PESCaDO

FP7-248594

Personalized Environmental Service Configuration and Delivery Orchestration

SEVENTH FRAMEWORK PROGRAMME

D5.1 Decision support request and problem specification language

Due date of deliverable: 31.08.2010
Actual submission date: 01.09.2010

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

Lead contractor for this deliverable: FBK <Revision 1.6>

D5.1	Decision support request and problem specification language
Project Acronym :	PESCaDO
Contract No :	FP7 - 248594
Due Date :	31.08.2010
Reply To:	Marco Rospocher rospocher@fbk.eu
Actual date of delivery:	01.09.2010

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 8, 31.08.2010
Actual date of delivery	Month 9, 01.09.2010
Deliverable number	D5.1
Deliverable name	Decision support request and problem specification language
Type	Report
Status & version	Submitted, V1
Number of pages	40
WP / Task responsible	FBK
Other contributors	UPF, USTUTT, HSY, FMI
Author(s)	Marco Rospocher, Luciano Serafini, Leo Wanner, Harald Bosch, Maria Myllynen, Ari Karppinen
Internal Reviewer	Harald Bosch
EC Project Officer	Manuel Monteiro
Abstract	This deliverable defines the initial version of the user-oriented problem description language (PDL), the specification language that the user can use to describe his/her decision support request to be submitted to the system.

Table of Contents

Executive Summary

This document presents the PESCaDO Problem Description Language (PESCaDO PDL) developed within the European STREP FP7-248594 Personalized Environmental Service Configuration and Delivery Orchestration (PESCaDO) project. The objective of the PESCaDO PDL is to formally represent the decision support requests of the users to be submitted to the PESCaDO-system.

The specification of the PESCaDO PDL has followed an iterative process, comprising the following activities:

- The analysis of the user needs with respect to decision support, according to the first definition of the PESCaDO pilot use-cases;
- The survey of the state-of-the-art in problem description languages, with the goal to investigate the possibility of a reuse of generalisable features of PDLs developed in other application domains;
- The formal definition of the PESCaDO PDL;
- The analysis of the possible interaction mechanisms between the user and the PDL, in order to identify the adequate support to be provided to the users and thus to allow them to express their requests in an easy and user-friendly way.

This document contains the results of these activities and concludes by presenting an outline of the assessment criteria that will be applied to evaluate the performance of the PESCaDO PDL.

1 Introduction

A problem description language (PDL) is an integral part of many planning and production systems. Its main purpose is to provide an intermediate representation of a problem that is understandable and processable by both the machine and human beings.

In the PESCaDO-system, a PDL is needed to support the formulation of environmentally-oriented user decision support requests. That is, the user will formulate a request (e.g. a request for information, decision support, or notification in case of warnings) regarding environmental aspects using the PDL. The formulated request (or problem description) will then be processed by the system with the goal to provide an adequate reply to the request. Typical examples of user requests in the context of the PESCaDO decision support scenarios are:

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

Although some PDL proposals have been previously described in the literature and applied in implemented systems (see Section 3 for more details), to the best of our knowledge, no environmentally-oriented problem description languages have been worked on so far. Therefore, as part of the PESCaDO activities we have developed an environmentally oriented problem description language: the PESCaDO PDL.

According to the design of the PESCaDO architecture, the decision support to the user will be implemented by reasoning techniques and strategies based on the information stored in ontologies. In order to facilitate reasoning and navigation on ontologies, we have decided to base the PESCaDO PDL on OWL Web Ontology Language (see Section 4 for more details). However, it is important to underline that the PESCaDO users will not be required to write their requests directly in OWL. Rather, the PDL will be embedded in a visual problem specification environment (VPSE), which will provide visual metaphors for the formulation of requests by direct user interaction (see Section 5 for more details). The VPSE will allow for a flexible combination of graphical input elements, template input elements, and restricted natural language input elements, enabling the user to formulate expressive requests in an intuitive easy-to-use way.

As a first step towards the development of the PESCaDO PDL, an investigation of the user problem and decision support request scenarios has been performed. This investigation shall facilitate a broad understanding of the required expressiveness of the PDL. The result of this investigation is reported in Section 2.

Some performance assessment criteria have been defined in order to verify whether the PESCaDO PDL adequately covers the needs of the users in terms of user decision support request formulation. These criteria are outlined in Section 6.

