provides the travelling trajectory as a set of geographical coordinates with time durations; the Related Aspects Computation Service outputs temperature and humidity as the relevant aspects; the Intersection service segments the route into smaller parts. Then, the Data Retrieval Service is called recursively for each part of the route to retrieve the relevant geographical areas, the |

| | |
|---|---|
| | corresponding measurements (i.e. temperature and humidity values retrieved from the environmental nodes 1, 2, 3 in Figure 11) and the quality and reliability metrics for each node. If a value is outdated (i.e. older than a specified time threshold), e.g. the humidity value of the environmental node 3. In that case, the DRS will connect directly to the corresponding environmental node and retrieve an updated value for the humidity. Then, the Data Retrieval Service delivers three different areas with different aspect measurements (Figure 11). In this case the coverage areas are considered rectangular for convenience; however they might be updated to cyclic areas in case that such a representation is more realistic for the Fusion Service. To be noted that in the example of Figure 11 the route was not segmented to smaller parts by the intersection service. |
| Comments | It is beyond the scope of this service to provide any fusion of the obtained results. |

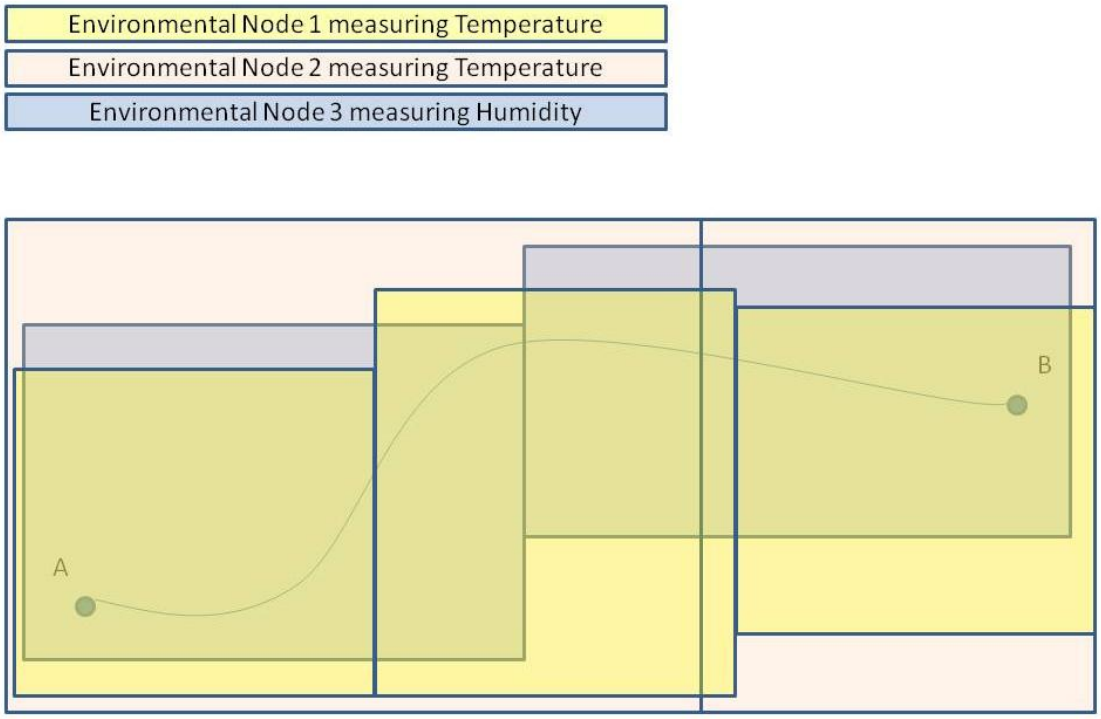


**Figure 11: Travelling from A to B and the corresponding nodes represented by their coverage area, while the colour denotes the type of the service.**

**Figure 12: Node access of the DRS**

## 5.4 Decision Service (DS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Decision Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>• PESCaDO-PDL (to become standard in PESCaDO)<br>and likely some others. |
| Description | The Decision Service retrieves the content that is relevant to the solution of the user's problem formulated in terms of PESCaDO-PDL statements. It thus<br>(i) accesses the KB in order to inspect the available knowledge elements and to determine their relevance to the problem of the user;<br>(ii) makes inferences to deduce knowledge elements not explicitly available in the KB;<br>(iii) makes inferences to deduce the necessity to solicit missing data from the Data Retrieval Service;<br>(iv) solicits data from the Data Retrieval Service.<br>The Decision Service provides its functionality through the following interfaces:<br>• *ObtainRelevantContent*. |

| | |
|---|---|
| | The Decision Service interacts with the following services:<br>• User Profile Management Service<br>• Knowledge Base Access Service<br>• Data Retrieval Service<br>• Answer Service |
| Interface *ObtainRelevantContent* | |
| *computeRelevantContent* | Provides to the Answer Service the content that is considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query which references also the contextual settings, including the profile of the user.<br>• The Knowledge Base.<br>The output of the function is:<br>• None (the decision service will write the inferred facts directly in the Knowledge Base containing the data for the current query – everything will be reachable from the instance representing the problem). |
| Comments | None at the moment. |

## 5.5 Fusion Service (FS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Fusion Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF |
| Description | The general task of the FS is to process the data received from different environmental service nodes via the Data Retrieval Service with the purpose to obtain the best and most complete data set to address the user's request for information or decision support. It thus<br>(v) identifies complementary pieces of data proceeding from different data sources of the same kind (i.e., AQ, meteorology, traffic, etc.) and fuses them into one coherent data block;<br>(vi) identifies alternative (or competing) pieces of data proceeding from different data sources, but concerning the same environmental phenomenon for |

| | |
|---|---|
| | the same location, assesses the quality of each of them using the two types of uncertainty metrics developed in PESCaDO (imprecision and confidence metrics), and selects the best for communication to the user.<br>The Fusion Service provides the functionality through the following interfaces:<br>• *FuseData*: Delivers the fused dataset.<br>The Fusion Service interacts with the following services:<br>• Data Retrieval Service<br>• Knowledge Base Access Service<br>• User Profile Management Service<br>• User Interaction Service |
| Interface *FuseData* | |
| *getFusedData* | Takes as input:<br>• The data from the data retrieval service and pointers to the corresponding service node fingerprints.<br>• Potential preferences of the user for specific service nodes.<br>The output of the function is:<br>• A fused data set. |
| Comments | None at the moment. |

## 5.6 Information Production Service (IPS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Information Production Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>and likely some others. |
| Description | The Information Production Service receives as input the Content Plan compiled by the Content Selection Service, the user profile and potentially other contextual setting restrictions and produces user problem targeted multimodal information as text, table and graphic, which is passed to the Answer Service.<br>The Information Production Service provides its functionality through the following interfaces:<br>• *Generate Information*. |

| | |
|---|---|
| | The Information Production Service interacts with the following services:<br>• User Profile Management Service<br>• Content Selection Service<br>• Answer Service<br>• User Interaction Service<br>External resources required by the Information Production service:<br>• Annotated corpora |
| Interface *GenerateInformation* | |
| *getInfo* | Takes as input a content plan, the user profile and contextual settings and provides information in one of the three modes (text, graphic, table) – or in a combination of thereof – in one of the languages of PESCaDO (Finnish, English or Swedish). |
| Comments | None |

## 5.7 Intersection Service (IS)

| | |
|---|---|
| Standard Specifications | No corresponding standard is known. |
| Description | The Intersection Service segments the route of interest into different parts (i.e. areas or smaller routes) based on ground and road condition parameters (e.g. normal traffic, width of roads, etc.). It should be noted that the parameters for segmenting a route are subject to the requirements for fusion. It takes as input:<br>• The route, which is provided by the Route Calculation Service.<br>The service processes this information in order to output non-overlapping geographical areas for the aspects of interest. In every segmented area, different measurements may be reported for the same aspect. These values are fused at a later stage by the Fusion Service.<br>The Intersection Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *SegmentationService*: This interface allows a client to make requests and receive areas of interest. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of a Data Node Repository Service instance. These include the functionalities of the service as well as the input and output specifications. |
| Interface *SegmentationService* | |
| *getSegmentedAreas* | This functionality returns the coordinates of each segmented area. |

| | |
|---|---|
| *getAspectsValues* | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Example usage | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Comments | It is beyond the scope of this service to provide any fusion of the obtained results (i.e. measurements of aspects). For each intersection area, it is possible that we have multiple measurements for the same aspect when the information is retrieved by different nodes.<br>This service can either provide direct input to the Fusion Service or return the result to the Data Retrieval Service.<br>*At the current stage of the project the exact functionality of the IS is not exactly clear as it is highly based on the requirements for fusion. Also the interaction with the Fusion Service needs further research.* |

## 5.8 Knowledge Base Access Service (KBAS)

| | |
|---|---|
| Standard Specifications | The Knowledge Base for the PESCaDO platform is currently planned to be based on the following ontology language:<br>• W3C OWL Web Ontology Language http://www.w3.org/TR/owl-features/<br>Relevant standard specification for this service are also:<br>• W3C RDFS http://www.w3.org/TR/rdf-schema/<br>• W3C RDF/XML Syntax Specification (Revised) http://www.w3.org/TR/rdf-syntax-grammar/<br>• W3C SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/<br>• W3C SWRL: A Semantic Web Rule Language Combining OWL and RuleML http://www.w3.org/Submission/SWRL/<br>• W3C URIs, URLs, and URNs: Clarifications and Recommendations 1.0 http://www.w3.org/TR/uri-clarification/ |
| Description | The Knowledge Base Access Service supports the access to the Knowledge Base on which the whole PESCaDO platform relies to provide user decision support. This service provides, via the *KBAccess* interface, access, manipulation and storage of ontologies (or ontology modules) stored in the Knowledge Base. Besides the common *KBAccess* interface retrieval methods, the *KBQuery* interface offers a generic mechanism for retrieval.<br><br>Some typical uses of this service are:<br>• Storing, updating or deleting available ontology modules;<br>• Retrieving (partially or fully) a stored ontology module; |

| | | | |
|---|---|---|---|
| | | | • Getting high-level information about some ontology module, such as the list of concepts, or the list of supported properties for a given concept;<br>• Inserting in the knowledge base actual data, e.g. those retrieved from the Data Repository, which will be used to fulfil the user problem request;<br>• Querying one of more ontology modules.<br><br>The Knowledge Base Access Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs the client about the common and specific capabilities of the Knowledge Base Access Service.<br>• *KBAccess*: Supports the management (storage, retrieval, and deletion) of ontology modules, provides a high-level view on ontologies, and allows for adding factual knowledge (data from the data repository, inferred data) to the knowledge base.<br>• *KBQuery*: Providing operations to query the ontology modules contained in the knowledge base. |
| | | Interface *ServiceCapabilities* | |
| | | *get Capabilities* | Informs the client about the common and specific capabilities of a Knowledge Base Access Service instance. Examples of specific capabilities are:<br>• The names of the ontology modules available in the Knowledge Base.<br>• The possible representation formats of query results which can be requested by clients.<br>• The query languages that can be used in knowledge base queries.<br>• The inference capabilities of the knowledge base applied when computing query results. |
| | | Interface *KBAccess* | |
| | | *setOntology* | Stores a new ontology module in the Knowledge Base. |
| | | *getOntology* | Retrieves an existing ontology module or a selection of an ontology module from the Knowledge Base. |
| | | *deleteOntology* | Removes an existing ontology module from the Knowledge Base. |
| | | *getTBox Vocabulary* | Given an ontology or a part (segment) of an ontology, returns a list of TBox statements ready to be used for creating a Knowledge Base. |
| | | *getABox Vocabulary* | Given an ontology or a part (segment) of an ontology, returns a list of ABox statements ready to be used for creating a Knowledge Base. |

| | |
|---|---|
| *addABoxStatement* | This function allows for inserting into the knowledge base the factual knowledge (data) that will be used to fulfil a user request (e.g. temperature values, pollutants concentrations). The data to be added may be, e.g. those returned by the Data Retrieval Service/Fusion Service, and the data inferred by the Decision Service. |
| *getListConcepts* | Returns a list of concepts of an ontology (module) or a part (segment) of an ontology (module). |
| Example usage | The *setOntology* operation may be used, for instance, when a new version of an ontology module has to be uploaded in the Knowledge Base. Other services of the PESCaDO platform (e.g. Fusion Service, Decision Service), may need to process all the information contained in an ontology modules to perform the operations needed to be able to invoke the *getOntology* operation. |
| Comments | Creating or editing of ontology modules is out of the scope of the Knowledge Base Access Service: this service's interface manages only the storage and the access to ontologies, but doesn't provide any tool to create ontology structures. |
| Interface *KBQuery* | |
| *queryOntology* | Submits a query to the ontologies stored in the knowledge base. The query has to be formulated in a query language (e.g. SPARQL) compatible with the knowledge representation model used by the knowledge base (e.g. RDF/OWL). |
| Example usage | The *queryOntology* operation may be invoked by other services like the Fusion Service, Decision Service or the Problem Description Generation Service to obtain the relevant information they need to take some decisions or perform some reasoning. For example, the Problem Description Generation Service may use this operation to retrieve from the corresponding ontology module in the Knowledge Base the relevant data for preparing a complete user problem description. |
| Comments | |

## 5.9 Problem Description Generation Service (PDGS)

| | |
|---|---|
| Standard Specifications | No specific corresponding standard known.<br>The Problem Description Language (PDL) is planned to be based on:<br>• W3C XML Extensible Markup Language http://www.w3.org/XML/ |
| Description | The goal of this service is to generate / check a description of the problem in PDL submitted by the user to the PESCaDO platform, according to the information provided by the user via the user interface / third party systems.<br>The Problem Description Generation Service provides the |

| | | | |
|---|---|---|---|
| | | | functionality through the following interfaces:<br>• ServiceCapabilities: Informs about the common and specific capabilities.<br>• *DescriptionGeneration*: generates a description in PDL of the problem submitted by the user to the PESCaDO platform, according to the information provided by the user via the user interface.<br>• *DescriptionChecking*: checks that a problem description expressed in PDL and submitted by third party systems is complete, i.e. no user information needed as part of the request is missing. |
| | | Interface *ServiceCapabilities* | |
| | | *getCapabilities* | Informs the client about the capabilities of a Problem Description Generation Service instance. Examples of specific capabilities are the type of problems that can be formulated, the required input parameters needed for a specific problem, and the kind of output a specific problem returns. |
| | | Interface *DescriptionGeneration* | |
| | | *generateProblemDescription* | This operation returns a structured description in PDL of the problem submitted by the user. Input parameter for this operation is the selected problem, together with a list of input data provided by the user via the user interface. The operation checks that all data needed to compile a complete problem description are available within the input parameter list. To perform this task, the operation needs to access (via the Knowledge Base Access Service) the ontology which describes the problem/data dependencies. If some required data has not been provided in the input, the operation can retrieve the missing data by accessing the User Profile via the User Profile Management Service.<br>Input parameters:<br>• The URI of the type of problem type selected by the user<br>• The URI of the type of activity selected by the user<br>• 0..n additional parameters related to the problem type<br>Returned value:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query. The PDGS will create also new instances and facts about the problem, and write the problem description directly in the Knowledge Base containing the data for the current query, but just the URI of the problem description will be returned to the AS. |
| | | Example usage | The typical application scenario for this service interface is that the *generateProblemDescription* operation is invoked by the Answer Service with the input data on the problem to be solved collected via the user interface. |
| | | Comments | None at the moment. |
| | | Interface *DescriptionChecking* | |

| | |
|---|---|
| *checkProblemDescription (optional)* | This operation checks whether a problem description expressed according to the PDL is complete (i.e. that there is no user input data missing). The input parameter for this operation is a description of a user problem in the PDL. The output of the operation is a yes/no answer. In the case of a negative answer, a description of the missing parameter can optionally also be returned. |
| Example usage | A typical application scenario for this service interface is that the *CheckProblemDescription* operation is invoked by the Answer Service in the case of user decision support requests not submitted via the PESCaDO platform user interface, like e.g. batch submission of third party organisation / system. |
| Comments | None at the moment. |

## 5.10 Related Aspects Computation Service (RACS)

| | |
|---|---|
| Standard Specifications | No specific corresponding standard known.<br>(Data format to query the knowledge base?) |
| Description | The Related Aspects Computation Service (RACS) is a service that, given the problem description generated for the request made by the user, selects the aspects that need to be considered for delivering an informative answer to the user.<br>The input of the service is planned to be an *action specification* (e.g. “travel from X to Y”) and the *query focus* (or topic, e.g. “health issues”) submitted by the user, as encoded in the PDL (Problem Description Language) format by the Problem Description Generation Service. The output is a set of focus-related aspects that have to be considered before an answer to the user query is given.<br>Given the action and focus of the user query, the service accesses the Knowledge Base containing general information about aspects relevant to an <action, focus> couple. The Knowledge Base returns a list of relevant aspects, with a possibly empty list of specific constraints that need to be verified to assess the relevance of an aspect for a specific user associated with each of the aspects in question. The RACS will filter the list of aspects returned by the Knowledge Base by matching the constraints associated with each aspect against the information contained in the user profile.<br>The RACS provides the functionality through the following interfaces:<br>• *RelevantAspects*: accesses the knowledge base retrieving all aspects relevant to the focus associated with the query.<br>• *InformationFiltering*: compares the focus-related aspects with the user profile and filters them in order to keep only the relevant ones. |
| Interface *RelevantAspects* | |
| *getRelevantAspects* | Given the action and focus of the user query, the operation |

| | |
|---|---|
| | returns a set of related aspects. Each aspect can have an associated list of relevance constraints to be matched against information contained in the user profile.<br><br>Input parameters:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query;<br><br>Returns a list of couples (0..n):<br>• The URI of the entity in the KB corresponding to a specific aspect (e.g. http://www.pescado-project.eu/ontology/pescado.owl#precipitation or http://www.pescado-project.eu/ontology/pescado.owl#O3)<br>• The URI of the class in the KB representing the type of the aspect (e.g. http://www.pescado-project.eu/ontology/pescado.owl#AirQualityType or http://www.pescado-project.eu/ontology/pescado.owl#WeatherType) |
| Interface *InformationFiltering* | |
| *filterClasses* | The aspects retrieved from the *RelevantAspects* interface are compared against the user profile information and only those that are relevant to the user are returned. |
| Example usage | A requestor wants to know if it is safe for him to travel from place X to destination Y. The user profile of the requestor says that he is a middle-age man with allergy to pollen and no other disease. RACS is asked to find aspects which are relevant to the requestor given the action “travel from X to Y” and the focus “health issues”. RACS accesses the Knowledge Base and finds that generally speaking “pollen” and “heat” are relevant aspects for a person that wants to travel and is concerned about health. The Knowledge Base also provides the information that “pollen” aspects are relevant only to people who have pollen allergies, and the “heat” aspect is relevant only if the user is old or has heart diseases. The RACS accesses the user profile and finds that the “allergy” constraint is matched (the user is allergic to pollen), whereas the “aged or heart disease” condition is not matched. The heat factor is not dangerous and can be filtered out. RACS outputs the pollen aspect as the only one which is relevant to the requestor and that needs to be further analyzed. |
| Comments | None at the moment. |

## 5.11 Route Calculation Service (RCS)

| | |
|---|---|
| Standard Specifications | Relevant standard specification for this service are:<br>• OpenGIS Geography Markup Language (GML) Encoding Standard<br>• Open Street Map http://www.openstreetmap.org/ |

| | |
|---|---|
| Description | The Route Calculation Service provides a range of services based on (freely) available spatial data, e.g. data provided by Open Street Map.<br>The Route Calculation Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities:* Informs about the common and specific capabilities.<br>• *LocationUtility*: Provides a Geocoder/Reverse Geocoder.<br>• *RouteCalculation:* Determines travel routes and navigation information. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of the Route Calculation Service, e.g. the supported output formats of calculated routes (e.g. GML/XML, GPX). |
| Interface *LocationUtility* | |
| *geoEncode* | The geoEncode operation transforms a description of a location, such as a place name, street address or postal code, into a normalized description of the location as point geometry. |
| *geoDecode* | The geoDecode operation transforms a normalized description of the location into a description of a location. |
| Interface *RouteCalculation* | |
| *calculateRoute* | The calculateRoute operation receives as input a start point and an end point. Moreover, diverse criteria such as the travelling means can be entered, e.g.<br>• car: fastest<br>• car: shortest<br>• bicycle<br>• walking<br>The operation then calculates automatically the route and returns it in the supported output format for further processing by other services or for rendering on a map by means of a user interface client. |
| Example usage | In a typical scenario, a user submits a request to the *Answer Service* that he/she wants to travel from point A to point B. To check the areas that the user will cross during his journey for potential health related aspects (e.g. pollen in case of an allergy, dangerous roads in winter time, etc.), the route must be known first. |
| Comments | • Further parameters which could be supplied to the calculateRoute operation (e.g. transit-stops, avoidance of motorways, avoidance of areas, etc.) need further discussion. Optionally, there should be a helper function for auto completion of road and city names.<br>• Ongoing research in PESCaDO might show that it would be better or sufficient not to have a route calculation but instead have a list of affected areas. To facilitate this option, e.g. the knowledge base must contain an area |

| | |
|---|---|
| | map (of Finland). |

## 5.12 User Interaction Service (UIS)

| | |
|---|---|
| Standard Specifications | PESCaDO's User Interaction Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• HTTP (Hypertext Transfer Protocol)<br>and likely some others. |
| Description | The User Interaction Service has two main functions. First, it ensures the communication between the expert and end users on the one side and the services that involve the intervention of the user for their optimal functioning on the other side. For the first version of the architecture of PESCaDO, these are: (i) the Fusion Service; (ii) the Content Selection Service, and the (iii) the Information Production Service. In the subsequent versions, the Data Node Retrieval Service will also make use of the UIS.<br>Second, it supports the communication with the end user and the Answer Service.<br>The User Interaction Service provides its functionality through the following interfaces:<br>• *FusionAndMetricsSupport*<br>• *ContentSelectionSupport*<br>• *DiscoursePlanningSupport*<br>• *AnswerServiceSupport*<br>The User Interaction Service interacts with the following services:<br>• Fusion Service<br>• Content selection Service<br>• Information Production Service<br>• Answer Service |
| Interface *FusionAndMetricsSupport* | |
| *displayMetrics* | Visualizes the metrics to the user. |
| *displayFusionSchema* | Visualizes to the user the picture of how data from specified data sources are fused to obtain the optimal value. |
| *modifyMetrics* | Allows the user to modify (in a graphical mode) the uncertainty / confidence metrics. |
| *modifyFusionSchema* | Allows the user to modify (in a graphical mode) the data fusion schema. |
| Interface *ContentSelectionSupport* | |

| | | | |
|---|---|---|---|
| | | *displayContent* | Displays a content element or the whole content plan to the user. |
| | | *getRelevance* | Requests the judgement of the user concerning the relevance of a content element (can be positive or negative). |
| | | Interface *DiscoursePlanningSupport* | |
| | | *displayDiscourse* | Displays a discourse structure or a discourse element to the user. |
| | | *getDiscourseRelation* | Requests the assignment of a discourse relation to a pair of content elements (the content elements can be complex). |
| | | *orderDiscourse* | Requests the (linear) ordering of several discourse elements among each other. |
| | | Comments | The offered functionality of the Interface *FusionAndMetricsSupport* is to be defined. |
| | | Interface *AnswerServiceSupport* | |
| | | *displayQuestionTypes* | Displays the types of inquiries a user can submit to the system |
| | | *selectQuestionType* | Prompts the user for the selection of one specific query type |
| | | *displayInputFormat* | Shows the format in which the user has to provide their problem description. |
| | | *setFormData* | Prompts the user for entering the data of its problem description. |
| | | Comments | None at the moment. |
| | 5.13 User Profile Management Service (UPMS)<br>5.12 User Interaction Service (UIS) | | |
| (2) Answer (health) question | 5.1 Answer Service (AS) | | |
| (3) Generate Pattern Description | 5.2 Content Selection Service (CSS) | | |

| | |
|---|---|
| Standard Specifications | PESCaDO's Content Selection Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF |
| Description | The Content Selection Service selects the content elements that are to be communicated to the user from the content set provided by the Decision Service in accordance with the profile of the user and other relevance criteria. The expert user is involved in the process of content selection.<br>The Content Selection Service provides the functionality through the following interfaces:<br>• *RetrieveContentPlan* |

| | |
|---|---|
| | The Content Selection Service interacts with the following services:<br>• Decision Service<br>• User Profile Management Service<br>• Answer Service<br>• User Interaction Service |
| Interface *RetrieveContentPlan* | |
| *getContentPlan* | Provides to the Answer Service the content, i.e., a set of content elements interlinked by content relations that are considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The content provided by the Decision Service.<br>• The contextual settings, including the profile of the user.<br>The output of the function is:<br>• A content plan, i.e., a set of content elements interlinked by content relations, to be communicated to the user. |
| Comments | None at the moment. |

## 5.3 Data Retrieval Service (DRS)

*Note: The* Data Node Repository Service *(DNRS) was removed due to the fact that its functionality was moved to the* Data Retrieval Service *(DRS). The relevant operations are* getCapabilities *and* describeSensor.

| | |
|---|---|
| Standard Specifications | The Data Retrieval Service uses the following standards for communication purposes with other services and modules:<br>• The XML (Extensible Markup Language), which is a subset of SGML (Standard Generalized Markup Language). SGML is a widely used international text processing standard that is standardised with ISO 8879:1986(E). XML's goal is to enable generic SGML to be served, received, and processed on the Web. (http://www.w3.org/TR/2008/REC-xml-20081126/)<br>• SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment and it is based on XML. SOAP can potentially be used in combination with a variety of other protocols such as HTTP. (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)<br>• The HTTP (Hypertext Transfer Protocol) is an application-level protocol. HTTP allows an open-ended set of methods to be used to indicate the purpose of a |

| | | | |
|---|---|---|---|
| | | | request. It builds upon the notion of reference provided by the Uniform Resource Identifier (URI) as a location (URL) for indicating the resource on which a method is to be applied. |
| | | Description | The Data Retrieval Service is responsible for the retrieval of the requested information (i.e. measurements of specific aspects) for the geographical area of interest. The DRS, as shown in Figure 12, plays also the role of managing the available environmental data nodes and the Intersection Service in the sense of processing data requests. If the results that are obtained from the Web do not include updated measurements for the involved aspects, the service connects directly to the nodes and downloads the latest measurements.<br>DRS takes as input:<br>• Area of interest, which is described either as a single route (i.e. trajectory of movement) or as a geographical area. It is defined by a set of points described by their geographical coordinates. This information is received from the Route Calculation Service.<br>• List of types of relevant aspects (e.g. air quality, temperature, pollen). This information is retrieved by the Related Aspects Computation Service.<br>The output of the service is:<br>• Measurements of several aspects in different (probably overlapping) areas with their specific time and space values.<br>The Data Node Repository Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *DataAccess*: Enables the client to submit requests and retrieve information. |
| | | Interface *ServiceCapabilities* | |
| | | *getCapabilities* | Informs the client about the capabilities of a Data Retrieval Service instance. The capabilities include the functionalities of the service as well as the type of data that can be retrieved, which is a list of the available sensors, covered areas, time ranges and quality information. Note: The term sensor is meant in a wider sense, e.g. an “environmental data node” in the PESCaDO terms. |
| | | *describeSensor* | Get the metadata description of a sensor (the term sensor is meant in a wider sense, e.g. an “environmental data node” in the PESCaDO terms). The metadata include information like the covered area, time range for which data is available, quality and reliability of the data, keywords that describe the service according to the node provider, etc. |
| | | Interface *DataAccess* | |
| | | *getData* | Get data from sensors. Data can be filtered by aspect, attribute, time and location (area). |

| | |
|---|---|
| Example usage | In the reference use scenario, the user has the intention of travelling from A to B (Figure 3). The Route Calculation Service provides the travelling trajectory as a set of geographical coordinates with time durations; the Related Aspects Computation Service outputs temperature and humidity as the relevant aspects; the Intersection service segments the route into smaller parts. Then, the Data Retrieval Service is called recursively for each part of the route to retrieve the relevant geographical areas, the corresponding measurements (i.e. temperature and humidity values retrieved from the environmental nodes 1, 2, 3 in Figure 11) and the quality and reliability metrics for each node. If a value is outdated (i.e. older than a specified time threshold), e.g. the humidity value of the environmental node 3. In that case, the DRS will connect directly to the corresponding environmental node and retrieve an updated value for the humidity. Then, the Data Retrieval Service delivers three different areas with different aspect measurements (Figure 11). In this case the coverage areas are considered rectangular for convenience; however they might be updated to cyclic areas in case that such a representation is more realistic for the Fusion Service. To be noted that in the example of Figure 11 the route was not segmented to smaller parts by the intersection service. |
| Comments | It is beyond the scope of this service to provide any fusion of the obtained results. |

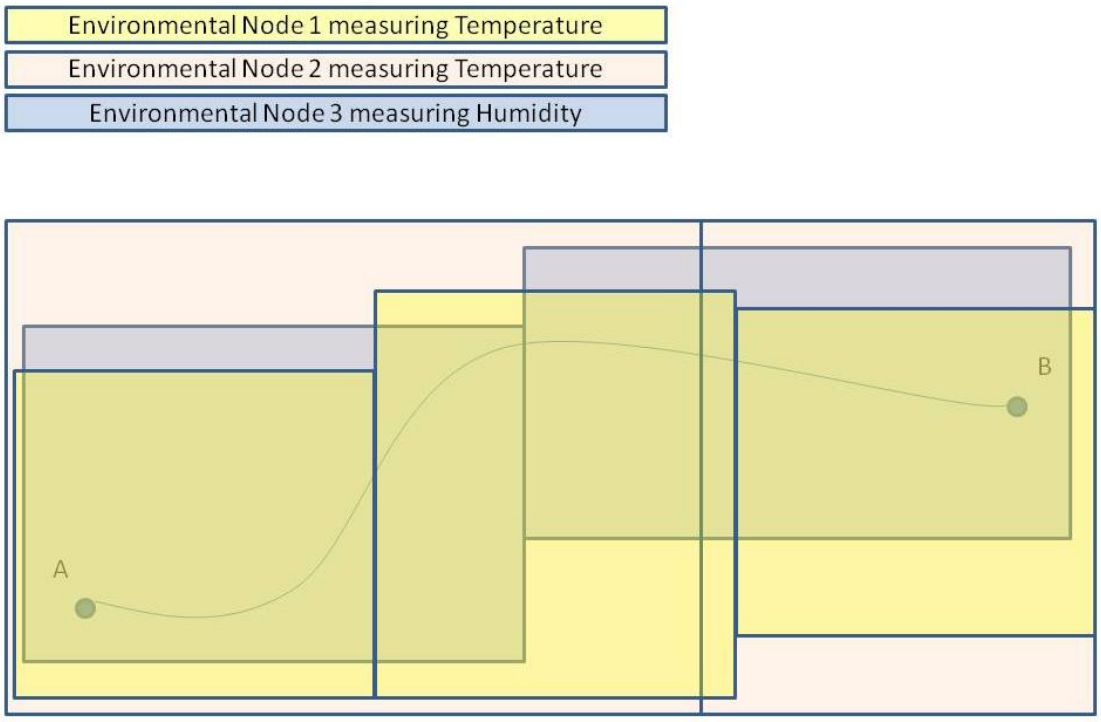


**Figure 11: Travelling from A to B and the corresponding nodes represented by their coverage area, while the colour denotes the type of the service.**

**Figure 12: Node access of the DRS**

## 5.4 Decision Service (DS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Decision Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>• PESCaDO-PDL (to become standard in PESCaDO)<br>and likely some others. |
| Description | The Decision Service retrieves the content that is relevant to the solution of the user's problem formulated in terms of PESCaDO-PDL statements. It thus<br>(vii) accesses the KB in order to inspect the available knowledge elements and to determine their relevance to the problem of the user;<br>(viii) makes inferences to deduce knowledge elements not explicitly available in the KB;<br>(ix) makes inferences to deduce the necessity to solicit missing data from the Data Retrieval Service;<br>(x) solicits data from the Data Retrieval Service.<br>The Decision Service provides its functionality through the following interfaces:<br>• *ObtainRelevantContent*. |

| | |
|---|---|
| | The Decision Service interacts with the following services:<br>• User Profile Management Service<br>• Knowledge Base Access Service<br>• Data Retrieval Service<br>• Answer Service |
| Interface *ObtainRelevantContent* | |
| *computeRelevantContent* | Provides to the Answer Service the content that is considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query which references also the contextual settings, including the profile of the user.<br>• The Knowledge Base.<br>The output of the function is:<br>• None (the decision service will write the inferred facts directly in the Knowledge Base containing the data for the current query – everything will be reachable from the instance representing the problem). |
| Comments | None at the moment. |

## 5.5 Fusion Service (FS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Fusion Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF |
| Description | The general task of the FS is to process the data received from different environmental service nodes via the Data Retrieval Service with the purpose to obtain the best and most complete data set to address the user's request for information or decision support. It thus<br>(xi) identifies complementary pieces of data proceeding from different data sources of the same kind (i.e., AQ, meteorology, traffic, etc.) and fuses them into one coherent data block;<br>(xii) identifies alternative (or competing) pieces of data proceeding from different data sources, but concerning the same environmental phenomenon for |

| | |
|---|---|
| | the same location, assesses the quality of each of them using the two types of uncertainty metrics developed in PESCaDO (imprecision and confidence metrics), and selects the best for communication to the user.<br>The Fusion Service provides the functionality through the following interfaces:<br>• *FuseData*: Delivers the fused dataset.<br>The Fusion Service interacts with the following services:<br>• Data Retrieval Service<br>• Knowledge Base Access Service<br>• User Profile Management Service<br>• User Interaction Service |
| Interface *FuseData* | |
| *getFusedData* | Takes as input:<br>• The data from the data retrieval service and pointers to the corresponding service node fingerprints.<br>• Potential preferences of the user for specific service nodes.<br>The output of the function is:<br>• A fused data set. |
| Comments | None at the moment. |

## 5.6 Information Production Service (IPS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Information Production Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>and likely some others. |
| Description | The Information Production Service receives as input the Content Plan compiled by the Content Selection Service, the user profile and potentially other contextual setting restrictions and produces user problem targeted multimodal information as text, table and graphic, which is passed to the Answer Service.<br>The Information Production Service provides its functionality through the following interfaces:<br>• *Generate Information*. |

| | |
|---|---|
| | The Information Production Service interacts with the following services:<br>• User Profile Management Service<br>• Content Selection Service<br>• Answer Service<br>• User Interaction Service<br>External resources required by the Information Production service:<br>• Annotated corpora |
| Interface *GenerateInformation* | |
| *getInfo* | Takes as input a content plan, the user profile and contextual settings and provides information in one of the three modes (text, graphic, table) – or in a combination of thereof – in one of the languages of PESCaDO (Finnish, English or Swedish). |
| Comments | None |

## 5.7 Intersection Service (IS)

| | |
|---|---|
| Standard Specifications | No corresponding standard is known. |
| Description | The Intersection Service segments the route of interest into different parts (i.e. areas or smaller routes) based on ground and road condition parameters (e.g. normal traffic, width of roads, etc.). It should be noted that the parameters for segmenting a route are subject to the requirements for fusion. It takes as input:<br>• The route, which is provided by the Route Calculation Service.<br>The service processes this information in order to output non-overlapping geographical areas for the aspects of interest. In every segmented area, different measurements may be reported for the same aspect. These values are fused at a later stage by the Fusion Service.<br>The Intersection Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *SegmentationService*: This interface allows a client to make requests and receive areas of interest. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of a Data Node Repository Service instance. These include the functionalities of the service as well as the input and output specifications. |
| Interface *SegmentationService* | |
| *getSegmentedAreas* | This functionality returns the coordinates of each segmented area. |

| | |
|---|---|
| *getAspectsValues* | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Example usage | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Comments | It is beyond the scope of this service to provide any fusion of the obtained results (i.e. measurements of aspects). For each intersection area, it is possible that we have multiple measurements for the same aspect when the information is retrieved by different nodes.<br>This service can either provide direct input to the Fusion Service or return the result to the Data Retrieval Service.<br>*At the current stage of the project the exact functionality of the IS is not exactly clear as it is highly based on the requirements for fusion. Also the interaction with the Fusion Service needs further research.* |

## 5.8 Knowledge Base Access Service (KBAS)

| | |
|---|---|
| Standard Specifications | The Knowledge Base for the PESCaDO platform is currently planned to be based on the following ontology language:<br>• W3C OWL Web Ontology Language http://www.w3.org/TR/owl-features/<br>Relevant standard specification for this service are also:<br>• W3C RDFS http://www.w3.org/TR/rdf-schema/<br>• W3C RDF/XML Syntax Specification (Revised) http://www.w3.org/TR/rdf-syntax-grammar/<br>• W3C SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/<br>• W3C SWRL: A Semantic Web Rule Language Combining OWL and RuleML http://www.w3.org/Submission/SWRL/<br>• W3C URIs, URLs, and URNs: Clarifications and Recommendations 1.0 http://www.w3.org/TR/uri-clarification/ |
| Description | The Knowledge Base Access Service supports the access to the Knowledge Base on which the whole PESCaDO platform relies to provide user decision support. This service provides, via the *KBAccess* interface, access, manipulation and storage of ontologies (or ontology modules) stored in the Knowledge Base. Besides the common *KBAccess* interface retrieval methods, the *KBQuery* interface offers a generic mechanism for retrieval.<br><br>Some typical uses of this service are:<br>• Storing, updating or deleting available ontology modules;<br>• Retrieving (partially or fully) a stored ontology module; |

| | | | |
|---|---|---|---|
| | | | • Getting high-level information about some ontology module, such as the list of concepts, or the list of supported properties for a given concept;<br>• Inserting in the knowledge base actual data, e.g. those retrieved from the Data Repository, which will be used to fulfil the user problem request;<br>• Querying one of more ontology modules.<br><br>The Knowledge Base Access Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs the client about the common and specific capabilities of the Knowledge Base Access Service.<br>• *KBAccess*: Supports the management (storage, retrieval, and deletion) of ontology modules, provides a high-level view on ontologies, and allows for adding factual knowledge (data from the data repository, inferred data) to the knowledge base.<br>• *KBQuery*: Providing operations to query the ontology modules contained in the knowledge base. |
| | | Interface *ServiceCapabilities* | |
| | | *get Capabilities* | Informs the client about the common and specific capabilities of a Knowledge Base Access Service instance. Examples of specific capabilities are:<br>• The names of the ontology modules available in the Knowledge Base.<br>• The possible representation formats of query results which can be requested by clients.<br>• The query languages that can be used in knowledge base queries.<br>• The inference capabilities of the knowledge base applied when computing query results. |
| | | Interface *KBAccess* | |
| | | *setOntology* | Stores a new ontology module in the Knowledge Base. |
| | | *getOntology* | Retrieves an existing ontology module or a selection of an ontology module from the Knowledge Base. |
| | | *deleteOntology* | Removes an existing ontology module from the Knowledge Base. |
| | | *getTBox Vocabulary* | Given an ontology or a part (segment) of an ontology, returns a list of TBox statements ready to be used for creating a Knowledge Base. |
| | | *getABox Vocabulary* | Given an ontology or a part (segment) of an ontology, returns a list of ABox statements ready to be used for creating a Knowledge Base. |

| | |
|---|---|
| *addABoxStatement* | This function allows for inserting into the knowledge base the factual knowledge (data) that will be used to fulfil a user request (e.g. temperature values, pollutants concentrations). The data to be added may be, e.g. those returned by the Data Retrieval Service/Fusion Service, and the data inferred by the Decision Service. |
| *getListConcepts* | Returns a list of concepts of an ontology (module) or a part (segment) of an ontology (module). |
| Example usage | The *setOntology* operation may be used, for instance, when a new version of an ontology module has to be uploaded in the Knowledge Base. Other services of the PESCaDO platform (e.g. Fusion Service, Decision Service), may need to process all the information contained in an ontology modules to perform the operations needed to be able to invoke the *getOntology* operation. |
| Comments | Creating or editing of ontology modules is out of the scope of the Knowledge Base Access Service: this service's interface manages only the storage and the access to ontologies, but doesn't provide any tool to create ontology structures. |
| Interface *KBQuery* | |
| *queryOntology* | Submits a query to the ontologies stored in the knowledge base. The query has to be formulated in a query language (e.g. SPARQL) compatible with the knowledge representation model used by the knowledge base (e.g. RDF/OWL). |
| Example usage | The *queryOntology* operation may be invoked by other services like the Fusion Service, Decision Service or the Problem Description Generation Service to obtain the relevant information they need to take some decisions or perform some reasoning. For example, the Problem Description Generation Service may use this operation to retrieve from the corresponding ontology module in the Knowledge Base the relevant data for preparing a complete user problem description. |
| Comments | |

## 5.9 Problem Description Generation Service (PDGS)

| | |
|---|---|
| Standard Specifications | No specific corresponding standard known.<br>The Problem Description Language (PDL) is planned to be based on:<br>• W3C XML Extensible Markup Language http://www.w3.org/XML/ |
| Description | The goal of this service is to generate / check a description of the problem in PDL submitted by the user to the PESCaDO platform, according to the information provided by the user via the user interface / third party systems.<br>The Problem Description Generation Service provides the |

| | | | |
|---|---|---|---|
| | | | functionality through the following interfaces:<br>• ServiceCapabilities: Informs about the common and specific capabilities.<br>• *DescriptionGeneration*: generates a description in PDL of the problem submitted by the user to the PESCaDO platform, according to the information provided by the user via the user interface.<br>• *DescriptionChecking*: checks that a problem description expressed in PDL and submitted by third party systems is complete, i.e. no user information needed as part of the request is missing. |
| | | Interface *ServiceCapabilities* | |
| | | *getCapabilities* | Informs the client about the capabilities of a Problem Description Generation Service instance. Examples of specific capabilities are the type of problems that can be formulated, the required input parameters needed for a specific problem, and the kind of output a specific problem returns. |
| | | Interface *DescriptionGeneration* | |
| | | *generateProblemDescription* | This operation returns a structured description in PDL of the problem submitted by the user. Input parameter for this operation is the selected problem, together with a list of input data provided by the user via the user interface. The operation checks that all data needed to compile a complete problem description are available within the input parameter list. To perform this task, the operation needs to access (via the Knowledge Base Access Service) the ontology which describes the problem/data dependencies. If some required data has not been provided in the input, the operation can retrieve the missing data by accessing the User Profile via the User Profile Management Service.<br>Input parameters:<br>• The URI of the type of problem type selected by the user<br>• The URI of the type of activity selected by the user<br>• 0..n additional parameters related to the problem type<br>Returned value:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query. The PDGS will create also new instances and facts about the problem, and write the problem description directly in the Knowledge Base containing the data for the current query, but just the URI of the problem description will be returned to the AS. |
| | | Example usage | The typical application scenario for this service interface is that the *generateProblemDescription* operation is invoked by the Answer Service with the input data on the problem to be solved collected via the user interface. |
| | | Comments | None at the moment. |
| | | Interface *DescriptionChecking* | |

| | |
|---|---|
| *checkProblemDescription (optional)* | This operation checks whether a problem description expressed according to the PDL is complete (i.e. that there is no user input data missing). The input parameter for this operation is a description of a user problem in the PDL. The output of the operation is a yes/no answer. In the case of a negative answer, a description of the missing parameter can optionally also be returned. |
| Example usage | A typical application scenario for this service interface is that the *CheckProblemDescription* operation is invoked by the Answer Service in the case of user decision support requests not submitted via the PESCaDO platform user interface, like e.g. batch submission of third party organisation / system. |
| Comments | None at the moment. |