2. PESCaDO users needs with respect to decision support

In order to allow for the description of the types of decision support the users wish to obtain from the PESCaDO system, we first introduce an overview of the initial PESCaDO pilot use cases (which will be fully documented in PESCaDO Deliverable D8.5)

2.1. Description of the use cases

The feasibility and usefulness of the technologies developed within PESCaDO will be demonstrated in the context of two pilot use cases. The initial outline of these use cases is sketched in what follows.

Pilot Use Case 1. The range of users addressed in the first pilot use case covers health-conscious citizens with no professional background on environmental services, expected to be engaged in some kind of outdoor activity or participate in traffic.

In the first scenario of Pilot Use Case 1, a user wants to plan a journey. As suggested by nowadays frequent situations in which due to poor air quality episodes traffic is restricted in city centers, the scenario requires the consideration of such environmental issues as weather, air quality, and traffic. A further issue to be considered in affected areas is the concentration of pollens.

The first implementation of this scenario is supposed to be elementary, giving answers to questions such as, e.g. "*Are there any health issues if I travel tomorrow from X to Y?*". Furthermore, in this first implementation the number of user types will be very restricted. The follow-up implementations will then consider additional user categories, activities and types of requests. Moreover, the users will be able to receive from the PESCaDO system alerts of issues they have prioritized, and obtain suggestions of appropriate actions to take.

Pilot Use Case 2. The range of users addressed in the second use case covers experts and administrative clients of the services offered by HSY for the Helsinki Metropolitan Area. It concentrates on administrative decision support (e.g. deciding whether to restrict traffic in a particular area of the city due to air quality issues) by combining information obtained from different fields and resources: environmental (AQ and weather) observed and forecasted data, administrative action plans, the information of other administrative personnel, and so on. The users will also have the possibility to receive warning notifications (and suggestions of appropriate actions to take) if critical thresholds regarding environmental related aspects (e.g. weather, air quality, pollens and traffic) are exceeded.

2.2. User needs with respect to decision support

The main goal of the PESCaDO system is to support users in taking decisions that may be influenced by environmental aspects. In this subsection, we present a first analysis of the user needs with respect to decision support. The analysis is based on the preliminary

definition of the pilot use cases and the currently considered user typology. The user needs will be further specified, detailed and revised in the course of the project.

The needs of a user are assumed to correlate with their background, activities, common requests, etc. The user categories, list of activities and types of requests supported by PESCaDO will be predefined.[1] As already mentioned above, the first pilot use case will address the needs of citizens with no professional background in environmental issues. Their characteristics with respect to the needs of receiving environmental information can be grouped into three main classes:

- Adults with normal health conditions;
- People doing outdoor sports;
- Adults or senior citizens suffering from respiratory or heart diseases.

The requests of these users for the PESCaDO-system are health or environmentally oriented: they seek advice regarding environmental conditions that affect health – for instance, air quality, pollens, weather etc. These issues are connected to the state of the environment and activities the user plans to perform.

The second pilot use case will address experts and administrative clients of the services offered by HSY for the Helsinki Metropolitan Area (HMA). Examples of users of this second use case are municipalities and institutions responsible for road maintenance and traffic restriction or experts in an environment-sensitive domain. The decision support sought by these users concerns the most appropriate measures to take if some environmental and environmental related conditions are met. For example, in the case of episodes of exceedance of PM_{10} or NO_2 thresholds, the PESCaDO system should provide to the road maintenance department of the municipality, recommendations on the actions (if any) to be taken; for instance:

- Moisture the streets with dilute calcium chloride solution (in case of PM_{10} episodes);
- Restrict or deviate traffic (in case of NO_2 episodes).

To the environmental department of the municipality, the system should provide a recommendation to inform the citizens that threshold levels have been exceeded.

In both use cases, the user needs support in taking decisions. The user requests can be categorized in three main types: *warnings*, *reports* and *instructions*. Warnings requests are typically formulated as "*Are there any health issues …*", reports requests are of the kind "*How is the level of ...*" or "*How does the weather looks like for …*", while instructions are of the kind "*What could I do to reduce my exposure to …*". Concrete examples of requests, in particular in the context of the first use case, are:

- Do I get symptoms from pollen today?
- Is there any health issue that I should take into consideration when...?
- Is it safe for me to travel from A to B tomorrow?
- Do I get symptoms from air quality today?

[1] In the advanced implementation of the PESCaDO-system, the user will be able to create their own detailed profile, which will be used by the system to return relevant information according to the request or providing the appropriate alerts.

- How is the level of ozone today?
- How does the weather affect my day?
- What is the air quality index?

In Pilot Use Case 2, the system will work mainly in proactive modality, notifying the users when a threshold is exceeded, and giving advices on suitable measures or actions to be taken.

The users need clear answers. Therefore, the answers returned by the system should be simple and relevant to the user, in accordance with the information contained in their profile. For instance, when evaluating the levels of the concentration of PM_{10} and thus deciding whether a warning should be issued or not, the system has to take into account that a user is asthmatic, or that the user is a municipality department which must invoke a specific administrative action plan if the guidelines require to do so. The answers have to include relevant facts found in the area (e.g. concentrations of pollen in city background) and reflect the conditions that affect the decision (e.g. temperature, rain and wind, air quality).

The system should behave as a collaborative system. That is, the formulation of the answers should be informative and appealing to the users. For example, given a request such as "*How will the weather for tomorrow look like?*", an answer by the system of the kind "*As it is expected to rain, you should take a raincoat with you*" would be a more informative response than "*It will rain, approximately 21mm.*" Or: "*The concentration of PM_{10} will be very high tomorrow and it is expected to remain high for the rest of this week. Therefore, you might want to consider postponing your travel by a week*" (which draws upon the air quality forecasts of the week after) is more appropriate than "*The concentration of PM_{10} will be very high tomorrow.*"

The answers should also include reasonable background information, if this is considered appropriate.

As far as the usability of the system is concerned, the needs of the users affect mainly the user interface and the way the content is delivered via this interface. The interface should be simple to use and the results should be grasped quickly. In order to ensure the acceptance of the system by the users, communication in the language of the respective user must be a further key aspect. The languages used in PESCaDO are English, Swedish and Finnish.

3. Overview of the state of the art in PDLs

Problem Description Languages have been widely used in several implemented system to foster both human-system communication and communication between different components within a system. As an example of the latter application, see the Brick Problem Description Language [Fun00]. In this overview, we will focus on PDL specifically devised for human-system interaction.

In [Joh86], the problem description language used in PROUST is described. PROUST is a program diagnosis system that aids novice programmers to check and fix non-syntactic bugs in the code they implemented. A PDL is used in PROUST in order to translate the requirements formulated in a natural language problem description in a form processable by the system. In particular, in PROUST the problem description is written in a simple notation (containing a few special keywords), and consists of the name of the problem, together with objects and goals definitions. In Figure 1, we show an example of problem description written according to the PROUST's PDL; it describes the problem of computing the average of some integer numbers until number "99999" is encountered.

```
DefProgram Average;

DefObject ?Avg:Input;

DefGoal Sentinel-Controlled Input Sequence(?Avg:Input, 99999);
DefGoal Output(Average(?Avg:Input));
```

Figure 1 - Example of PROUST's Problem Description

PDLs are also used in network surveillance expert systems, like shown in [Bie92]. Here, the PDL follows an English-like syntax, which resembles vaguely the syntax of a programming language. The problem descriptions considered in this system involve elements like rules, actions, events, and conditions. The users modify or create problem descriptions with the help of a customization tool, and the resulting descriptions are saved in ASCII files called problem scripts. These files are then compiled into the record structures used by the expert system. In Figure 2, we show an excerpt of a problem description according to the proposed PDL.

```
Problem "AM Fail"
    applies to switch
    is triggered by alarms with
        (device=AM or device=RM or device=PM)
            and (((component_code=0410 or component_code=0400)
            and failure_code=3100)
                or (component_code=0512 and failure_code=0004
                    and action=ENG)
                or (component_code=FFFF
                    and 30FF<=failure_code<=FFFF)))
    accepts alarms with
        device=AM or device=RM or device=PM
    retrieves from last 5 minutes
        NCS alarms from same switch with severity=MAJOR
        or severity=MINOR or action=CLR
    suppresses
        PE problem "PE Fail" for same switch
```

Figure 2 - Example of Problem Description for a Network Surveillance Expert System

In [Fic87], a PDL is used in the context of a system implementing a development tool for a rule-base language called ORBS. In this system, a limited domain-specific problem description language has been implemented that should support automatic design and code generation from problem descriptions. The PDL is based on NIKL, a member of the family of KL-One knowledge representation languages. However, in this system, the user does not write problem descriptions directly in NIKL, but via a graphical editor that hides the syntax of the language.