## 5.10 Related Aspects Computation Service (RACS)

| | |
|---|---|
| Standard Specifications | No specific corresponding standard known.<br>(Data format to query the knowledge base?) |
| Description | The Related Aspects Computation Service (RACS) is a service that, given the problem description generated for the request made by the user, selects the aspects that need to be considered for delivering an informative answer to the user.<br>The input of the service is planned to be an *action specification* (e.g. “travel from X to Y”) and the *query focus* (or topic, e.g. “health issues”) submitted by the user, as encoded in the PDL (Problem Description Language) format by the Problem Description Generation Service. The output is a set of focus-related aspects that have to be considered before an answer to the user query is given.<br>Given the action and focus of the user query, the service accesses the Knowledge Base containing general information about aspects relevant to an <action, focus> couple. The Knowledge Base returns a list of relevant aspects, with a possibly empty list of specific constraints that need to be verified to assess the relevance of an aspect for a specific user associated with each of the aspects in question. The RACS will filter the list of aspects returned by the Knowledge Base by matching the constraints associated with each aspect against the information contained in the user profile.<br>The RACS provides the functionality through the following interfaces:<br>• *RelevantAspects*: accesses the knowledge base retrieving all aspects relevant to the focus associated with the query.<br>• *InformationFiltering*: compares the focus-related aspects with the user profile and filters them in order to keep only the relevant ones. |
| Interface *RelevantAspects* | |
| *getRelevantAspects* | Given the action and focus of the user query, the operation |

| | |
|---|---|
| | returns a set of related aspects. Each aspect can have an associated list of relevance constraints to be matched against information contained in the user profile.<br><br>Input parameters:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query;<br><br>Returns a list of couples (0..n):<br>• The URI of the entity in the KB corresponding to a specific aspect (e.g. http://www.pescado-project.eu/ontology/pescado.owl#precipitation or http://www.pescado-project.eu/ontology/pescado.owl#O3)<br>• The URI of the class in the KB representing the type of the aspect (e.g. http://www.pescado-project.eu/ontology/pescado.owl#AirQualityType or http://www.pescado-project.eu/ontology/pescado.owl#WeatherType) |
| Interface *InformationFiltering* | |
| *filterClasses* | The aspects retrieved from the *RelevantAspects* interface are compared against the user profile information and only those that are relevant to the user are returned. |
| Example usage | A requestor wants to know if it is safe for him to travel from place X to destination Y. The user profile of the requestor says that he is a middle-age man with allergy to pollen and no other disease. RACS is asked to find aspects which are relevant to the requestor given the action “travel from X to Y” and the focus “health issues”. RACS accesses the Knowledge Base and finds that generally speaking “pollen” and “heat” are relevant aspects for a person that wants to travel and is concerned about health. The Knowledge Base also provides the information that “pollen” aspects are relevant only to people who have pollen allergies, and the “heat” aspect is relevant only if the user is old or has heart diseases. The RACS accesses the user profile and finds that the “allergy” constraint is matched (the user is allergic to pollen), whereas the “aged or heart disease” condition is not matched. The heat factor is not dangerous and can be filtered out. RACS outputs the pollen aspect as the only one which is relevant to the requestor and that needs to be further analyzed. |
| Comments | None at the moment. |

## 5.11 Route Calculation Service (RCS)

| | |
|---|---|
| Standard Specifications | Relevant standard specification for this service are:<br>• OpenGIS Geography Markup Language (GML) Encoding Standard<br>• Open Street Map http://www.openstreetmap.org/ |

| | |
|---|---|
| Description | The Route Calculation Service provides a range of services based on (freely) available spatial data, e.g. data provided by Open Street Map.<br>The Route Calculation Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities:* Informs about the common and specific capabilities.<br>• *LocationUtility*: Provides a Geocoder/Reverse Geocoder.<br>• *RouteCalculation:* Determines travel routes and navigation information. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of the Route Calculation Service, e.g. the supported output formats of calculated routes (e.g. GML/XML, GPX). |
| Interface *LocationUtility* | |
| *geoEncode* | The geoEncode operation transforms a description of a location, such as a place name, street address or postal code, into a normalized description of the location as point geometry. |
| *geoDecode* | The geoDecode operation transforms a normalized description of the location into a description of a location. |
| Interface *RouteCalculation* | |
| *calculateRoute* | The calculateRoute operation receives as input a start point and an end point. Moreover, diverse criteria such as the travelling means can be entered, e.g.<br>• car: fastest<br>• car: shortest<br>• bicycle<br>• walking<br>The operation then calculates automatically the route and returns it in the supported output format for further processing by other services or for rendering on a map by means of a user interface client. |
| Example usage | In a typical scenario, a user submits a request to the *Answer Service* that he/she wants to travel from point A to point B. To check the areas that the user will cross during his journey for potential health related aspects (e.g. pollen in case of an allergy, dangerous roads in winter time, etc.), the route must be known first. |
| Comments | • Further parameters which could be supplied to the calculateRoute operation (e.g. transit-stops, avoidance of motorways, avoidance of areas, etc.) need further discussion. Optionally, there should be a helper function for auto completion of road and city names.<br>• Ongoing research in PESCaDO might show that it would be better or sufficient not to have a route calculation but instead have a list of affected areas. To facilitate this option, e.g. the knowledge base must contain an area |

| | |
|---|---|
| | map (of Finland). |

## 5.12 User Interaction Service (UIS)

| | |
|---|---|
| Standard Specifications | PESCaDO's User Interaction Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• HTTP (Hypertext Transfer Protocol)<br>and likely some others. |
| Description | The User Interaction Service has two main functions. First, it ensures the communication between the expert and end users on the one side and the services that involve the intervention of the user for their optimal functioning on the other side. For the first version of the architecture of PESCaDO, these are: (i) the Fusion Service; (ii) the Content Selection Service, and the (iii) the Information Production Service. In the subsequent versions, the Data Node Retrieval Service will also make use of the UIS.<br>Second, it supports the communication with the end user and the Answer Service.<br>The User Interaction Service provides its functionality through the following interfaces:<br>• *FusionAndMetricsSupport*<br>• *ContentSelectionSupport*<br>• *DiscoursePlanningSupport*<br>• *AnswerServiceSupport*<br>The User Interaction Service interacts with the following services:<br>• Fusion Service<br>• Content selection Service<br>• Information Production Service<br>• Answer Service |
| Interface *FusionAndMetricsSupport* | |
| *displayMetrics* | Visualizes the metrics to the user. |
| *displayFusionSchema* | Visualizes to the user the picture of how data from specified data sources are fused to obtain the optimal value. |
| *modifyMetrics* | Allows the user to modify (in a graphical mode) the uncertainty / confidence metrics. |
| *modifyFusionSchema* | Allows the user to modify (in a graphical mode) the data fusion schema. |
| Interface *ContentSelectionSupport* | |

| | | |
|---|---|---|
| | *displayContent* | Displays a content element or the whole content plan to the user. |
| | *getRelevance* | Requests the judgement of the user concerning the relevance of a content element (can be positive or negative). |
| | Interface *DiscoursePlanningSupport* | |
| | *displayDiscourse* | Displays a discourse structure or a discourse element to the user. |
| | *getDiscourseRelation* | Requests the assignment of a discourse relation to a pair of content elements (the content elements can be complex). |
| | *orderDiscourse* | Requests the (linear) ordering of several discourse elements among each other. |
| | Comments | The offered functionality of the Interface *FusionAndMetricsSupport* is to be defined. |
| | Interface *AnswerServiceSupport* | |
| | *displayQuestionTypes* | Displays the types of inquiries a user can submit to the system |
| | *selectQuestionType* | Prompts the user for the selection of one specific query type |
| | *displayInputFormat* | Shows the format in which the user has to provide their problem description. |
| | *setFormData* | Prompts the user for entering the data of its problem description. |
| | Comments | None at the moment. |

5.13 User Profile Management Service (UPMS)
5.9 Problem Description Generation Service (PDGS)

(4) Infer required data types

## 5.6 Information Production Service (IPS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Information Production Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>and likely some others. |
| Description | The Information Production Service receives as input the Content Plan compiled by the Content Selection Service, the user profile and potentially other contextual setting restrictions and produces user problem targeted multimodal information as text, table and graphic, which is passed to the Answer Service.<br>The Information Production Service provides its functionality through the following interfaces:<br>• *Generate Information*.<br>The Information Production Service interacts with the |

| | |
|---|---|
| | following services:<br>• User Profile Management Service<br>• Content Selection Service<br>• Answer Service<br>• User Interaction Service<br>External resources required by the Information Production service:<br>• Annotated corpora |
| Interface *GenerateInformation* | |
| *getInfo* | Takes as input a content plan, the user profile and contextual settings and provides information in one of the three modes (text, graphic, table) – or in a combination of thereof – in one of the languages of PESCaDO (Finnish, English or Swedish). |
| Comments | None |

## 5.7 Intersection Service (IS)

| | |
|---|---|
| Standard Specifications | No corresponding standard is known. |
| Description | The Intersection Service segments the route of interest into different parts (i.e. areas or smaller routes) based on ground and road condition parameters (e.g. normal traffic, width of roads, etc.). It should be noted that the parameters for segmenting a route are subject to the requirements for fusion. It takes as input:<br>• The route, which is provided by the Route Calculation Service.<br>The service processes this information in order to output non-overlapping geographical areas for the aspects of interest. In every segmented area, different measurements may be reported for the same aspect. These values are fused at a later stage by the Fusion Service.<br>The Intersection Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *SegmentationService*: This interface allows a client to make requests and receive areas of interest. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of a Data Node Repository Service instance. These include the functionalities of the service as well as the input and output specifications. |
| Interface *SegmentationService* | |
| *getSegmentedAreas* | This functionality returns the coordinates of each segmented area. |

| | | |
|---|---|---|
| | *getAspectsValues* | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| | Example usage | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| | Comments | It is beyond the scope of this service to provide any fusion of the obtained results (i.e. measurements of aspects). For each intersection area, it is possible that we have multiple measurements for the same aspect when the information is retrieved by different nodes.<br>This service can either provide direct input to the Fusion Service or return the result to the Data Retrieval Service.<br>*At the current stage of the project the exact functionality of the IS is not exactly clear as it is highly based on the requirements for fusion. Also the interaction with the Fusion Service needs further research.* |
| | 5.8 Knowledge Base Access Service (KBAS)<br>5.10 Related Aspects Computation Service (RACS)<br>5.11 Route Calculation Service (RCS) | |
| (5) Lookup Nodes | **¡Error! No se encuentra el origen de la referencia.** | |
| (6) Retrieve Data | 5.6 Information Production Service (IPS) | |
| | Standard Specifications | PESCaDO's Information Production Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>and likely some others. |
| | Description | The Information Production Service receives as input the Content Plan compiled by the Content Selection Service, the user profile and potentially other contextual setting restrictions and produces user problem targeted multimodal information as text, table and graphic, which is passed to the Answer Service.<br>The Information Production Service provides its functionality through the following interfaces:<br>• *Generate Information*.<br>The Information Production Service interacts with the following services:<br>• User Profile Management Service |

| | |
|---|---|
| | • Content Selection Service<br>• Answer Service<br>• User Interaction Service<br>External resources required by the Information Production service:<br>• Annotated corpora |
| Interface *GenerateInformation* | |
| *getInfo* | Takes as input a content plan, the user profile and contextual settings and provides information in one of the three modes (text, graphic, table) – or in a combination of thereof – in one of the languages of PESCaDO (Finnish, English or Swedish). |
| Comments | None |

5.7 Intersection Service (IS)

## 5.2 Content Selection Service (CSS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Content Selection Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF |
| Description | The Content Selection Service selects the content elements that are to be communicated to the user from the content set provided by the Decision Service in accordance with the profile of the user and other relevance criteria. The expert user is involved in the process of content selection.<br>The Content Selection Service provides the functionality through the following interfaces:<br>• *RetrieveContentPlan*<br>The Content Selection Service interacts with the following services:<br>• Decision Service<br>• User Profile Management Service<br>• Answer Service<br>• User Interaction Service |
| Interface *RetrieveContentPlan* | |

| | | | |
|---|---|---|---|
| | | *getContentPlan* | Provides to the Answer Service the content, i.e., a set of content elements interlinked by content relations that are considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The content provided by the Decision Service.<br>• The contextual settings, including the profile of the user.<br>The output of the function is:<br>• A content plan, i.e., a set of content elements interlinked by content relations, to be communicated to the user. |
| | | Comments | None at the moment. |
| | 5.3 Data Retrieval Service (DRS) | | |
| (7) Fuse data based on uncertainty metrics | 5.6 Information Production Service (IPS) | | |
| | | Standard Specifications | PESCaDO's Information Production Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>and likely some others. |
| | | Description | The Information Production Service receives as input the Content Plan compiled by the Content Selection Service, the user profile and potentially other contextual setting restrictions and produces user problem targeted multimodal information as text, table and graphic, which is passed to the Answer Service.<br>The Information Production Service provides its functionality through the following interfaces:<br>• *Generate Information*.<br>The Information Production Service interacts with the following services:<br>• User Profile Management Service<br>• Content Selection Service<br>• Answer Service<br>• User Interaction Service<br>External resources required by the Information Production |

| | |
|---|---|
| | service:<br>• Annotated corpora |
| Interface *GenerateInformation* | |
| *getInfo* | Takes as input a content plan, the user profile and contextual settings and provides information in one of the three modes (text, graphic, table) – or in a combination of thereof – in one of the languages of PESCaDO (Finnish, English or Swedish). |
| Comments | None |

## 5.7 Intersection Service (IS)

| | |
|---|---|
| Standard Specifications | No corresponding standard is known. |
| Description | The Intersection Service segments the route of interest into different parts (i.e. areas or smaller routes) based on ground and road condition parameters (e.g. normal traffic, width of roads, etc.). It should be noted that the parameters for segmenting a route are subject to the requirements for fusion. It takes as input:<br>• The route, which is provided by the Route Calculation Service.<br>The service processes this information in order to output non-overlapping geographical areas for the aspects of interest. In every segmented area, different measurements may be reported for the same aspect. These values are fused at a later stage by the Fusion Service.<br>The Intersection Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *SegmentationService*: This interface allows a client to make requests and receive areas of interest. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of a Data Node Repository Service instance. These include the functionalities of the service as well as the input and output specifications. |
| Interface *SegmentationService* | |
| *getSegmentedAreas* | This functionality returns the coordinates of each segmented area. |
| *getAspectsValues* | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Example usage | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Comments | It is beyond the scope of this service to provide any fusion of the obtained results (i.e. measurements of aspects). For each intersection area, it is possible that we have multiple |

| | |
|---|---|
| | measurements for the same aspect when the information is retrieved by different nodes.<br>This service can either provide direct input to the Fusion Service or return the result to the Data Retrieval Service.<br>*At the current stage of the project the exact functionality of the IS is not exactly clear as it is highly based on the requirements for fusion. Also the interaction with the Fusion Service needs further research.* |

5.8 Knowledge Base Access Service (KBAS)

## 5.2 Content Selection Service (CSS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Content Selection Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF |
| Description | The Content Selection Service selects the content elements that are to be communicated to the user from the content set provided by the Decision Service in accordance with the profile of the user and other relevance criteria. The expert user is involved in the process of content selection.<br>The Content Selection Service provides the functionality through the following interfaces:<br>• *RetrieveContentPlan*<br>The Content Selection Service interacts with the following services:<br>• Decision Service<br>• User Profile Management Service<br>• Answer Service<br>• User Interaction Service |
| Interface *RetrieveContentPlan* | |

| | |
|---|---|
| *getContentPlan* | Provides to the Answer Service the content, i.e., a set of content elements interlinked by content relations that are considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The content provided by the Decision Service.<br>• The contextual settings, including the profile of the user.<br>The output of the function is:<br>• A content plan, i.e., a set of content elements interlinked by content relations, to be communicated to the user. |
| Comments | None at the moment. |

5.3 Data Retrieval Service (DRS)

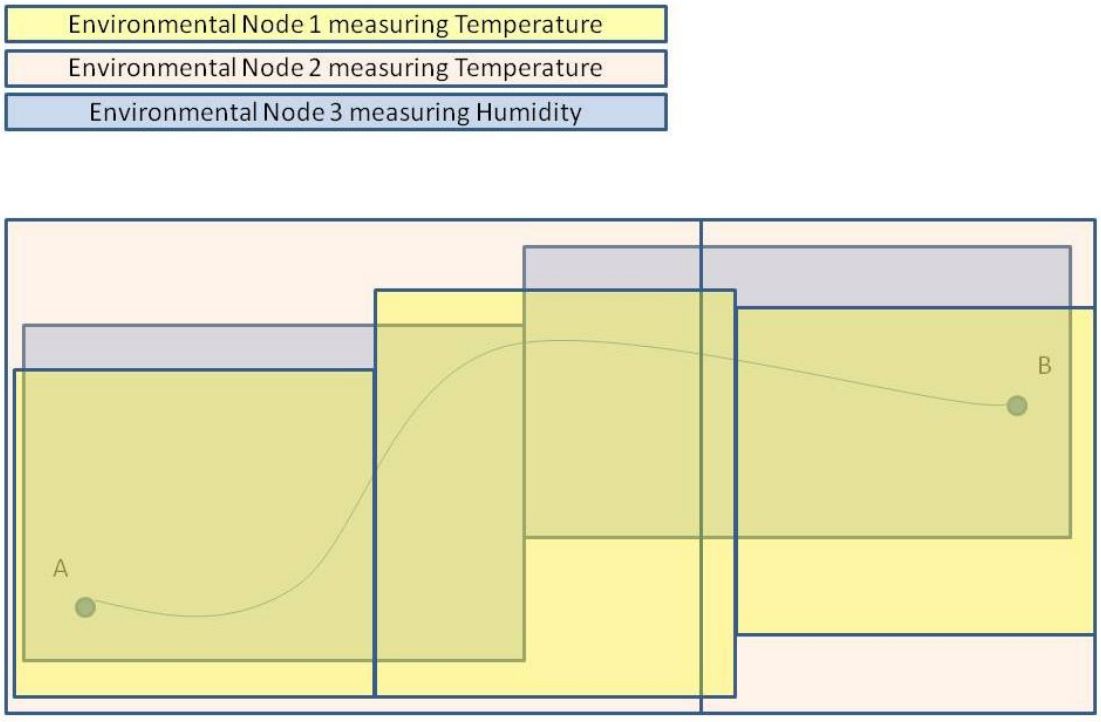


**Figure 11: Travelling from A to B and the corresponding nodes represented by their coverage area, while the colour denotes the type of the service.**

**Figure 12: Node access of the DRS**

## 5.4 Decision Service (DS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Decision Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>• PESCaDO-PDL (to become standard in PESCaDO)<br>and likely some others. |
| Description | The Decision Service retrieves the content that is relevant to the solution of the user's problem formulated in terms of PESCaDO-PDL statements. It thus<br>(xiii) accesses the KB in order to inspect the available knowledge elements and to determine their relevance to the problem of the user;<br>(xiv) makes inferences to deduce knowledge elements not explicitly available in the KB;<br>(xv) makes inferences to deduce the necessity to solicit missing data from the Data Retrieval Service;<br>(xvi) solicits data from the Data Retrieval Service.<br>The Decision Service provides its functionality through the following interfaces:<br>• *ObtainRelevantContent*. |

| | | |
|---|---|---|
| | | The Decision Service interacts with the following services:<br>• User Profile Management Service<br>• Knowledge Base Access Service<br>• Data Retrieval Service<br>• Answer Service |
| | Interface *ObtainRelevantContent* | |
| | *computeRelevantContent* | Provides to the Answer Service the content that is considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query which references also the contextual settings, including the profile of the user.<br>• The Knowledge Base.<br>The output of the function is:<br>• None (the decision service will write the inferred facts directly in the Knowledge Base containing the data for the current query – everything will be reachable from the instance representing the problem). |
| | Comments | None at the moment. |

5.5 Fusion Service (FS)

(8) Infer Result

## 5.6 Information Production Service (IPS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Information Production Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>and likely some others. |
| Description | The Information Production Service receives as input the Content Plan compiled by the Content Selection Service, the user profile and potentially other contextual setting restrictions and produces user problem targeted multimodal information as text, table and graphic, which is passed to the Answer Service.<br>The Information Production Service provides its functionality through the following interfaces:<br>• *Generate Information*. |

| | |
|---|---|
| | The Information Production Service interacts with the following services:<br>• User Profile Management Service<br>• Content Selection Service<br>• Answer Service<br>• User Interaction Service<br>External resources required by the Information Production service:<br>• Annotated corpora |
| Interface *GenerateInformation* | |
| *getInfo* | Takes as input a content plan, the user profile and contextual settings and provides information in one of the three modes (text, graphic, table) – or in a combination of thereof – in one of the languages of PESCaDO (Finnish, English or Swedish). |
| Comments | None |