A more recent approach is presented in the MONET EU project [Mon01a]. The aim of MONET is to demonstrate the applicability of the latest ideas of the Semantic Web to the world of mathematical software, and in particular of mathematical services. The Mathematical Problem Description Language (MPDL) is an XML-based application that allows the user to describe a mathematical problem in terms of its inputs, outputs, and pre/post-conditions. In Figure 3, we show an example of a MONET problem description corresponding to a primarily test problem.

```
<mql:mqol xmlns:mql="http://monet.nag.co.uk/monet/mql"
             xmlns:msdl="http://monet.nag.co.uk/monet/msdl"
             xmlns="http://www.openmath.org/OpenMath">

<mql:query>
    <mql:mproblem>
    <msdl:classifications>
         <msdl:directive-type>
               http://monet.nag.co.uk/monet/directive#decide
         </msdl:directive-type>
    </msdl:classifications>
    <msdl:problem  name='primality'>
    <msdl:header/>
    <msdl:body>
      <msdl:output name='P'>
       <OMOBJ><OMA><OMS cd="numbertheory1" name="isprime"/>
              <OMI>1234567891</OMI>
            </OMA>
       </OMOBJ>
      </msdl:output>
    </msdl:body>
  </msdl:problem>
</mpl:mproblem>
<mql:mqol>
```

Figure 3 – Example of MONET's Problem Description

The created descriptions are grouped into a library for future reuse. More details can be found in the MONET Deliverable D14 [Mon01b].

To the best of our knowledge, no environmental-oriented problem description languages (PDLs) are available in the literature.

4. Definition of the PESCaDO PDL

The first specification of the PESCaDO Problem Description Language is based on OWL, the state-of-the-art Semantic Web Ontology Language[2]. This choice has been made, among other reasons, to facilitate reasoning and navigation on ontologies. We recall that the decision support provided to the user upon a user problem submission will be achieved by reasoning on an ontological representation of the environmental domain.

The users of the PESCaDO system will not directly interact with the PDL when formulating a request. Rather, they will submit a user problem via the user interface implemented in the system (see Section 5). We recall that the goal of submitting a user request to the system is to obtain an answer or support for the decision related to the request. To achieve this, the problem description will be submitted to the system, and the system will return the user the most appropriate content in relation to the request.

The current structuring of a problem description has been motivated by a first analysis of the user needs in terms of decision support, according to the preliminary description of the pilot use cases (see Section 2). It addresses mainly the user requests envisaged in the context of the first use case, as this use case will be the first to be realized – although some elements relevant to the second use case have been already implemented (e.g. some user typologies). The PDL will be further extended and refined during the project.

A problem description will comprise the following main components:

- Information on the kind of activity (or activities) the user wants to perform;
- The kind of the request asked by the user regarding the activity he/she wants to perform;
- Information on the user (or users) performing the chosen activity.

The above structure of a problem description is defined in the problem ontology module of the PESCaDO KB. Furthermore, each of the components of a problem description will be described in a tailored ontology module. Below, we analyze each of the mentioned ontology modules in details.

4.1 Problem

The problem ontology module formalizes the structure of a problem description. It describes a generic class “problem”, and properties characterizing it:

- `hasProblemActivity`: relates a problem with the activity (or activities) mentioned in the user problem;
- `hasProblemRequest`: relates a problem with the kind of request mentioned in the user problem;
- `hasProblemUser`: relates a problem with the users mentioned in the user problem.

[2] http://www.w3.org/TR/owl-features/

Each problem can have multiple users or activities, but only a single request. The RDF/XML serialization of the current version of the OWL problem ontology module is provided in the Appendix.

4.2 User Activities

This component specifies the activity that the user wants to perform. Examples of activities are: travelling, doing some outdoor physical activity, attending a particular event, etc. The activities supported by the system will be formally described in an ontology module, where for each activity the relevant data and characterizing parameters will be defined. For example, a "travelling" activity will most likely be characterized by a departure and arrival location, date and time information on when the user wants to travel, maybe the travel means the user wants to travel with, etc. Similarly, an "attending an event" activity will be characterized by the location of the event, date and time information on when the event is scheduled, the expected duration of the event, etc.