## 5.7 Intersection Service (IS)

| | |
|---|---|
| Standard Specifications | No corresponding standard is known. |
| Description | The Intersection Service segments the route of interest into different parts (i.e. areas or smaller routes) based on ground and road condition parameters (e.g. normal traffic, width of roads, etc.). It should be noted that the parameters for segmenting a route are subject to the requirements for fusion. It takes as input:<br>• The route, which is provided by the Route Calculation Service.<br>The service processes this information in order to output non-overlapping geographical areas for the aspects of interest. In every segmented area, different measurements may be reported for the same aspect. These values are fused at a later stage by the Fusion Service.<br>The Intersection Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *SegmentationService*: This interface allows a client to make requests and receive areas of interest. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of a Data Node Repository Service instance. These include the functionalities of the service as well as the input and output specifications. |
| Interface *SegmentationService* | |
| *getSegmentedAreas* | This functionality returns the coordinates of each segmented area. |

| | |
|---|---|
| *getAspectsValues* | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Example usage | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Comments | It is beyond the scope of this service to provide any fusion of the obtained results (i.e. measurements of aspects). For each intersection area, it is possible that we have multiple measurements for the same aspect when the information is retrieved by different nodes.<br>This service can either provide direct input to the Fusion Service or return the result to the Data Retrieval Service.<br>*At the current stage of the project the exact functionality of the IS is not exactly clear as it is highly based on the requirements for fusion. Also the interaction with the Fusion Service needs further research.* |

5.8 Knowledge Base Access Service (KBAS)

## 5.2 Content Selection Service (CSS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Content Selection Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF |
| Description | The Content Selection Service selects the content elements that are to be communicated to the user from the content set provided by the Decision Service in accordance with the profile of the user and other relevance criteria. The expert user is involved in the process of content selection.<br>The Content Selection Service provides the functionality through the following interfaces:<br>• *RetrieveContentPlan*<br>The Content Selection Service interacts with the following services:<br>• Decision Service<br>• User Profile Management Service<br>• Answer Service<br>• User Interaction Service |
| Interface *RetrieveContentPlan* | |

| | |
|---|---|
| *getContentPlan* | Provides to the Answer Service the content, i.e., a set of content elements interlinked by content relations that are considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The content provided by the Decision Service.<br>• The contextual settings, including the profile of the user.<br>The output of the function is:<br>• A content plan, i.e., a set of content elements interlinked by content relations, to be communicated to the user. |
| Comments | None at the moment. |

5.3 Data Retrieval Service (DRS)

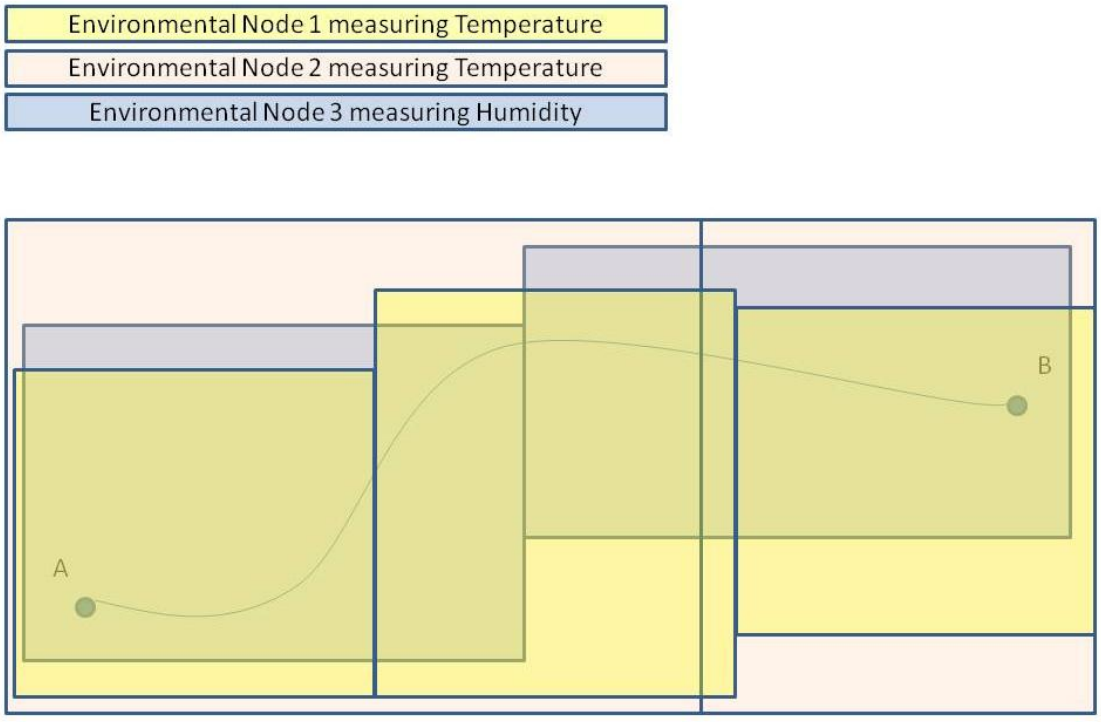


**Figure 11: Travelling from A to B and the corresponding nodes represented by their coverage area, while the colour denotes the type of the service.**

**Figure 12: Node access of the DRS**

5.4 Decision Service (DS)
5.2 Content Selection Service (CSS)

(9) Deliver Information

5.6 Information Production Service (IPS)
5.12 User Interaction Service (UIS)

## 5.2 Content Selection Service (CSS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Content Selection Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF |
| Description | The Content Selection Service selects the content elements that are to be communicated to the user from the content set provided by the Decision Service in accordance with the profile of the user and other relevance criteria. The expert user is involved in the process of content selection.<br>The Content Selection Service provides the functionality through the following interfaces:<br>• *RetrieveContentPlan*<br>The Content Selection Service interacts with the following services:<br>• Decision Service<br>• User Profile Management Service<br>• Answer Service |

| | |
|---|---|
| | • User Interaction Service |
| Interface *RetrieveContentPlan* | |
| *getContentPlan* | Provides to the Answer Service the content, i.e., a set of content elements interlinked by content relations that are considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The content provided by the Decision Service.<br>• The contextual settings, including the profile of the user.<br>The output of the function is:<br>• A content plan, i.e., a set of content elements interlinked by content relations, to be communicated to the user. |
| Comments | None at the moment. |

## 5.3 Data Retrieval Service (DRS)

*Note: The* Data Node Repository Service *(DNRS) was removed due to the fact that its functionality was moved to the* Data Retrieval Service *(DRS). The relevant operations are* getCapabilities *and* describeSensor*.*

| | |
|---|---|
| Standard Specifications | The Data Retrieval Service uses the following standards for communication purposes with other services and modules:<br>• The XML (Extensible Markup Language), which is a subset of SGML (Standard Generalized Markup Language). SGML is a widely used international text processing standard that is standardised with ISO 8879:1986(E). XML's goal is to enable generic SGML to be served, received, and processed on the Web. (http://www.w3.org/TR/2008/REC-xml-20081126/)<br>• SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment and it is based on XML. SOAP can potentially be used in combination with a variety of other protocols such as HTTP. (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)<br>• The HTTP (Hypertext Transfer Protocol) is an application-level protocol. HTTP allows an open-ended set of methods to be used to indicate the purpose of a request. It builds upon the notion of reference provided by the Uniform Resource Identifier (URI) as a location (URL) for indicating the resource on which a method is to be applied. |
| Description | The Data Retrieval Service is responsible for the retrieval of the requested information (i.e. measurements of specific aspects) for the geographical area of interest. The DRS, as |

| | | | |
|---|---|---|---|
| | | | shown in Figure 12, plays also the role of managing the available environmental data nodes and the Intersection Service in the sense of processing data requests. If the results that are obtained from the Web do not include updated measurements for the involved aspects, the service connects directly to the nodes and downloads the latest measurements.<br>DRS takes as input:<br>• Area of interest, which is described either as a single route (i.e. trajectory of movement) or as a geographical area. It is defined by a set of points described by their geographical coordinates. This information is received from the Route Calculation Service.<br>• List of types of relevant aspects (e.g. air quality, temperature, pollen). This information is retrieved by the Related Aspects Computation Service.<br>The output of the service is:<br>• Measurements of several aspects in different (probably overlapping) areas with their specific time and space values.<br>The Data Node Repository Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *DataAccess*: Enables the client to submit requests and retrieve information. |
| | | Interface *ServiceCapabilities* | |
| | | *getCapabilities* | Informs the client about the capabilities of a Data Retrieval Service instance. The capabilities include the functionalities of the service as well as the type of data that can be retrieved, which is a list of the available sensors, covered areas, time ranges and quality information. Note: The term sensor is meant in a wider sense, e.g. an "environmental data node" in the PESCaDO terms. |
| | | *describeSensor* | Get the metadata description of a sensor (the term sensor is meant in a wider sense, e.g. an "environmental data node" in the PESCaDO terms). The metadata include information like the covered area, time range for which data is available, quality and reliability of the data, keywords that describe the service according to the node provider, etc. |
| | | Interface *DataAccess* | |
| | | *getData* | Get data from sensors. Data can be filtered by aspect, attribute, time and location (area). |
| | | Example usage | In the reference use scenario, the user has the intention of travelling from A to B (Figure 3). The Route Calculation Service provides the travelling trajectory as a set of geographical coordinates with time durations; the Related Aspects Computation Service outputs temperature and humidity as the relevant aspects; the Intersection service segments the route into smaller parts. Then, the Data |

| | |
|---|---|
| | Retrieval Service is called recursively for each part of the route to retrieve the relevant geographical areas, the corresponding measurements (i.e. temperature and humidity values retrieved from the environmental nodes 1, 2, 3 in Figure 11) and the quality and reliability metrics for each node. If a value is outdated (i.e. older than a specified time threshold), e.g. the humidity value of the environmental node 3. In that case, the DRS will connect directly to the corresponding environmental node and retrieve an updated value for the humidity. Then, the Data Retrieval Service delivers three different areas with different aspect measurements (Figure 11). In this case the coverage areas are considered rectangular for convenience; however they might be updated to cyclic areas in case that such a representation is more realistic for the Fusion Service. To be noted that in the example of Figure 11 the route was not segmented to smaller parts by the intersection service. |
| Comments | It is beyond the scope of this service to provide any fusion of the obtained results. |

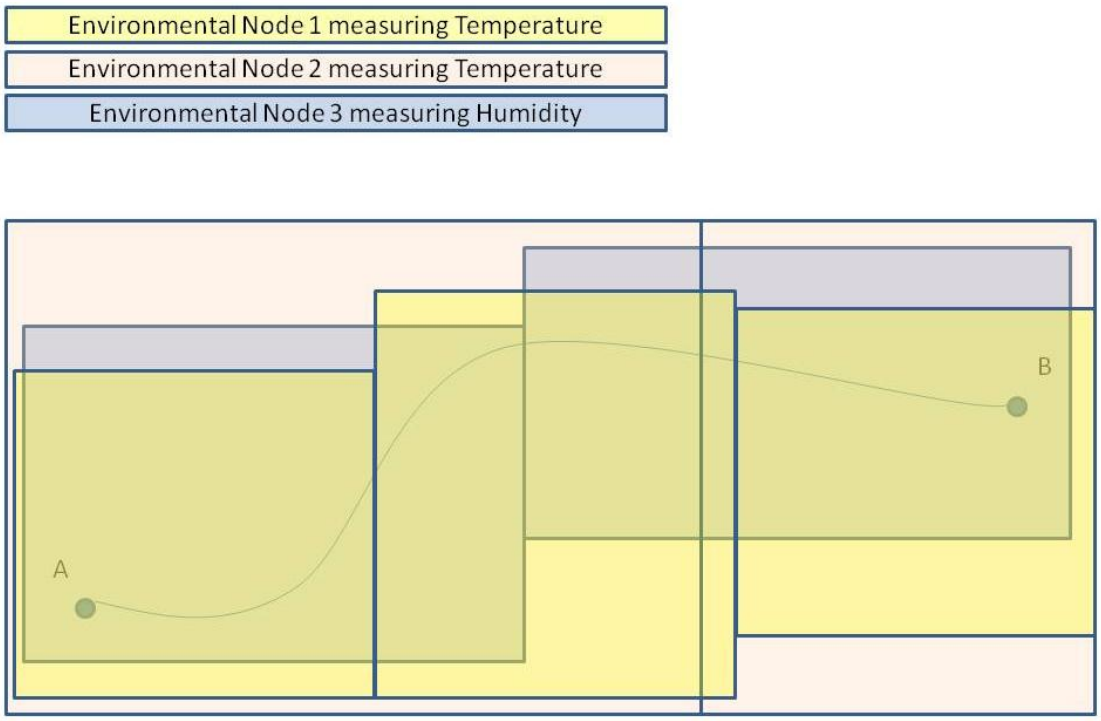


**Figure 11: Travelling from A to B and the corresponding nodes represented by their coverage area, while the colour denotes the type of the service.**

**Figure 12: Node access of the DRS**

## 5.4 Decision Service (DS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Decision Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>• PESCaDO-PDL (to become standard in PESCaDO)<br>and likely some others. |
| Description | The Decision Service retrieves the content that is relevant to the solution of the user's problem formulated in terms of PESCaDO-PDL statements. It thus<br>(xvii) accesses the KB in order to inspect the available knowledge elements and to determine their relevance to the problem of the user;<br>(xviii) makes inferences to deduce knowledge elements not explicitly available in the KB;<br>(xix) makes inferences to deduce the necessity to solicit missing data from the Data Retrieval Service;<br>(xx) solicits data from the Data Retrieval Service.<br>The Decision Service provides its functionality through the following interfaces:<br>• *ObtainRelevantContent*. |

| | |
|---|---|
| | The Decision Service interacts with the following services:<br>• User Profile Management Service<br>• Knowledge Base Access Service<br>• Data Retrieval Service<br>• Answer Service |
| Interface *ObtainRelevantContent* | |
| *computeRelevantContent* | Provides to the Answer Service the content that is considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query which references also the contextual settings, including the profile of the user.<br>• The Knowledge Base.<br>The output of the function is:<br>• None (the decision service will write the inferred facts directly in the Knowledge Base containing the data for the current query – everything will be reachable from the instance representing the problem). |
| Comments | None at the moment. |

## 5.5 Fusion Service (FS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Fusion Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF |
| Description | The general task of the FS is to process the data received from different environmental service nodes via the Data Retrieval Service with the purpose to obtain the best and most complete data set to address the user's request for information or decision support. It thus<br>(xxi) identifies complementary pieces of data proceeding from different data sources of the same kind (i.e., AQ, meteorology, traffic, etc.) and fuses them into one coherent data block;<br>(xxii) identifies alternative (or competing) pieces of data proceeding from different data sources, but concerning the same environmental phenomenon for |

| | |
|---|---|
| | the same location, assesses the quality of each of them using the two types of uncertainty metrics developed in PESCaDO (imprecision and confidence metrics), and selects the best for communication to the user.<br>The Fusion Service provides the functionality through the following interfaces:<br>• *FuseData*: Delivers the fused dataset.<br>The Fusion Service interacts with the following services:<br>• Data Retrieval Service<br>• Knowledge Base Access Service<br>• User Profile Management Service<br>• User Interaction Service |
| Interface *FuseData* | |
| *getFusedData* | Takes as input:<br>• The data from the data retrieval service and pointers to the corresponding service node fingerprints.<br>• Potential preferences of the user for specific service nodes.<br>The output of the function is:<br>• A fused data set. |
| Comments | None at the moment. |

## 5.6 Information Production Service (IPS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Information Production Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>and likely some others. |
| Description | The Information Production Service receives as input the Content Plan compiled by the Content Selection Service, the user profile and potentially other contextual setting restrictions and produces user problem targeted multimodal information as text, table and graphic, which is passed to the Answer Service.<br>The Information Production Service provides its functionality through the following interfaces:<br>• *Generate Information*. |

| | |
|---|---|
| | The Information Production Service interacts with the following services:<br>• User Profile Management Service<br>• Content Selection Service<br>• Answer Service<br>• User Interaction Service<br>External resources required by the Information Production service:<br>• Annotated corpora |
| Interface *GenerateInformation* | |
| *getInfo* | Takes as input a content plan, the user profile and contextual settings and provides information in one of the three modes (text, graphic, table) – or in a combination of thereof – in one of the languages of PESCaDO (Finnish, English or Swedish). |
| Comments | None |

## 5.7 Intersection Service (IS)

| | |
|---|---|
| Standard Specifications | No corresponding standard is known. |
| Description | The Intersection Service segments the route of interest into different parts (i.e. areas or smaller routes) based on ground and road condition parameters (e.g. normal traffic, width of roads, etc.). It should be noted that the parameters for segmenting a route are subject to the requirements for fusion. It takes as input:<br>• The route, which is provided by the Route Calculation Service.<br>The service processes this information in order to output non-overlapping geographical areas for the aspects of interest. In every segmented area, different measurements may be reported for the same aspect. These values are fused at a later stage by the Fusion Service.<br>The Intersection Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *SegmentationService*: This interface allows a client to make requests and receive areas of interest. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of a Data Node Repository Service instance. These include the functionalities of the service as well as the input and output specifications. |
| Interface *SegmentationService* | |
| *getSegmentedAreas* | This functionality returns the coordinates of each segmented area. |

| | |
|---|---|
| *getAspectsValues* | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Example usage | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Comments | It is beyond the scope of this service to provide any fusion of the obtained results (i.e. measurements of aspects). For each intersection area, it is possible that we have multiple measurements for the same aspect when the information is retrieved by different nodes.<br>This service can either provide direct input to the Fusion Service or return the result to the Data Retrieval Service.<br>*At the current stage of the project the exact functionality of the IS is not exactly clear as it is highly based on the requirements for fusion. Also the interaction with the Fusion Service needs further research.* |

## 5.8 Knowledge Base Access Service (KBAS)

| | |
|---|---|
| Standard Specifications | The Knowledge Base for the PESCaDO platform is currently planned to be based on the following ontology language:<br>• W3C OWL Web Ontology Language http://www.w3.org/TR/owl-features/<br>Relevant standard specification for this service are also:<br>• W3C RDFS http://www.w3.org/TR/rdf-schema/<br>• W3C RDF/XML Syntax Specification (Revised) http://www.w3.org/TR/rdf-syntax-grammar/<br>• W3C SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/<br>• W3C SWRL: A Semantic Web Rule Language Combining OWL and RuleML http://www.w3.org/Submission/SWRL/<br>• W3C URIs, URLs, and URNs: Clarifications and Recommendations 1.0 http://www.w3.org/TR/uri-clarification/ |
| Description | The Knowledge Base Access Service supports the access to the Knowledge Base on which the whole PESCaDO platform relies to provide user decision support. This service provides, via the *KBAccess* interface, access, manipulation and storage of ontologies (or ontology modules) stored in the Knowledge Base. Besides the common *KBAccess* interface retrieval methods, the *KBQuery* interface offers a generic mechanism for retrieval.<br><br>Some typical uses of this service are:<br>• Storing, updating or deleting available ontology modules;<br>• Retrieving (partially or fully) a stored ontology module; |

| | | | |
|---|---|---|---|
| | | | • Getting high-level information about some ontology module, such as the list of concepts, or the list of supported properties for a given concept;<br>• Inserting in the knowledge base actual data, e.g. those retrieved from the Data Repository, which will be used to fulfil the user problem request;<br>• Querying one of more ontology modules.<br><br>The Knowledge Base Access Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs the client about the common and specific capabilities of the Knowledge Base Access Service.<br>• *KBAccess*: Supports the management (storage, retrieval, and deletion) of ontology modules, provides a high-level view on ontologies, and allows for adding factual knowledge (data from the data repository, inferred data) to the knowledge base.<br>• *KBQuery*: Providing operations to query the ontology modules contained in the knowledge base. |
| | | Interface *ServiceCapabilities* | |
| | | *get Capabilities* | Informs the client about the common and specific capabilities of a Knowledge Base Access Service instance. Examples of specific capabilities are:<br>• The names of the ontology modules available in the Knowledge Base.<br>• The possible representation formats of query results which can be requested by clients.<br>• The query languages that can be used in knowledge base queries.<br>• The inference capabilities of the knowledge base applied when computing query results. |
| | | Interface *KBAccess* | |
| | | *setOntology* | Stores a new ontology module in the Knowledge Base. |
| | | *getOntology* | Retrieves an existing ontology module or a selection of an ontology module from the Knowledge Base. |
| | | *deleteOntology* | Removes an existing ontology module from the Knowledge Base. |
| | | *getTBox Vocabulary* | Given an ontology or a part (segment) of an ontology, returns a list of TBox statements ready to be used for creating a Knowledge Base. |
| | | *getABox Vocabulary* | Given an ontology or a part (segment) of an ontology, returns a list of ABox statements ready to be used for creating a Knowledge Base. |

| | |
|---|---|
| *addABoxStatement* | This function allows for inserting into the knowledge base the factual knowledge (data) that will be used to fulfil a user request (e.g. temperature values, pollutants concentrations). The data to be added may be, e.g. those returned by the Data Retrieval Service/Fusion Service, and the data inferred by the Decision Service. |
| *getListConcepts* | Returns a list of concepts of an ontology (module) or a part (segment) of an ontology (module). |
| Example usage | The *setOntology* operation may be used, for instance, when a new version of an ontology module has to be uploaded in the Knowledge Base. Other services of the PESCaDO platform (e.g. Fusion Service, Decision Service), may need to process all the information contained in an ontology modules to perform the operations needed to be able to invoke the *getOntology* operation. |
| Comments | Creating or editing of ontology modules is out of the scope of the Knowledge Base Access Service: this service's interface manages only the storage and the access to ontologies, but doesn't provide any tool to create ontology structures. |
| Interface *KBQuery* | |
| *queryOntology* | Submits a query to the ontologies stored in the knowledge base. The query has to be formulated in a query language (e.g. SPARQL) compatible with the knowledge representation model used by the knowledge base (e.g. RDF/OWL). |
| Example usage | The *queryOntology* operation may be invoked by other services like the Fusion Service, Decision Service or the Problem Description Generation Service to obtain the relevant information they need to take some decisions or perform some reasoning. For example, the Problem Description Generation Service may use this operation to retrieve from the corresponding ontology module in the Knowledge Base the relevant data for preparing a complete user problem description. |
| Comments | |

## 5.9 Problem Description Generation Service (PDGS)

| | |
|---|---|
| Standard Specifications | No specific corresponding standard known.<br>The Problem Description Language (PDL) is planned to be based on:<br>• W3C XML Extensible Markup Language http://www.w3.org/XML/ |
| Description | The goal of this service is to generate / check a description of the problem in PDL submitted by the user to the PESCaDO platform, according to the information provided by the user via the user interface / third party systems.<br>The Problem Description Generation Service provides the |

| | | | |
|---|---|---|---|
| | | | functionality through the following interfaces:<br>• ServiceCapabilities: Informs about the common and specific capabilities.<br>• *DescriptionGeneration*: generates a description in PDL of the problem submitted by the user to the PESCaDO platform, according to the information provided by the user via the user interface.<br>• *DescriptionChecking*: checks that a problem description expressed in PDL and submitted by third party systems is complete, i.e. no user information needed as part of the request is missing. |
| | | Interface *ServiceCapabilities* | |
| | | *getCapabilities* | Informs the client about the capabilities of a Problem Description Generation Service instance. Examples of specific capabilities are the type of problems that can be formulated, the required input parameters needed for a specific problem, and the kind of output a specific problem returns. |
| | | Interface *DescriptionGeneration* | |
| | | *generateProblemDescription* | This operation returns a structured description in PDL of the problem submitted by the user. Input parameter for this operation is the selected problem, together with a list of input data provided by the user via the user interface. The operation checks that all data needed to compile a complete problem description are available within the input parameter list. To perform this task, the operation needs to access (via the Knowledge Base Access Service) the ontology which describes the problem/data dependencies. If some required data has not been provided in the input, the operation can retrieve the missing data by accessing the User Profile via the User Profile Management Service.<br>Input parameters:<br>• The URI of the type of problem type selected by the user<br>• The URI of the type of activity selected by the user<br>• 0..n additional parameters related to the problem type<br>Returned value:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query. The PDGS will create also new instances and facts about the problem, and write the problem description directly in the Knowledge Base containing the data for the current query, but just the URI of the problem description will be returned to the AS. |
| | | Example usage | The typical application scenario for this service interface is that the *generateProblemDescription* operation is invoked by the Answer Service with the input data on the problem to be solved collected via the user interface. |
| | | Comments | None at the moment. |
| | | Interface *DescriptionChecking* | |

| | |
|---|---|
| *checkProblemDescription (optional)* | This operation checks whether a problem description expressed according to the PDL is complete (i.e. that there is no user input data missing). The input parameter for this operation is a description of a user problem in the PDL. The output of the operation is a yes/no answer. In the case of a negative answer, a description of the missing parameter can optionally also be returned. |
| Example usage | A typical application scenario for this service interface is that the *CheckProblemDescription* operation is invoked by the Answer Service in the case of user decision support requests not submitted via the PESCaDO platform user interface, like e.g. batch submission of third party organisation / system. |
| Comments | None at the moment. |

## 5.10 Related Aspects Computation Service (RACS)

| | |
|---|---|
| Standard Specifications | No specific corresponding standard known.<br>(Data format to query the knowledge base?) |
| Description | The Related Aspects Computation Service (RACS) is a service that, given the problem description generated for the request made by the user, selects the aspects that need to be considered for delivering an informative answer to the user.<br>The input of the service is planned to be an *action specification* (e.g. "travel from X to Y") and the *query focus* (or topic, e.g. "health issues") submitted by the user, as encoded in the PDL (Problem Description Language) format by the Problem Description Generation Service. The output is a set of focus-related aspects that have to be considered before an answer to the user query is given.<br>Given the action and focus of the user query, the service accesses the Knowledge Base containing general information about aspects relevant to an <action, focus> couple. The Knowledge Base returns a list of relevant aspects, with a possibly empty list of specific constraints that need to be verified to assess the relevance of an aspect for a specific user associated with each of the aspects in question. The RACS will filter the list of aspects returned by the Knowledge Base by matching the constraints associated with each aspect against the information contained in the user profile.<br>The RACS provides the functionality through the following interfaces:<br>• *RelevantAspects*: accesses the knowledge base retrieving all aspects relevant to the focus associated with the query.<br>• *InformationFiltering*: compares the focus-related aspects with the user profile and filters them in order to keep only the relevant ones. |
| Interface *RelevantAspects* | |
| *getRelevantAspects* | Given the action and focus of the user query, the operation |

| | |
|---|---|
| | returns a set of related aspects. Each aspect can have an associated list of relevance constraints to be matched against information contained in the user profile.<br><br>Input parameters:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query;<br><br>Returns a list of couples (0..n):<br>• The URI of the entity in the KB corresponding to a specific aspect (e.g. http://www.pescado-project.eu/ontology/pescado.owl#precipitation or http://www.pescado-project.eu/ontology/pescado.owl#O3)<br>• The URI of the class in the KB representing the type of the aspect (e.g. http://www.pescado-project.eu/ontology/pescado.owl#AirQualityType or http://www.pescado-project.eu/ontology/pescado.owl#WeatherType) |
| Interface *InformationFiltering* | |
| *filterClasses* | The aspects retrieved from the *RelevantAspects* interface are compared against the user profile information and only those that are relevant to the user are returned. |
| Example usage | A requestor wants to know if it is safe for him to travel from place X to destination Y. The user profile of the requestor says that he is a middle-age man with allergy to pollen and no other disease. RACS is asked to find aspects which are relevant to the requestor given the action “travel from X to Y” and the focus “health issues”. RACS accesses the Knowledge Base and finds that generally speaking “pollen” and “heat” are relevant aspects for a person that wants to travel and is concerned about health. The Knowledge Base also provides the information that “pollen” aspects are relevant only to people who have pollen allergies, and the “heat” aspect is relevant only if the user is old or has heart diseases. The RACS accesses the user profile and finds that the “allergy” constraint is matched (the user is allergic to pollen), whereas the “aged or heart disease” condition is not matched. The heat factor is not dangerous and can be filtered out. RACS outputs the pollen aspect as the only one which is relevant to the requestor and that needs to be further analyzed. |
| Comments | None at the moment. |

## 5.11 Route Calculation Service (RCS)

| | |
|---|---|
| Standard Specifications | Relevant standard specification for this service are:<br>• OpenGIS Geography Markup Language (GML) Encoding Standard<br>• Open Street Map http://www.openstreetmap.org/ |

| | |
|---|---|
| Description | The Route Calculation Service provides a range of services based on (freely) available spatial data, e.g. data provided by Open Street Map.<br>The Route Calculation Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities:* Informs about the common and specific capabilities.<br>• *LocationUtility*: Provides a Geocoder/Reverse Geocoder.<br>• *RouteCalculation:* Determines travel routes and navigation information. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of the Route Calculation Service, e.g. the supported output formats of calculated routes (e.g. GML/XML, GPX). |
| Interface *LocationUtility* | |
| *geoEncode* | The geoEncode operation transforms a description of a location, such as a place name, street address or postal code, into a normalized description of the location as point geometry. |
| *geoDecode* | The geoDecode operation transforms a normalized description of the location into a description of a location. |
| Interface *RouteCalculation* | |
| *calculateRoute* | The calculateRoute operation receives as input a start point and an end point. Moreover, diverse criteria such as the travelling means can be entered, e.g.<br>• car: fastest<br>• car: shortest<br>• bicycle<br>• walking<br>The operation then calculates automatically the route and returns it in the supported output format for further processing by other services or for rendering on a map by means of a user interface client. |
| Example usage | In a typical scenario, a user submits a request to the *Answer Service* that he/she wants to travel from point A to point B. To check the areas that the user will cross during his journey for potential health related aspects (e.g. pollen in case of an allergy, dangerous roads in winter time, etc.), the route must be known first. |
| Comments | • Further parameters which could be supplied to the calculateRoute operation (e.g. transit-stops, avoidance of motorways, avoidance of areas, etc.) need further discussion. Optionally, there should be a helper function for auto completion of road and city names.<br>• Ongoing research in PESCaDO might show that it would be better or sufficient not to have a route calculation but instead have a list of affected areas. To facilitate this option, e.g. the knowledge base must contain an area |

| | |
|---|---|
| | map (of Finland). |

## 5.12 User Interaction Service (UIS)

| | |
|---|---|
| Standard Specifications | PESCaDO's User Interaction Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• HTTP (Hypertext Transfer Protocol)<br>and likely some others. |
| Description | The User Interaction Service has two main functions. First, it ensures the communication between the expert and end users on the one side and the services that involve the intervention of the user for their optimal functioning on the other side. For the first version of the architecture of PESCaDO, these are: (i) the Fusion Service; (ii) the Content Selection Service, and the (iii) the Information Production Service. In the subsequent versions, the Data Node Retrieval Service will also make use of the UIS.<br>Second, it supports the communication with the end user and the Answer Service.<br>The User Interaction Service provides its functionality through the following interfaces:<br>• *FusionAndMetricsSupport*<br>• *ContentSelectionSupport*<br>• *DiscoursePlanningSupport*<br>• *AnswerServiceSupport*<br>The User Interaction Service interacts with the following services:<br>• Fusion Service<br>• Content selection Service<br>• Information Production Service<br>• Answer Service |
| Interface *FusionAndMetricsSupport* | |
| *displayMetrics* | Visualizes the metrics to the user. |
| *displayFusionSchema* | Visualizes to the user the picture of how data from specified data sources are fused to obtain the optimal value. |
| *modifyMetrics* | Allows the user to modify (in a graphical mode) the uncertainty / confidence metrics. |
| *modifyFusionSchema* | Allows the user to modify (in a graphical mode) the data fusion schema. |
| Interface *ContentSelectionSupport* | |

| | | |
|---|---|---|
| | *displayContent* | Displays a content element or the whole content plan to the user. |
| | *getRelevance* | Requests the judgement of the user concerning the relevance of a content element (can be positive or negative). |
| | Interface *DiscoursePlanningSupport* | |
| | *displayDiscourse* | Displays a discourse structure or a discourse element to the user. |
| | *getDiscourseRelation* | Requests the assignment of a discourse relation to a pair of content elements (the content elements can be complex). |
| | *orderDiscourse* | Requests the (linear) ordering of several discourse elements among each other. |
| | Comments | The offered functionality of the Interface *FusionAndMetricsSupport* is to be defined. |
| | Interface *AnswerServiceSupport* | |
| | *displayQuestionTypes* | Displays the types of inquiries a user can submit to the system |
| | *selectQuestionType* | Prompts the user for the selection of one specific query type |
| | *displayInputFormat* | Shows the format in which the user has to provide their problem description. |
| | *setFormData* | Prompts the user for entering the data of its problem description. |
| | Comments | None at the moment. |
| | 5.13 User Profile Management Service (UPMS) | |

The following subsections describe each of the identified services. The template for the description of a service is taken from [ORCHESTRA, 2008], Section 9.6.

## 5.1 Answer Service (AS)

| | |
|---|---|
| Standard Specifications | *No corresponding standard known at the moment.* However the concrete implementation may use a standard for service orchestrations (e.g. BPEL) or rule engines. |
| Description | This service initiates an answer to a request from the field of the PESCaDO topics. To reach this goal, it orchestrates other PESCaDO services in accordance with the established workflow. Each type of request may have its specific orchestration but the main algorithm is always the same (each step lists the main involved services):<br>1. *Generate a Pattern Description for the request*<br>• User Profile Management Service<br>• Problem Description Generation Service<br>2. *Get auxiliary information*<br>• For instance, AS looks at the generated PDL and notices that it contains a travel plan with a starting point and destination. Therefore, it concludes that a route needs to be calculated.<br>• Route Calculation Service (or similar |

| | |
|---|---|
| | services computing auxiliary information)<br>3. *Infer the required data types*<br>• Knowledge Base Access Service<br>• Related Aspects Computation Service<br>4. Lookup required data nodes<br>• Data Retrieval Service<br>5. Retrieve the data<br>• Intersection Service<br>• Data Retrieval Service<br>6. Fuse data based on uncertainty metrics<br>• Fusion Service<br>• Knowledge Base Access Service<br>• Data Retrieval Service<br>7. Infer Result and generate formal output document<br>• Knowledge Base Access Service<br>• Data Retrieval Service<br>• Decision Service<br>• Content Selection Service<br><br>The Answer Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities: Informs about the common and specific capabilities.*<br>• *Query*: Answer PESCaDO questions based on user input |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of the Answer Service. This is the information about the problem types that can be formulated, the required input parameters needed for a specific problem, and the kind of output a specific problem returns.<br><br>Return values for each problem type:<br>• A textual title of the problem type, language specific<br>• A textual description of the problem type, language specific<br>• The required input parameters needed for the specific problem<br>• The kind of output a specific problem returns |
| Interface *Query* | |
| *computeAnswer* | Computes the answer to a PESCaDO question.<br>The operation returns the answer as a multimodal document.<br><br>Input parameters are:<br>• The URI of the type of problem (reference to the KB) for which to compute an answer.<br>• Required parameters for the problem type.<br>• The parameter property, reference to the KB. |
| Example usage | See Figure 3: A possible question and the system's reaction in Use Case 1.<br>The Answer Service (used in step 2) orchestrates the |

| | |
|---|---|
| | services involved in step 3 to 8. |
| Comments | None at the moment. |

## 5.2 Content Selection Service (CSS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Content Selection Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF |
| Description | The Content Selection Service selects the content elements that are to be communicated to the user from the content set provided by the Decision Service in accordance with the profile of the user and other relevance criteria. The expert user is involved in the process of content selection.<br>The Content Selection Service provides the functionality through the following interfaces:<br>• *RetrieveContentPlan*<br>The Content Selection Service interacts with the following services:<br>• Decision Service<br>• User Profile Management Service<br>• Answer Service<br>• User Interaction Service |
| Interface *RetrieveContentPlan* | |
| *getContentPlan* | Provides to the Answer Service the content, i.e., a set of content elements interlinked by content relations that are considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The content provided by the Decision Service.<br>• The contextual settings, including the profile of the user.<br>The output of the function is:<br>• A content plan, i.e., a set of content elements interlinked by content relations, to be communicated to the user. |
| Comments | None at the moment. |

## 5.3 Data Retrieval Service (DRS)

*Note: The* Data Node Repository Service *(DNRS) was removed due to the fact that its functionality was moved to the* Data Retrieval Service *(DRS). The relevant operations are* getCapabilities *and* describeSensor.

| | |
|---|---|
| Standard Specifications | The Data Retrieval Service uses the following standards for communication purposes with other services and modules:<br>• The XML (Extensible Markup Language), which is a subset of SGML (Standard Generalized Markup Language). SGML is a widely used international text processing standard that is standardised with ISO 8879:1986(E). XML's goal is to enable generic SGML to be served, received, and processed on the Web. (http://www.w3.org/TR/2008/REC-xml-20081126/)<br>• SOAP is a lightweight protocol for exchange of information in a decentralized, distributed environment and it is based on XML. SOAP can potentially be used in combination with a variety of other protocols such as HTTP. (http://www.w3.org/TR/2000/NOTE-SOAP-20000508/)<br>• The HTTP (Hypertext Transfer Protocol) is an application-level protocol. HTTP allows an open-ended set of methods to be used to indicate the purpose of a request. It builds upon the notion of reference provided by the Uniform Resource Identifier (URI) as a location (URL) for indicating the resource on which a method is to be applied. |
| Description | The Data Retrieval Service is responsible for the retrieval of the requested information (i.e. measurements of specific aspects) for the geographical area of interest. The DRS, as shown in Figure 12, plays also the role of managing the available environmental data nodes and the Intersection Service in the sense of processing data requests. If the results that are obtained from the Web do not include updated measurements for the involved aspects, the service connects directly to the nodes and downloads the latest measurements.<br>DRS takes as input:<br>• Area of interest, which is described either as a single route (i.e. trajectory of movement) or as a geographical area. It is defined by a set of points described by their geographical coordinates. This information is received from the Route Calculation Service.<br>• List of types of relevant aspects (e.g. air quality, temperature, pollen). This information is retrieved by the Related Aspects Computation Service.<br>The output of the service is:<br>• Measurements of several aspects in different (probably overlapping) areas with their specific time and space values.<br>The Data Node Repository Service provides the functionality |

| | |
|---|---|
| | through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *DataAccess*: Enables the client to submit requests and retrieve information. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of a Data Retrieval Service instance. The capabilities include the functionalities of the service as well as the type of data that can be retrieved, which is a list of the available sensors, covered areas, time ranges and quality information. Note: The term sensor is meant in a wider sense, e.g. an "environmental data node" in the PESCaDO terms. |
| *describeSensor* | Get the metadata description of a sensor (the term sensor is meant in a wider sense, e.g. an "environmental data node" in the PESCaDO terms). The metadata include information like the covered area, time range for which data is available, quality and reliability of the data, keywords that describe the service according to the node provider, etc. |
| Interface *DataAccess* | |
| *getData* | Get data from sensors. Data can be filtered by aspect, attribute, time and location (area). |
| Example usage | In the reference use scenario, the user has the intention of travelling from A to B (Figure 3). The Route Calculation Service provides the travelling trajectory as a set of geographical coordinates with time durations; the Related Aspects Computation Service outputs temperature and humidity as the relevant aspects; the Intersection service segments the route into smaller parts. Then, the Data Retrieval Service is called recursively for each part of the route to retrieve the relevant geographical areas, the corresponding measurements (i.e. temperature and humidity values retrieved from the environmental nodes 1, 2, 3 in Figure 11) and the quality and reliability metrics for each node. If a value is outdated (i.e. older than a specified time threshold), e.g. the humidity value of the environmental node 3. In that case, the DRS will connect directly to the corresponding environmental node and retrieve an updated value for the humidity. Then, the Data Retrieval Service delivers three different areas with different aspect measurements (Figure 11). In this case the coverage areas are considered rectangular for convenience; however they might be updated to cyclic areas in case that such a representation is more realistic for the Fusion Service. To be noted that in the example of Figure 11 the route was not segmented to smaller parts by the intersection service. |
| Comments | It is beyond the scope of this service to provide any fusion of the obtained results. |

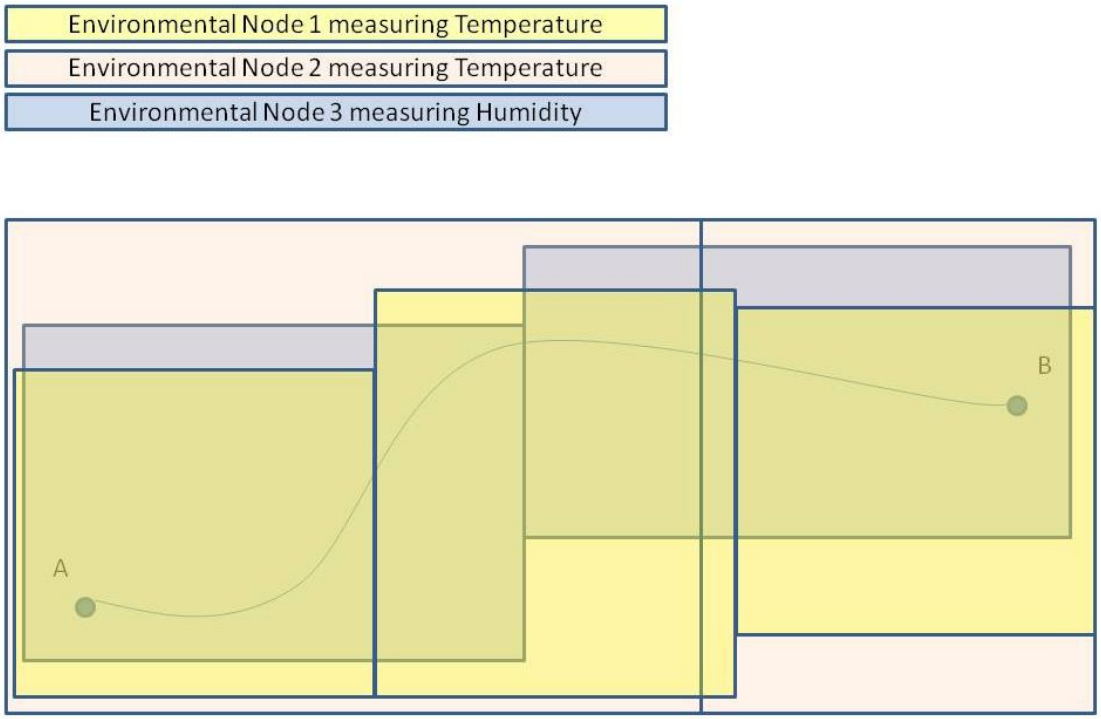


**Figure 11: Travelling from A to B and the corresponding nodes represented by their coverage area, while the colour denotes the type of the service.**