In Figure 4 we show the taxonomy of user activities currently described in the user activities ontology module. We recall that in a taxonomy, a transitive "subclass of" relation holds between an element and any of its ancestors (e.g. between "Jogging" and "PhysicalOutdoorActivity"). This implies that all properties defined for any ancestor are inherited by the element. The RDF/XML serialization of the OWL user activities ontology module is provided in the Appendix.

Figure 4 - Taxonomy of user activities

In details, for any activity, a property `hasActivityDateTime` is defined to allow us to specify the date and time it will be undertaken.

`Travelling` activities will be characterized by the following properties:

- `hasActivityFromLocation`: allows the specification of the departure location of a travel;
- `hasActivityToLocation`: allows the specification of the arrival location of a travel;
- `hasActivityTransportation`: allows the specification of the travel means chosen to travel.

The remaining activities are currently characterized by the following properties:

- `hasActivityLocation`: allows the specification of the location or area where the activity is / will be undertaken;
- `hasActivityDuration`: allows the specification of (an estimate of) how long the activity will be performed.

The possible values of properties `hasActivityFromLocation`, `hasActivityToLocation`, and `hasActivityLocation` are defined in the ontology module of the PESCaDO Knowledge Base (KB) describing locations (currently the GeoNames ontology and dataset). The possible values of property `hasActivityTransportation` are defined in the ontology module of the PESCaDO KB describing travel means. The remaining properties are defined as datatype properties.

4.3 User Requests

This component specifies the kind of environmental-related requests the users can pose regarding the activity they wants to perform. Examples of user requests are: "*Will I suffer of any symptom from air quality today?*", "*How does the weather forecasts look like?*", "*Is it better if I take a raincoat with me*", etc?

We can roughly divide the kind of requests in three main types, under which all other requests will be categorized:

- `Report` requests - requests for getting information on weather conditions/air quality/pollen/traffic situation;
- `Warning` requests - requests for receiving warnings in case of specific environmental conditions are met;
- `Instruction` requests - requests for getting suggestion on decision or actions to be taken in case of specific environmental conditions are met.

The taxonomy of user requests currently described in the user requests ontology module follows the above first-level classification of requests, as shown in Figure 5. The RDF/XML serialization of the OWL user request ontology module is provided in the Appendix.

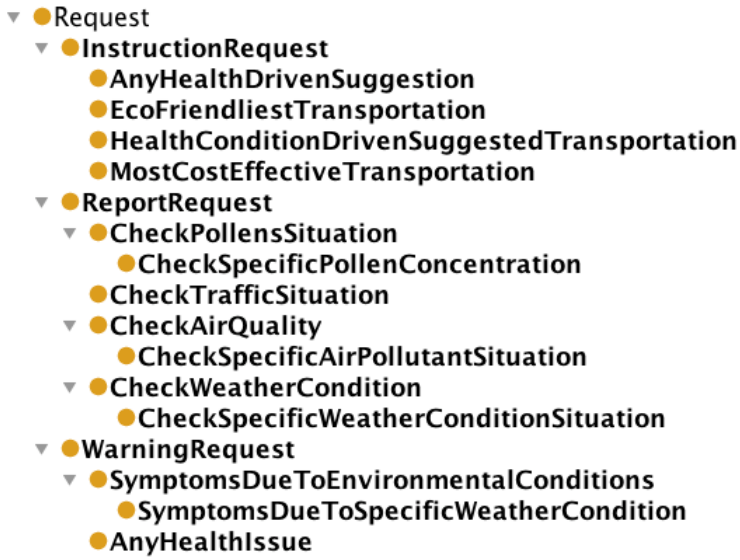

Figure 5 - Taxonomy of user requests

For those requests involving a specific environmental condition, some additional properties are defined in the ontology, in order to correctly model them. In particular, we add the following three object properties:

- `hasRequestSpecificAirPollutant`: allows the specification of the specific air pollutant to be checked for a `CheckSpecificAirPollutant` request;
- `hasRequestSpecificPollen`: allows the specification of the specific pollen to be checked for a `CheckSpecificPollen` request;
- `hasRequestSpecificWeatherCondition`: allows the specification of the specific weather condition to be checked for a `CheckSpecificWeatherCondition` request.