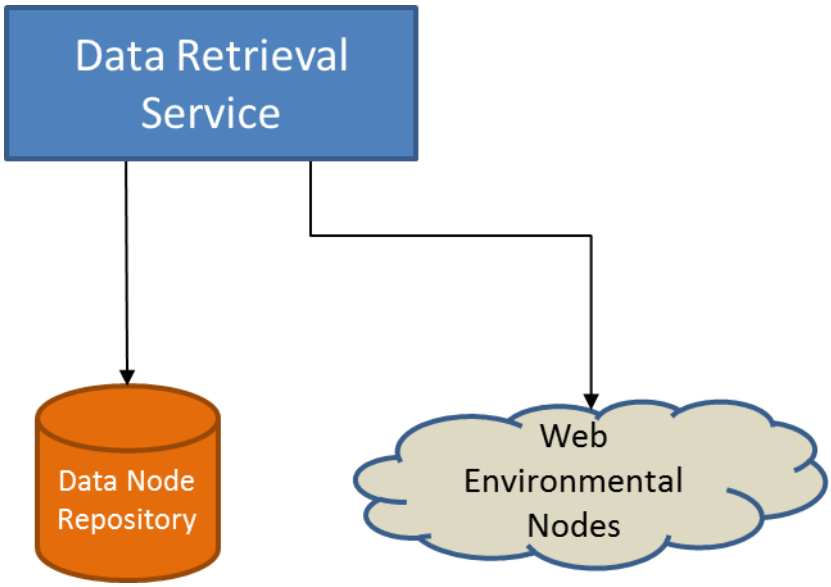


**Figure 12: Node access of the DRS**

## 5.4 Decision Service (DS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Decision Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL |

| | |
|---|---|
| | • HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>• PESCaDO-PDL (to become standard in PESCaDO)<br>and likely some others. |
| Description | The Decision Service retrieves the content that is relevant to the solution of the user's problem formulated in terms of PESCaDO-PDL statements. It thus<br>(xxiii) accesses the KB in order to inspect the available knowledge elements and to determine their relevance to the problem of the user;<br>(xxiv) makes inferences to deduce knowledge elements not explicitly available in the KB;<br>(xxv) makes inferences to deduce the necessity to solicit missing data from the Data Retrieval Service;<br>(xxvi) solicits data from the Data Retrieval Service.<br>The Decision Service provides its functionality through the following interfaces:<br>• *ObtainRelevantContent*.<br>The Decision Service interacts with the following services:<br>• User Profile Management Service<br>• Knowledge Base Access Service<br>• Data Retrieval Service<br>• Answer Service |
| Interface *ObtainRelevantContent* | |
| *computeRelevantContent* | Provides to the Answer Service the content that is considered relevant to the solution of the problem of the user.<br>Takes as input:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query which references also the contextual settings, including the profile of the user.<br>• The Knowledge Base.<br>The output of the function is:<br>• None (the decision service will write the inferred facts directly in the Knowledge Base containing the data for the current query – everything will be reachable from the instance representing the problem). |
| Comments | None at the moment. |

## 5.5 Fusion Service (FS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Fusion Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF |
| Description | The general task of the FS is to process the data received from different environmental service nodes via the Data Retrieval Service with the purpose to obtain the best and most complete data set to address the user's request for information or decision support. It thus<br>(xxvii) identifies complementary pieces of data proceeding from different data sources of the same kind (i.e., AQ, meteorology, traffic, etc.) and fuses them into one coherent data block;<br>(xxviii) identifies alternative (or competing) pieces of data proceeding from different data sources, but concerning the same environmental phenomenon for the same location, assesses the quality of each of them using the two types of uncertainty metrics developed in PESCaDO (imprecision and confidence metrics), and selects the best for communication to the user.<br>The Fusion Service provides the functionality through the following interfaces:<br>• *FuseData*: Delivers the fused dataset.<br>The Fusion Service interacts with the following services:<br>• Data Retrieval Service<br>• Knowledge Base Access Service<br>• User Profile Management Service<br>• User Interaction Service |
| Interface *FuseData* | |
| *getFusedData* | Takes as input:<br>• The data from the data retrieval service and pointers to the corresponding service node fingerprints.<br>• Potential preferences of the user for specific service nodes.<br>The output of the function is:<br>• A fused data set. |
| Comments | None at the moment. |

## 5.6 Information Production Service (IPS)

| | |
|---|---|
| Standard Specifications | PESCaDO's Information Production Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• W3C SWRL<br>• HTTP (Hypertext Transfer Protocol)<br>• W3C SPARQL Query Language for RDF<br>and likely some others. |
| Description | The Information Production Service receives as input the Content Plan compiled by the Content Selection Service, the user profile and potentially other contextual setting restrictions and produces user problem targeted multimodal information as text, table and graphic, which is passed to the Answer Service.<br>The Information Production Service provides its functionality through the following interfaces:<br>• *Generate Information*.<br>The Information Production Service interacts with the following services:<br>• User Profile Management Service<br>• Content Selection Service<br>• Answer Service<br>• User Interaction Service<br>External resources required by the Information Production service:<br>• Annotated corpora |
| Interface *GenerateInformation* | |
| *getInfo* | Takes as input a content plan, the user profile and contextual settings and provides information in one of the three modes (text, graphic, table) – or in a combination of thereof – in one of the languages of PESCaDO (Finnish, English or Swedish). |
| Comments | None |

## 5.7 Intersection Service (IS)

| | |
|---|---|
| Standard Specifications | No corresponding standard is known. |
| Description | The Intersection Service segments the route of interest into different parts (i.e. areas or smaller routes) based on ground and road condition parameters (e.g. normal traffic, width of roads, etc.). It should be noted that the parameters for |

| | |
|---|---|
| | segmenting a route are subject to the requirements for fusion. It takes as input:<br>• The route, which is provided by the Route Calculation Service.<br><br>The service processes this information in order to output non-overlapping geographical areas for the aspects of interest. In every segmented area, different measurements may be reported for the same aspect. These values are fused at a later stage by the Fusion Service.<br>The Intersection Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs about the common and specific capabilities.<br>• *SegmentationService*: This interface allows a client to make requests and receive areas of interest. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of a Data Node Repository Service instance. These include the functionalities of the service as well as the input and output specifications. |
| Interface *SegmentationService* | |
| *getSegmentedAreas* | This functionality returns the coordinates of each segmented area. |
| *getAspectsValues* | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Example usage | In the usage scenario of Figure 3, the Intersection Service receives the input by the Route Calculation Service and segments the route into smaller parts or areas. |
| Comments | It is beyond the scope of this service to provide any fusion of the obtained results (i.e. measurements of aspects). For each intersection area, it is possible that we have multiple measurements for the same aspect when the information is retrieved by different nodes.<br>This service can either provide direct input to the Fusion Service or return the result to the Data Retrieval Service.<br><br>*At the current stage of the project the exact functionality of the IS is not exactly clear as it is highly based on the requirements for fusion. Also the interaction with the Fusion Service needs further research.* |

## 5.8 Knowledge Base Access Service (KBAS)

| | |
|---|---|
| Standard Specifications | The Knowledge Base for the PESCaDO platform is currently planned to be based on the following ontology language:<br>• W3C OWL Web Ontology Language http://www.w3.org/TR/owl-features/<br>Relevant standard specification for this service are also:<br>• W3C RDFS http://www.w3.org/TR/rdf-schema/ |

| | |
|---|---|
| | • W3C RDF/XML Syntax Specification (Revised) http://www.w3.org/TR/rdf-syntax-grammar/<br>• W3C SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/<br>• W3C SWRL: A Semantic Web Rule Language Combining OWL and RuleML http://www.w3.org/Submission/SWRL/<br>• W3C URIs, URLs, and URNs: Clarifications and Recommendations 1.0 http://www.w3.org/TR/uri-clarification/ |
| Description | The Knowledge Base Access Service supports the access to the Knowledge Base on which the whole PESCaDO platform relies to provide user decision support. This service provides, via the *KBAccess* interface, access, manipulation and storage of ontologies (or ontology modules) stored in the Knowledge Base. Besides the common *KBAccess* interface retrieval methods, the *KBQuery* interface offers a generic mechanism for retrieval.<br><br>Some typical uses of this service are:<br>• Storing, updating or deleting available ontology modules;<br>• Retrieving (partially or fully) a stored ontology module;<br>• Getting high-level information about some ontology module, such as the list of concepts, or the list of supported properties for a given concept;<br>• Inserting in the knowledge base actual data, e.g. those retrieved from the Data Repository, which will be used to fulfil the user problem request;<br>• Querying one of more ontology modules.<br><br>The Knowledge Base Access Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities*: Informs the client about the common and specific capabilities of the Knowledge Base Access Service.<br>• *KBAccess*: Supports the management (storage, retrieval, and deletion) of ontology modules, provides a high-level view on ontologies, and allows for adding factual knowledge (data from the data repository, inferred data) to the knowledge base.<br>• *KBQuery*: Providing operations to query the ontology modules contained in the knowledge base. |
| Interface *ServiceCapabilities* | |
| *get Capabilities* | Informs the client about the common and specific capabilities of a Knowledge Base Access Service instance. Examples of specific capabilities are:<br>• The names of the ontology modules available in the Knowledge Base. |

| | |
|---|---|
| | • The possible representation formats of query results which can be requested by clients.<br>• The query languages that can be used in knowledge base queries.<br>• The inference capabilities of the knowledge base applied when computing query results. |
| Interface *KBAccess* | |
| *setOntology* | Stores a new ontology module in the Knowledge Base. |
| *getOntology* | Retrieves an existing ontology module or a selection of an ontology module from the Knowledge Base. |
| *deleteOntology* | Removes an existing ontology module from the Knowledge Base. |
| *getTBox Vocabulary* | Given an ontology or a part (segment) of an ontology, returns a list of TBox statements ready to be used for creating a Knowledge Base. |
| *getABox Vocabulary* | Given an ontology or a part (segment) of an ontology, returns a list of ABox statements ready to be used for creating a Knowledge Base. |
| *addABoxStatement* | This function allows for inserting into the knowledge base the factual knowledge (data) that will be used to fulfil a user request (e.g. temperature values, pollutants concentrations). The data to be added may be, e.g. those returned by the Data Retrieval Service/Fusion Service, and the data inferred by the Decision Service. |
| *getListConcepts* | Returns a list of concepts of an ontology (module) or a part (segment) of an ontology (module). |
| Example usage | The *setOntology* operation may be used, for instance, when a new version of an ontology module has to be uploaded in the Knowledge Base. Other services of the PESCaDO platform (e.g. Fusion Service, Decision Service), may need to process all the information contained in an ontology modules to perform the operations needed to be able to invoke the *getOntology* operation. |
| Comments | Creating or editing of ontology modules is out of the scope of the Knowledge Base Access Service: this service's interface manages only the storage and the access to ontologies, but doesn't provide any tool to create ontology structures. |
| Interface *KBQuery* | |
| *queryOntology* | Submits a query to the ontologies stored in the knowledge base. The query has to be formulated in a query language (e.g. SPARQL) compatible with the knowledge representation model used by the knowledge base (e.g. RDF/OWL). |

| | |
|---|---|
| Example usage | The *queryOntology* operation may be invoked by other services like the Fusion Service, Decision Service or the Problem Description Generation Service to obtain the relevant information they need to take some decisions or perform some reasoning. For example, the Problem Description Generation Service may use this operation to retrieve from the corresponding ontology module in the Knowledge Base the relevant data for preparing a complete user problem description. |
| Comments | |

## 5.9 Problem Description Generation Service (PDGS)

| | |
|---|---|
| Standard Specifications | No specific corresponding standard known.<br>The Problem Description Language (PDL) is planned to be based on:<br>• W3C XML Extensible Markup Language http://www.w3.org/XML/ |
| Description | The goal of this service is to generate / check a description of the problem in PDL submitted by the user to the PESCaDO platform, according to the information provided by the user via the user interface / third party systems.<br>The Problem Description Generation Service provides the functionality through the following interfaces:<br>• ServiceCapabilities: Informs about the common and specific capabilities.<br>• *DescriptionGeneration*: generates a description in PDL of the problem submitted by the user to the PESCaDO platform, according to the information provided by the user via the user interface.<br>• *DescriptionChecking*: checks that a problem description expressed in PDL and submitted by third party systems is complete, i.e. no user information needed as part of the request is missing. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of a Problem Description Generation Service instance. Examples of specific capabilities are the type of problems that can be formulated, the required input parameters needed for a specific problem, and the kind of output a specific problem returns. |
| Interface *DescriptionGeneration* | |
| *generateProblemDescription* | This operation returns a structured description in PDL of the problem submitted by the user[2]. Input parameter for this operation is the selected problem, together with a list of input data provided by the user via the user interface. The operation checks that all data needed to compile a complete problem description are available within the input parameter |

[2] Note that this means that an URI, which is the identifier of an individual of the problem in the knowledge base, is returned, not the document itself.

| | |
|---|---|
| | list. To perform this task, the operation needs to access (via the Knowledge Base Access Service) the ontology which describes the problem/data dependencies. If some required data has not been provided in the input, the operation can retrieve the missing data by accessing the User Profile via the User Profile Management Service.<br><br>Input parameters:<br>• The URI of the type of problem type selected by the user<br>• The URI of the type of activity selected by the user<br>• 0..n additional parameters related to the problem type<br>Returned value:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query. The PDGS will create also new instances and facts about the problem, and write the problem description directly in the Knowledge Base containing the data for the current query, but just the URI of the problem description will be returned to the AS. |
| Example usage | The typical application scenario for this service interface is that the *generateProblemDescription* operation is invoked by the Answer Service with the input data on the problem to be solved collected via the user interface. |
| Comments | None at the moment. |
| Interface *DescriptionChecking* | |
| *checkProblemDescription (optional)* | This operation checks whether a problem description expressed according to the PDL is complete (i.e. that there is no user input data missing). The input parameter for this operation is a description of a user problem in the PDL. The output of the operation is a yes/no answer. In the case of a negative answer, a description of the missing parameter can optionally also be returned. |
| Example usage | A typical application scenario for this service interface is that the *CheckProblemDescription* operation is invoked by the Answer Service in the case of user decision support requests not submitted via the PESCaDO platform user interface, like e.g. batch submission of third party organisation / system. |
| Comments | None at the moment. |

## 5.10 Related Aspects Computation Service (RACS)

| | |
|---|---|
| Standard Specifications | No specific corresponding standard known.<br>(Data format to query the knowledge base?) |
| Description | The Related Aspects Computation Service (RACS) is a service that, given the problem description generated for the request made by the user, selects the aspects that need to be considered for delivering an informative answer to the user.<br>The input of the service is planned to be an *action* |

| | |
|---|---|
| | *specification* (e.g. “travel from X to Y”) and the *query focus* (or topic, e.g. “health issues”) submitted by the user, as encoded in the PDL (Problem Description Language) format by the Problem Description Generation Service. The output is a set of focus-related aspects that have to be considered before an answer to the user query is given.<br>Given the action and focus of the user query, the service accesses the Knowledge Base containing general information about aspects relevant to an <action, focus> couple. The Knowledge Base returns a list of relevant aspects, with a possibly empty list of specific constraints that need to be verified to assess the relevance of an aspect for a specific user associated with each of the aspects in question. The RACS will filter the list of aspects returned by the Knowledge Base by matching the constraints associated with each aspect against the information contained in the user profile.<br>The RACS provides the functionality through the following interfaces:<br>• *RelevantAspects*: accesses the knowledge base retrieving all aspects relevant to the focus associated with the query.<br>• *InformationFiltering*: compares the focus-related aspects with the user profile and filters them in order to keep only the relevant ones. |
| Interface *RelevantAspects* | |
| *getRelevantAspects* | Given the action and focus of the user query, the operation returns a set of related aspects. Each aspect can have an associated list of relevance constraints to be matched against information contained in the user profile.<br><br>Input parameters:<br>• The URI of the instance representing the problem in the knowledge base containing the data for the current query;<br><br>Returns a list of couples (0..n):<br>• The URI of the entity in the KB corresponding to a specific aspect (e.g. http://www.pescado-project.eu/ontology/pescado.owl#precipitation or http://www.pescado-project.eu/ontology/pescado.owl#O3)<br>• The URI of the class in the KB representing the type of the aspect (e.g. http://www.pescado-project.eu/ontology/pescado.owl#AirQualityType or http://www.pescado-project.eu/ontology/pescado.owl#WeatherType) |
| Interface *InformationFiltering* | |
| *filterClasses* | The aspects retrieved from the *RelevantAspects* interface are compared against the user profile information and only those that are relevant to the user are returned. |
| Example usage | A requestor wants to know if it is safe for him to travel from |

| | |
|---|---|
| | place X to destination Y. The user profile of the requestor says that he is a middle-age man with allergy to pollen and no other disease. RACS is asked to find aspects which are relevant to the requestor given the action "travel from X to Y" and the focus "health issues". RACS accesses the Knowledge Base and finds that generally speaking "pollen" and "heat" are relevant aspects for a person that wants to travel and is concerned about health. The Knowledge Base also provides the information that "pollen" aspects are relevant only to people who have pollen allergies, and the "heat" aspect is relevant only if the user is old or has heart diseases. The RACS accesses the user profile and finds that the "allergy" constraint is matched (the user is allergic to pollen), whereas the "aged or heart disease" condition is not matched. The heat factor is not dangerous and can be filtered out. RACS outputs the pollen aspect as the only one which is relevant to the requestor and that needs to be further analyzed. |
| Comments | None at the moment. |

## 5.11 Route Calculation Service (RCS)

| | |
|---|---|
| Standard Specifications | Relevant standard specification for this service are:<br>• OpenGIS Geography Markup Language (GML) Encoding Standard<br>• Open Street Map http://www.openstreetmap.org/ |
| Description | The Route Calculation Service provides a range of services based on (freely) available spatial data, e.g. data provided by Open Street Map.<br>The Route Calculation Service provides the functionality through the following interfaces:<br>• *ServiceCapabilities:* Informs about the common and specific capabilities.<br>• *LocationUtility*: Provides a Geocoder/Reverse Geocoder.<br>• *RouteCalculation:* Determines travel routes and navigation information. |
| Interface *ServiceCapabilities* | |
| *getCapabilities* | Informs the client about the capabilities of the Route Calculation Service, e.g. the supported output formats of calculated routes (e.g. GML/XML, GPX). |
| Interface *LocationUtility* | |
| *geoEncode* | The geoEncode operation transforms a description of a location, such as a place name, street address or postal code, into a normalized description of the location as point geometry. |
| *geoDecode* | The geoDecode operation transforms a normalized description of the location into a description of a location. |
| Interface *RouteCalculation* | |

| | |
|---|---|
| *calculateRoute* | The calculateRoute operation receives as input a start point and an end point. Moreover, diverse criteria such as the travelling means can be entered, e.g.<br>• car: fastest<br>• car: shortest<br>• bicycle<br>• walking<br>The operation then calculates automatically the route and returns it in the supported output format for further processing by other services or for rendering on a map by means of a user interface client. |
| Example usage | In a typical scenario, a user submits a request to the *Answer Service* that he/she wants to travel from point A to point B. To check the areas that the user will cross during his journey for potential health related aspects (e.g. pollen in case of an allergy, dangerous roads in winter time, etc.), the route must be known first. |
| Comments | • Further parameters which could be supplied to the calculateRoute operation (e.g. transit-stops, avoidance of motorways, avoidance of areas, etc.) need further discussion. Optionally, there should be a helper function for auto completion of road and city names.<br>• Ongoing research in PESCaDO might show that it would be better or sufficient not to have a route calculation but instead have a list of affected areas. To facilitate this option, e.g. the knowledge base must contain an area map (of Finland). |

## 5.12 User Interaction Service (UIS)

| | |
|---|---|
| Standard Specifications | PESCaDO's User Interaction Service uses the following standards for communication purposes with other services and modules:<br>• XML (Extensible Markup Language)<br>• HTTP (Hypertext Transfer Protocol)<br>and likely some others. |
| Description | The User Interaction Service has two main functions. First, it ensures the communication between the expert and end users on the one side and the services that involve the intervention of the user for their optimal functioning on the other side. For the first version of the architecture of PESCaDO, these are: (i) the Fusion Service; (ii) the Content Selection Service, and the (iii) the Information Production Service. In the subsequent versions, the Data Node Retrieval Service will also make use of the UIS.<br>Second, it supports the communication with the end user and the Answer Service.<br>The User Interaction Service provides its functionality |

| | |
|---|---|
| | through the following interfaces:<br>• *FusionAndMetricsSupport*<br>• *ContentSelectionSupport*<br>• *DiscoursePlanningSupport*<br>• *AnswerServiceSupport*<br>The User Interaction Service interacts with the following services:<br>• Fusion Service<br>• Content selection Service<br>• Information Production Service<br>• Answer Service |
| Interface *FusionAndMetricsSupport* | |
| *displayMetrics* | Visualizes the metrics to the user. |
| *displayFusionSchema* | Visualizes to the user the picture of how data from specified data sources are fused to obtain the optimal value. |
| *modifyMetrics* | Allows the user to modify (in a graphical mode) the uncertainty / confidence metrics. |
| *modifyFusionSchema* | Allows the user to modify (in a graphical mode) the data fusion schema. |
| Interface *ContentSelectionSupport* | |
| *displayContent* | Displays a content element or the whole content plan to the user. |
| *getRelevance* | Requests the judgement of the user concerning the relevance of a content element (can be positive or negative). |
| Interface *DiscoursePlanningSupport* | |
| *displayDiscourse* | Displays a discourse structure or a discourse element to the user. |
| *getDiscourseRelation* | Requests the assignment of a discourse relation to a pair of content elements (the content elements can be complex). |
| *orderDiscourse* | Requests the (linear) ordering of several discourse elements among each other. |
| Comments | The offered functionality of the Interface *FusionAndMetricsSupport* is to be defined. |
| Interface *AnswerServiceSupport* | |
| *displayQuestionTypes* | Displays the types of inquiries a user can submit to the system |
| *selectQuestionType* | Prompts the user for the selection of one specific query type |
| *displayInputFormat* | Shows the format in which the user has to provide their problem description. |
| *setFormData* | Prompts the user for entering the data of its problem description. |
| Comments | None at the moment. |

## 5.13 User Profile Management Service (UPMS)

| | |
|---|---|
| Standard Specifications | Relevant standard specification for this service are:<br>• W3C RDFS http://www.w3.org/TR/rdf-schema/<br>• W3C RDF/XML Syntax Specification (Revised) http://www.w3.org/TR/rdf-syntax-grammar/<br>• W3C SPARQL Query Language for RDF http://www.w3.org/TR/rdf-sparql-query/<br>• W3C SWRL: A Semantic Web Rule Language Combining OWL and RuleML http://www.w3.org/Submission/SWRL/<br>• W3C URIs, URLs, and URNs: Clarifications and Recommendations 1.0 http://www.w3.org/TR/uri-clarification/<br>• W3C OWL Web Ontology Language http://www.w3.org/TR/owl-features/ |
| Description | The PESCADO *User Profile Management Service* manages user profiles. User profiles can be created, written, deleted and queried.<br>The service also supports anonymous (temporary) users and default profiles for default user categories (like infant, kid, teenager, adult, senior, etc.).<br>The user profiles are stored in the Knowledge Base of the PESCaDO platform. Each user is identified by a unique ID (an URI of the knowledge base entity). A user may also be identified by an ID which is created outside of the UPMS and is given as a parameter on profile creation.<br>The User Profile Management Service provides the functionality through the following interfaces:<br>• Management: Handles creation, storage and deletion of user profiles<br>• Query: Querying of user profiles |
| Interface *Management* | |
| *createProfile* | Creates a new profile for a user. As an optional parameter, an external ID may be given which identifies the user and can be used alternatively in all functions. The operation returns an URI which identifies the user and can be used for accessing the user profile. |
| *createTempProfile* | Creates a new temporary profile for an anonymous user. Works like *createProfile* but does not allow the storage of an external ID. Temporary profiles are automatically deleted after a configurable time. |
| *createDefaultProfile* | Creates a new default profile for a user category. Works like *createProfile* but does not allow the storage of an external ID. The operation requires the user category as a parameter. |
| *deleteProfile* | Deletes an existing profile. |
| *changeProfileFeature* | Changes a setting in an existing profile. |

| | |
|---|---|
| *storeProfile* | Writes new user settings into an existing profile. The settings may be provided as an OWL document or as name/value-pairs. |
| *getUserProfile* | The user profile is retrieved, containing all personal data of the user, their preferences, health issues, etc. |
| *getUsers* | Get a list of all user IDs for which a user profile exists. |
| *hasProfile* | Check if a user profile exists for a given user. |
| *getProfileOntology* | Get the ontology in OWL format which defines a user profile. |
| Interface *Query* | |
| *getProfile* | Get the whole profile of a user as an OWL document. |
| *getDefaultProfile* | Get a default profile for a given user category. |
| *query* | The query has to be formulated in a query language (e.g. SPARQL) compatible with the knowledge representation model used by the knowledge base (e.g. RDF/OWL). See the *Knowledge Base Access Service* for further information. |
| Example usage | The user interface calls the management operations of the service to allow the user the configuration of his profile. Other services such as the *Decision Service* query the user profile to adapt their functionality to the users' needs. |
| Comments | None at the moment. |

# 6 Technology Viewpoint

## 6.1 SOA Technologies

For the orchestration of the PESCaDO services, the Business Process Execution Language (BPEL) may be the potential language of choice. BPEL, short for *Web Services Business Process Execution Language* (WS-BPEL), is an OASIS standard executable language for specifying interactions with Web Services [Alves, 2006]. Processes in Business Process Execution Language export and import information by using Web Service interfaces exclusively.
WS-BPEL is the most credited language for expressing concrete service chains; among other languages are XLANG and WSFL (WS-BPEL inherits all the main design constructs of both languages). Currently, WS-BPEL is the only one which is continuously extended (currently 2.0 is available), and it is also the only one equipped with stable engines able to execute the service chain. Since service engines supply the execution of aggregated services by means of a single service, the WSDL standard is used to describe the corresponding interface.
Nevertheless, BPEL code tends to become quite complex – especially if a sufficient error handling is introduced. Therefore, other techniques may be considered as well.
The Web Service interfaces will be described with WSDL 2.0 (Web Services Description Language) and accessed via SOAP[3] 1.2 (over HTTP). Implementations of the SOAP protocol are available for a variety of programming languages like Java, C++, C#, etc. This enables the PESCaDO partners to implement services in their preferred programming language.

## 6.2 The Knowledge Base

For the representation of ontologies, PESCaDO uses *OWL* (Web Ontology Language), which is currently the most widespread standard (cf. also Section 4.2.7 above).
For accessing OWL ontologies via a programming interface, the *OWL API* will be made use of. The OWL API is a Java API and reference implementation for creating, manipulating and serialising OWL Ontologies. It is evolving into a de facto standard at the moment. Thus, it is already used by the most widespread Ontology editor Protégé[4].
The latest version 3.x of the API is oriented towards OWL 2 and provides an interface for plugging in a reasoner. The reference implementation delivered with the API is an in-memory only implementation, which means that the data is lost after a restart. Also, the size of ontologies is limited to the main memory of the computer. To overcome this problem, Fraunhofer IOSB already implemented a persistence solution called *owldb*. It was developed in THESEUS, a research program initiated by the German Federal Ministry of Economy and Technology (BMWi) and is available under the LGPL from http://owldb.sourceforge.net/.
For the persistence of OWL ontologies, owldb uses a relational database. For the storage of OWL API entities and axioms, an object relational approach is used. This component guarantees that only the entities currently used are kept in memory. This allows for the use of large ontologies. Once stored, the axioms and entities of an ontology can be retrieved directly from the database and the ontology can be directly manipulated within the database. Furthermore, it is possible to perform format conversions from all ontology formats currently supported by the OWL API to the new database format and vice versa.

[3] SOAP was originally defined as Simple Object Access Protocol but this acronym was dropped with Version 1.2 of the standard
[4] http://protege.stanford.edu/

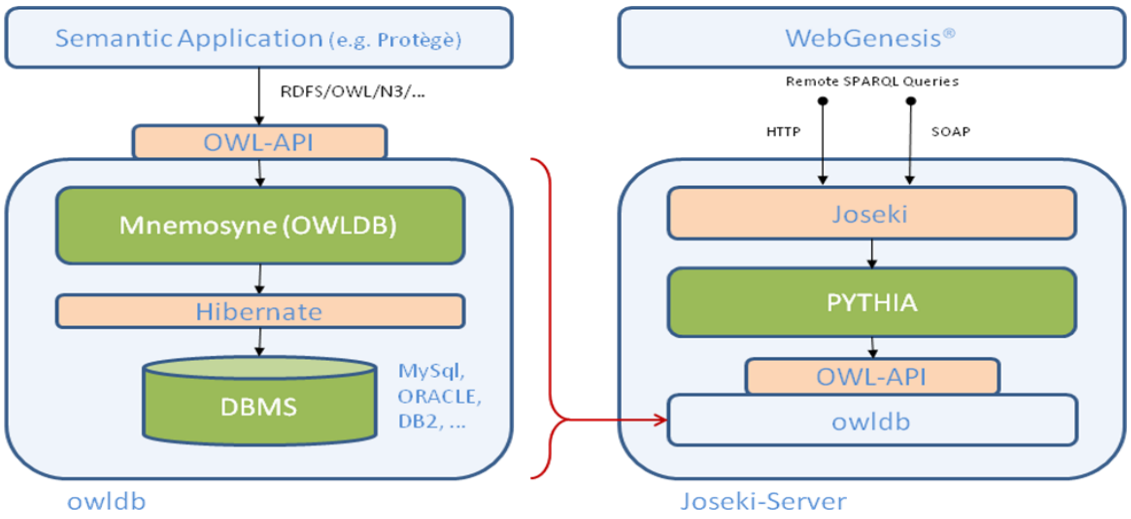


Figure 13: Ontology Persistence Stack

The left part of Figure 13 shows the internal architecture of owldb and how it is integrated into a semantic application. The right part displays another outcome of the THESEUS program called Pythia. Pythia is a plug-in for the open source SPARQL server *Joseki*[5], which supports the W3C SPARQL remote protocol. By plugging Pythia into Joseki, one can quickly provide OWL ontologies as SPARQL endpoints, which can be queried by clients that support protocols such as the Fraunhofer IOSB product WebGenesis®.

## 6.3 User interface technologies

The user interface of the PESCaDO Workbench will be run in a Web Browser. To make the user interface accessible to as many different browser implementations as possible, we will use standardized technologies like HTML and CSS.
For advanced interface techniques, a technology like Flash, which provides more advanced programming possibilities, might be necessary. Nevertheless, a fallback option should be available to access the functionality, even accepting a loss of comfort.

## 6.4 Further technologies

Further technologies are not decided upon yet. The following table displays the technologies that are in use by the PESCaDO partners in relevant technology fields.

| | IOSB | FBK | CERTH | UPF | USTUTT |
|---|---|---|---|---|---|
| OS | Windows<br>Linux | Windows<br>Linux<br>Mac OS<br>Sun OS | Windows<br>Linux | Windows<br>Linux | Windows<br>(Linux) |
| Programming Language | Java (JDK 6)<br>HTML/CSS/<br>JavaScript | Perl, Prolog<br>PHP, Java | PHP, Java, C<br>HTML/CSS/JavaScript | PHP, Java,<br>Perl,<br>JavaScript | Java<br>.NET, Flash |
| IDE | Eclipse | Eclipse | Eclipse | Eclipse | Eclipse |

[5] http://www.joseki.org/

| | | | | | |
|---|---|---|---|---|---|
| | | | | | Visual Studio, Flash Developer |
| DBMS | MySql<br>ORACLE<br>H2<br>Derby | MySql | MySql<br>PostgreSQL | MySql | Any |
| Application Server | Tomcat<br>Jetty | Apache Server, Tomcat PHP/Java-Bridge | Apache Server, Tomcat | Apache Server, Tomcat | Tomcat |
| Reasoner | Pellet | Pellet<br>Fact | - | - | - |
| WFS/WMS-Server | GeoServer | - | - | - | - |
| BPEL Engine | Apache ODE<br>OpenESB | - | - | - | - |

# 7 Engineering Viewpoint

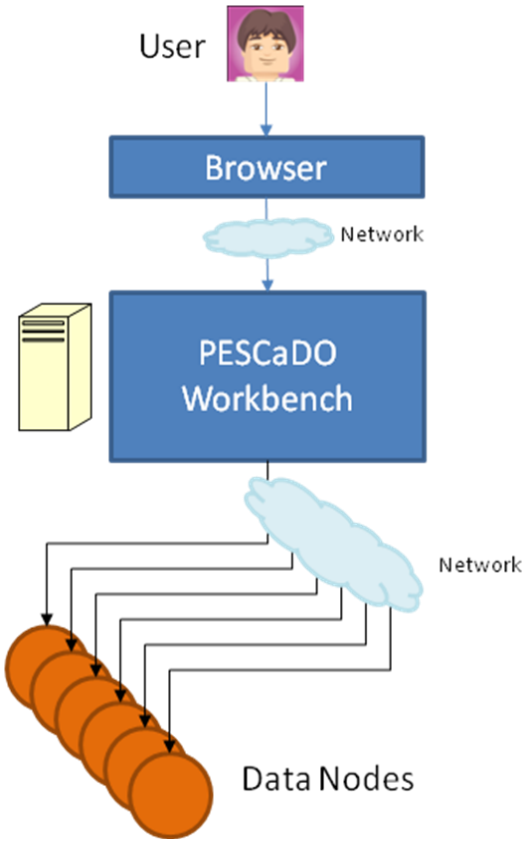


Figure 14: Hardware and Network view on the PESCaDO Workbench

Figure 14 shows the PESCaDO Workbench running on a single computer. Due to the service oriented architecture, on the other extreme, each service could be run on a dedicated hardware. The distributed scenario, with a single machine running only one or few PESCaDO services, is depicted in Figure 15. The concrete distribution of the services across the hardware will depend on the performance requirements of each specific service. These requirements are not known yet. For simplicity reasons, the strategy is to start with as few machines as possible and add new hardware for performance intensive services if required.

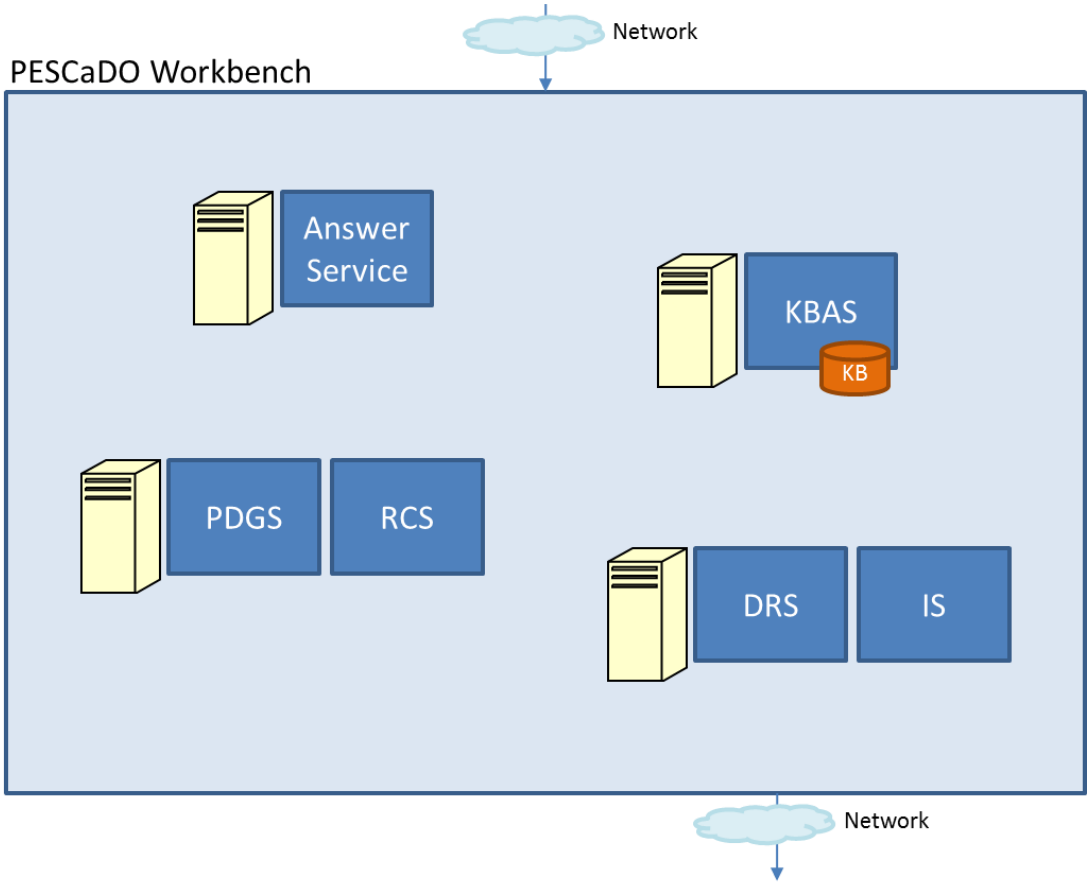


Figure 15: Possibilities for distributing the PESCaDO services on specific hardware

# 8 Description of Demonstrators

## 8.1 Description of the D8.6 Demonstrator

The D8.6 demonstrator shows the functionality envisaged for Milestone 2. MS2 marks the completion of the setup of the computational infrastructure of the PESCaDO system:

a. Presentation of the completion of the data and semantics representation framework evaluation, demonstration of the first proof-of-concept adaption and of the first proof-of-concept local content and data repository set up.
b. Demonstration of the availability of the skeletons of each module (1. discovery and targeted environmental node search, 2. service confidence assessment, 3. service node orchestration, 4. ontology web search, 5. ontology alignment and learning, 6. content distillation, 7. decision support, 8. visualization and user interaction, 9. information generation) and proof-of-concept implementations and their demonstration on restricted knowledge and data repositories, possibly with dummy data.
c. Demonstration of the setup of the communication and data.

All available technical features are demonstrated by using the Scenario presented in Section 3. The communication infrastructure connects the *User Interface* to the *Answer Service* which orchestrates the required services to solve the problem type “Health and Safety Related Decision Support” for the activities “Travel” and “Outdoor Activity”. The communication infrastructure uses SOAP calls via HTTP.
All involved services are present in the demonstrator and return the expected values. Already available functionality which goes beyond a *dummy* implementation (which means that the expected values of the scenario are hard coded) is described in the following sections.

### 8.1.1 User Interface

The following figures illustrate the possibilities of the user interface and the ideas of how to support a user in its intended task.