4.4 User Information

Information about the user(s) submitting a request is clearly relevant to the capability of providing an adequate answer. For example, in the case of a request of the kind "*Is it safe to play tennis tomorrow as far as my health is concerned?*", in order to appropriately reply to it, it is important to know if the user suffers from some diseases, if he/she is healthy, if he/she is fit, etc.

The information about the user will be stored in the user profile ontology module of the PESCaDO KB. This module will store the detailed profile of the users who are registered at the PESCaDO system, as well as some precompiled user profiles describing typical users: in the situation of seldom/occasional request to the system, these precompiled profiles could eliminate the need to create a detailed user profile.

It has to be noted that there is no need to store in the problem description the whole user profile: a reference to the specific or precompiled user profile is enough.

We briefly describe the current version of the PESCaDO user profile ontology module, which will be refined and extended in view of the results of Deliverable D8.4 - User Typology in PESCaDO. A user is characterized by the following properties:

- `hasAge`: states the age of the current user;
- `hasGender`: states the gender of the current user;
- `hasPreferredLanguage`: states the (preferred) language in which the user would like to get is answer;
- `isPregnant`: states whether a female user is pregnant or not;
- `hasPreferredTransportation`: states the user's preferences regarding transport means;
- `hasDefaultLocation`: states the user's default location (to be used e.g. if no location is specified in a user request);
- `suffersOf`: states whether the user suffers of any particular disease;
- `hasPhysicalExerciseLevel`: states, among a predefined range of values, the fitness level of the user with respect to physical exercise;
- `mayDoActivitiesWithUser`: states users (if any) which may participate in some activities together with the current user, and thus may be included in a same problem description;
- `hasPreferredOutputFormat`: states the preferred output format for the user (e.g. text, graphic, table).
- `hasAreaOfExpertise`: states the area of expertise of the user;

In addition, in order to categorize frequent types of users, some sub-specifications of the general user "class" have been defined. For the users considered in the first use case, the following sub-specifications are considered:

- `HealthyAdult`: a user at least 18 years old, not suffering from any disease;
- `SportyUser`: a user with a physical exercise level adequate for sport practicing;
- `UserSufferingOfRespiratoryOrHeartDiseases`: a user suffering from a respiratory or heart related disease;
- `HealthRisksExposedUser`: a user having limiting health conditions, e.g senior citizens, pregnant, infants.

For the users considered in the second use case, we foresee the followings sub-specifications:

- `AdministrativeUser`: a user working in a municipality administration, e.g. the person in charge of deciding whether to restrict city traffic due to environmental conditions;
- `AirQualityExpert`: a user expert in the air quality domain.

The RDF/XML serialization of the current OWL user profile ontology module is provided in the Appendix.

4.5 A Problem Description

A user problem will be represented in a problem description as a set of ontology individuals (members of some classes defined in the ontologies mentioned in Section 4.1 to 4.4) and some facts about them. Given a generic user problem, the procedure to build a concrete problem description is as follows:

1. An individual belonging to the class "problem" is defined in order to uniquely identify the current problem instance under formalization;
2. An individual is created for each user activity described in the user problem; the individual is set to be member of the class corresponding to the activity considered (e.g. `travelling` activity);
3. An individual is created for the user request described in the user problem; the individual is set to be member of the class corresponding to the request submitted (e.g. `anyHealthIssue` request);
4. The individual corresponding to the problem is related to the individuals describing activities, request, and users, via the `hasProblemAcitivity`, `hasProblemRequest`, and `hasProblemUser` properties;
5. The additional data (if any) provided for activities (e.g. location, date/time of a `travelling` activity) and request (e.g. the weather condition to check in a `checkSpecificWeatherCondition` request) are formalized as facts in the problem description.

Below, we show a few examples of problem description formalization.

4.6 Example A: Any health issue related to a travelling activity.

User problem: "*I want to travel by car with my son from location A to location B right now: is there any environmental related health issue for us?*"

In Figure 6 we show a graphical representation of the problem description taken from the Protégé ontology editor interface; the RDF/XML serialization of the problem description is provided in the Appendix.