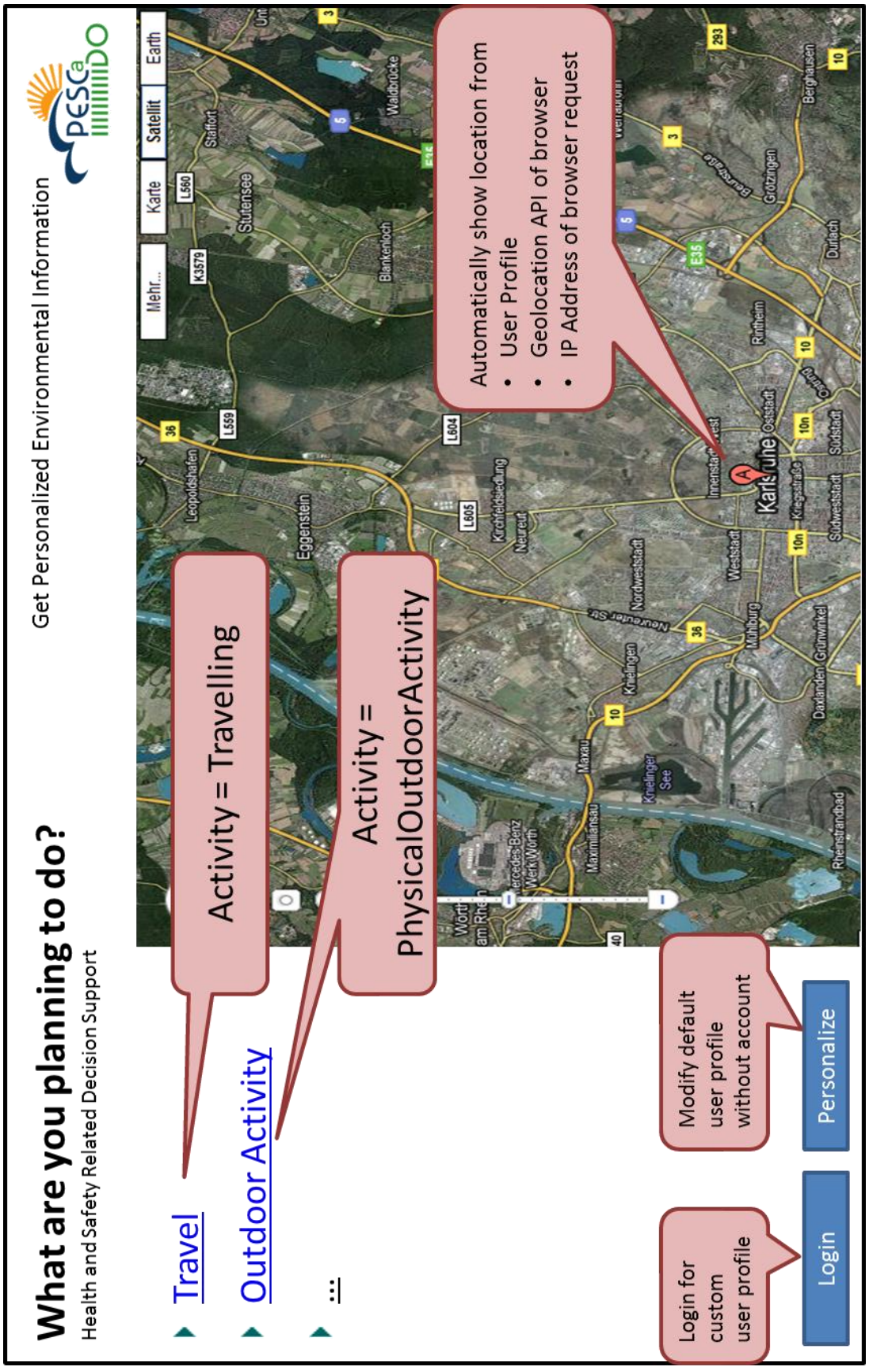


**Figure 16: Selection of the activity**

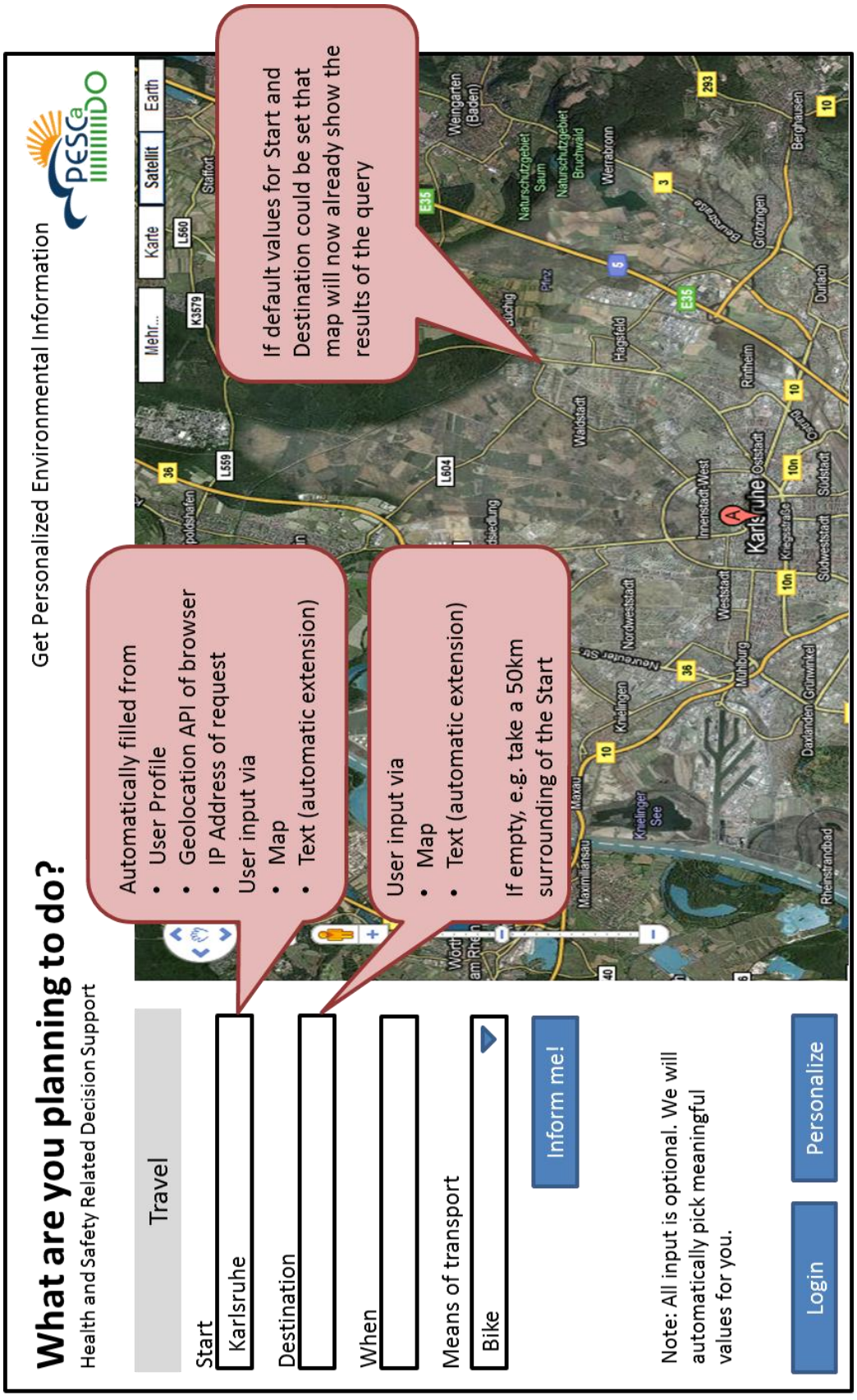


**Figure 17: Input form of the travel activity**

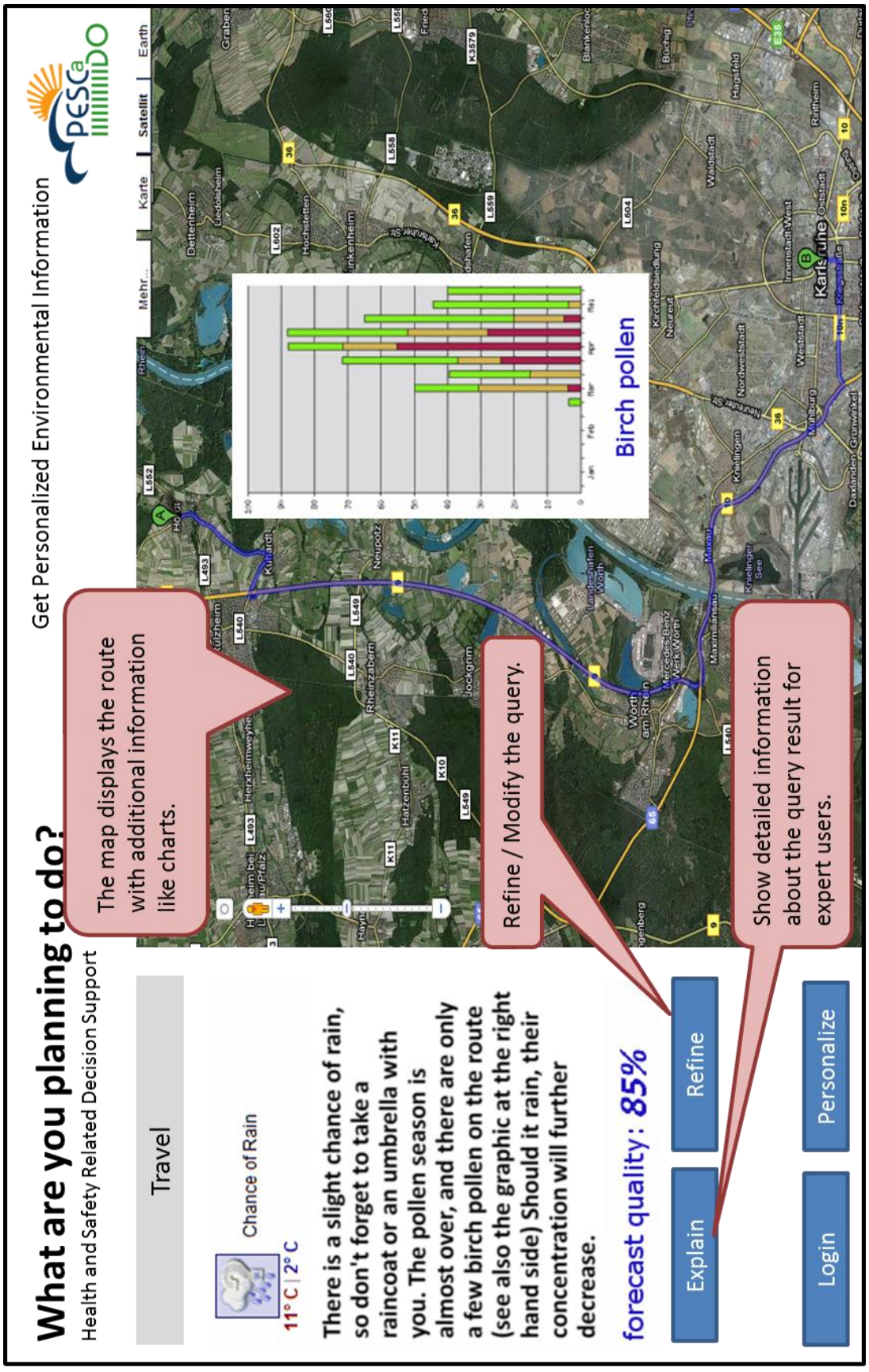


**Figure 18: The result of the decision support**

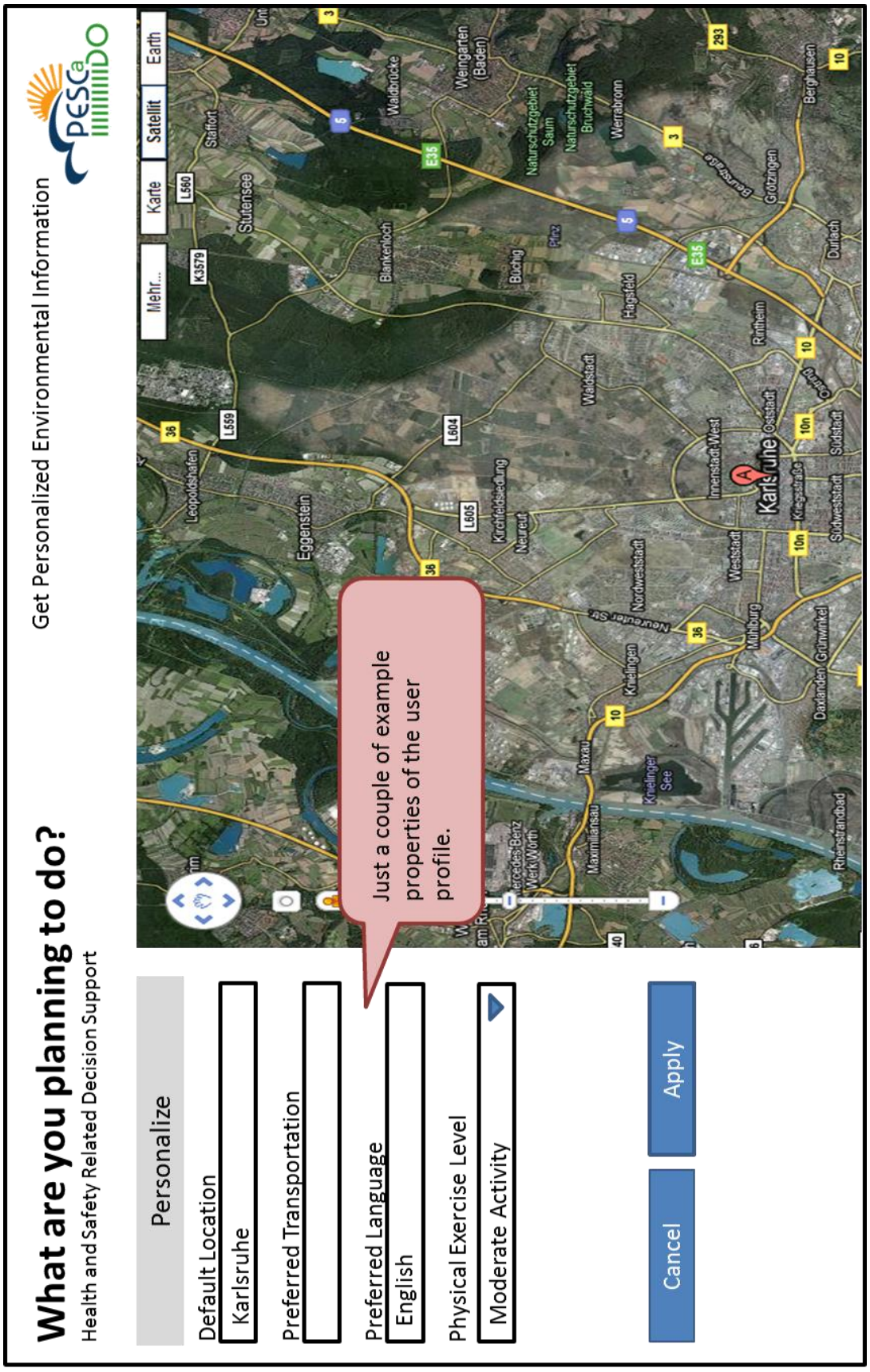


**Figure 19: Personalization of a user profile**

### 8.1.2 Answer Service

The Answer Service provides a basic plugin concept which allows the implementation of new orchestrations of existing PESCaDO services to fulfil new Use Cases. The first implemented plugin will call the required services for the scenario in the given order and hand over the required data. It uses SOAP calls to access the services.

### 8.1.3 Content Selection Service

The service receives a reference to the knowledge base repository, a user profile, and a query producing as result a set of content units appropriate to the input data. This service covers part of task 7.2 in Work Package 7.
The dummy functionality of this service is implemented. The implementation of an interactive reinforcement learning-based content selection strategy is under way.

### 8.1.4 Data Retrieval Service

The data retrieval service constitutes the front end of the node search and discovery infrastructure as the latter is setup in Task 2.1 of WP 2. In this demonstrator, we plan to show the whole functional architecture of environmental node search and discovery process as is depicted in Figure 20, which includes the DRS.

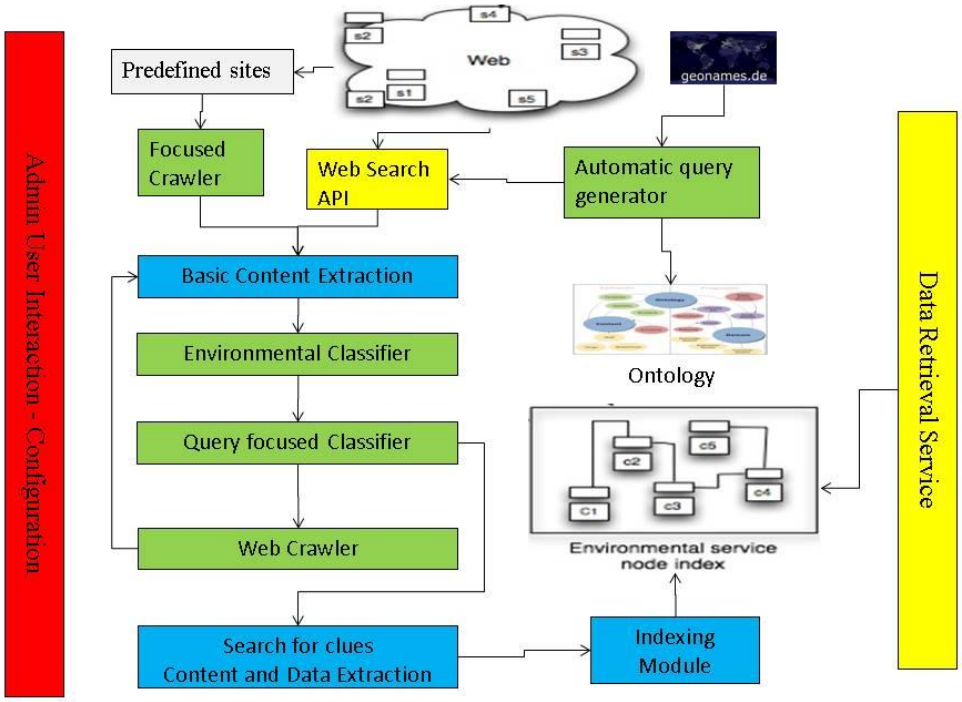


**Figure 20: Environmental Node Discovery Architecture**

The whole discovery process is coordinated by an interface through which an administrative user could intervene and optimize the performance of the system. This functional version of the infrastructure includes the following modules, the parameterization of which is accessible through the aforementioned interface:

- Dummy focussed Crawler implementation
- Functional general text web search
- Parsing and keyphrase extraction from webpages
- Limited functional extensive link web crawler

- Dummy classifier implementation
- Limited functional clues/keywords search in web content and metadata
- Dummy implementation of query expansion with keyword spice
- Functional module for language identification of web sites.
- Functional automatic query formulation based on the environmental ontology and geonames webservice (dummy interface implementation with ontology and geonames).
- Limited functional content extraction from web nodes
- Limited functional repository structure and indexing

The current version of the Data Retrieval Services is integrated with the PESCaDO architecture and supports the following functions:

- getData (limited functionality)
- getCapabilities (limited functionality)
- getSensor (dummy implementation)
- getStatistics (dummy implementation)

The existing information in the functional node repository includes manually inserted real data from FMI stations complemented by fake data.

### 8.1.5 Decision Service

The dummy functionalities of this service are implemented. This service does not provide its result as return values. Instead, the expected values of the scenario are written as static data in the Knowledge Base (e.g. assertions that relate quantitative measures to qualitative values, assertions that relate measures to warnings that should be reported, and so on).

### 8.1.6 Fusion Service

This service so far contains only a dummy implementation. It takes as input (1) data respectively content configurations stemming from different environmental service nodes, (2) relationship between the data/content (complementary or alternative), (3) characteristics of the environmental service nodes from which the data/ content has been obtained, and (4) uncertainty and confidence metrics for the assessment of the data/content and provides as output fused data / content.

### 8.1.7 Information Production Service

The service produces multilingual and multimodal output from a content plan produced by the content selection service. The service receives as input a set of content units, a user profile and a user query. The service covers tasks 7.3 (discourse planning) and 7.5 (purpose-driven multimodal generation) in Work Package 7. Resources developed in Task 4.3 will be used in this service.
For this demonstrator, an operational rule-based generator for the production of a selection of English and Finnish air quality reports has been implemented and, as a preparatory task for machine learning-based generation, a corpus of Finnish of about 1000 sentences has been annotated with POS and surface-syntactic structures.

### 8.1.8 Intersection Service

This service so far contains only a dummy implementation.

### 8.1.9 Knowledge Base Access Service

This service facilitates the components *Pythia* and *owldb* described in section *6.2 The Knowledge Base* to implement the operation *queryOntology*. By using this function other services can already query the knowledge base. Inserting statements into the knowledge base is currently done directly via the OWL-API interface.

### 8.1.10 Problem Description Generation Service

The dummy functionalities of this service are implemented. The service returns a unique identifier for the problem considered (URI corresponding to the problem individual generated in the knowledge), and writes the expected values of the scenario as static data in the Knowledge Base (e.g. it creates an individual for the problem, an individual for the type of request, it asserts that the request is associated to the problem, and so on).

### 8.1.11 Related Aspects Computation Service

The dummy functionalities of this service are implemented, thus returning the expected values of the scenario according to the interfaces (couples [specific aspect, aspect typology], like [PM10, Air Quality], [Temperature, Weather], and so on).

### 8.1.12 Route Calculation Service

The Route Calculation Service utilizes two internet services to fulfil its tasks. The *geoDecode* and *geoEncode* functions use the service *http://www.geonames.org/*. This service has a REST interface for which an API is available. It supports a lot of object types but seems mainly to contain cities. This means that the conversion of roads to geo coordinates and vice versa is not supported at the moment.
The *calculateRoute* function uses the online Google Direction API which does not require a Google key (in contrast to the Google Maps API). The returned values are already segmented which means that I only return the start and end of a step and not all of the points in that step (but this feature could be easily added if required).
Additionally the service contains an alternative implementation for routing which uses *Open Street Map* data from http://www.openstreetmap.org/. But the only freely available routing algorithm is rather slow and unstable.

### 8.1.13 User Interaction Service

The User Interaction Service is mainly a dummy implementation but already supports the basic querying for available question types (*displayQuestionTypes*) and returning the user inputs back to the Answer Service (*setFormData*).

### 8.1.14 User Profile Management Service

This service so far contains a dummy implementation for the management part of the service. The profile querying already accesses the knowledge base in which the user profiles are stored and can return a preconfigured default profile.

# 9 References

## 9.1 Glossary

| Acronym | Description |
|---|---|
| AJAX | Asynchronous JavaScript and XML |
| API | Application Programming Interface |
| BPEL | Business Process Execution Language |
| CSS | Cascading Style Sheets |
| DBMS | Database Management System |
| GML | OpenGIS Geography Markup Language Encoding Standard |
| HTML | Hypertext Markup Language |
| HTTP | Hypertext Transfer Protocol |
| IDE | Integrated development environment - a software development system |
| JSON | JavaScript Object Notation - a lightweight text-based open standard designed for human-readable data interchange |
| KB | Knowledge Base – a place where ontologies are stored |
| LGPL | GNU Lesser General Public License |
| ORCHESTRA | Open Architecture and Spatial Data Infrastructure for Risk Management - an European Integrated Project |
| OS | Operating System - the software that manages the hardware and software of a computer |
| OWL | Web Ontology Language – a language for authoring ontologies |
| PARCH | The PESCaDO Architecture |
| RDF | Resource Description Framework – a standard model for data interchange on the Web. |
| RDFS | RDF Schema - an extensible knowledge representation language, providing basic elements for the description of ontologies, otherwise called Resource Description Framework (RDF) vocabularies, intended to structure RDF resources. |
| REST | Representational State Transfer - a style of software architecture for distributed hypermedia systems such as the World Wide Web |
| RM-OA | Reference Model for the ORCHESTRA Architecture |
| RM-ODP | Reference Model for Open Distributed Processing |
| SANY | Sensors Anywhere - an European Integrated Project |
| SOA | Service Oriented Architecture - type of software architecture for creating and using business processes, packaged as services |
| SOAP | Simple Object Access Protocol - a protocol specification for exchanging structured information in the implementation of Web Services |
| SPARQL | A recursive acronym for 'SPARQL Protocol and RDF Query Language' – SPARQL is a query language for RDF graphs |
| SQL | Structured Query Language |
| SWRL | A Semantic Web Rule Language Combining OWL and RuleML |
| UI | User Interface |
| UML | Unified Modelling Language - an object modelling and specification language used in software engineering |
| URI | Uniform Resource Identifier – a string of characters used to identify a name or a resource on the Internet. |

| | |
|---|---|
| WFS | Web Feature Service |
| WMS | Web Map Service |
| WS | Web Service |
| WSDL | Web Service Description Language |
| XML | Extensible Markup Language |

## 9.2 Bibliography

[Alves, 2006] A. Alves (Ed.). Web Services Business Process Execution Language Version 2.0 - Public Review Draft, OASIS, 2006

[Dekker, 2008] M. Dekker. Global illustration of the RM-ODP Viewpoints, http://en.wikipedia.org/wiki/File:RM-ODP_viewpoints.jpg, 17 December 2008

[ORCHESTRA, 2008] ORCHESTRA consortium. ORCHESTRA - an open service architecture for risk management, 2008

[Usländer, 2007] T. Usländer (Ed.). Reference Model for the ORCHESTRA Architecture (RM-OA) V2 (Rev 2.1), Open Geospatial Consortium Inc., 2007