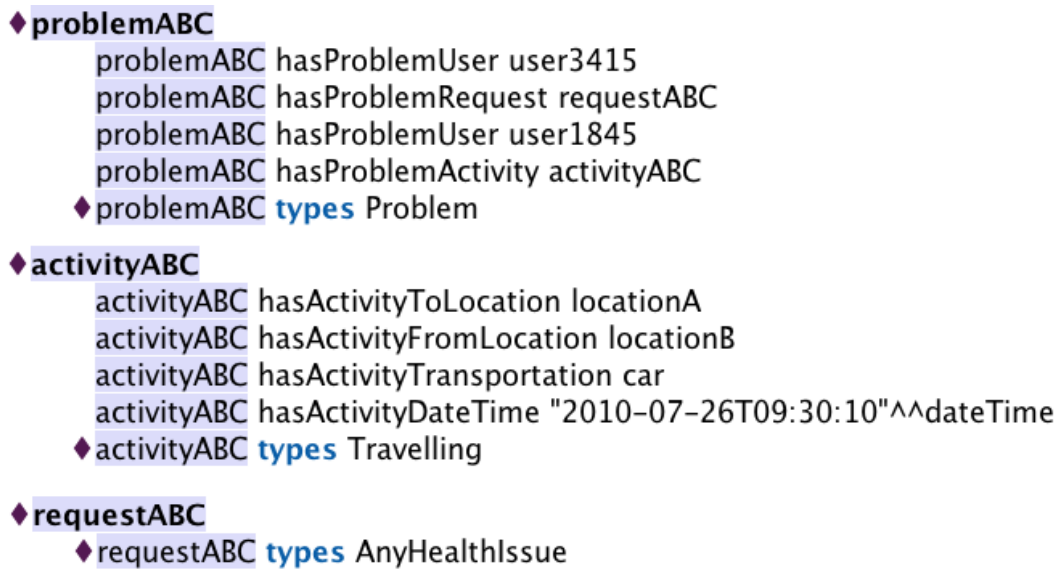

Figure 6 - Problem Description corresponding to the Example A user request

4.7 Example B: Forecasted weather conditions related to a bike tour.

User problem: "*I want to go for a mountain bike tour this Sunday in the location X. How is the weather forecast for Sunday?*"

In Figure 7 we show a graphical representation of the problem description taken from the Protégé ontology editor interface; the RDF/XML serialization of the problem description is provided in the Appendix.

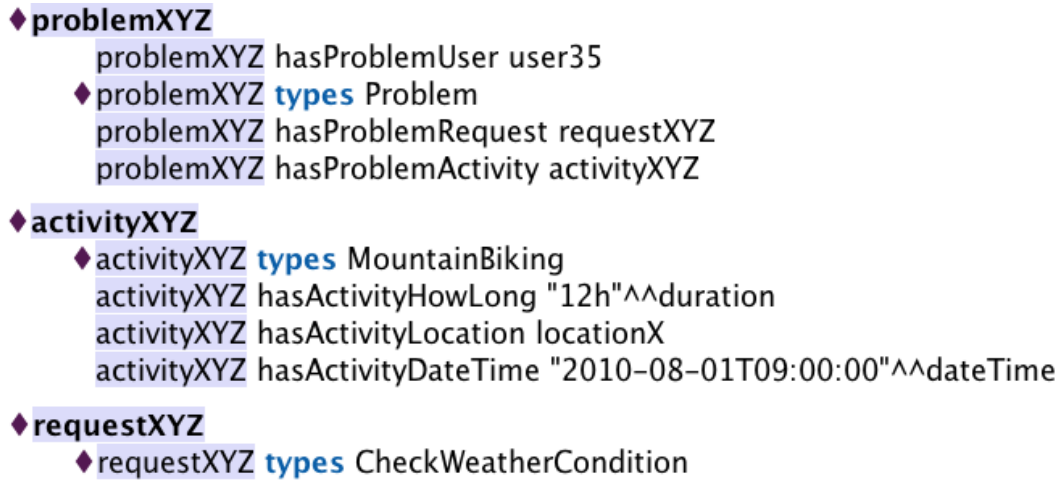

Figure 7 - Problem Description corresponding to the Example B user request

5. User – PDL Interaction

As already pointed out several times, the user will not interact directly with the PDL, i.e. write problem descriptions directly in PDL. Rather, this will be done via a user interface. This interface will "hide" the syntax and complexity of the PDL, allowing the user to express his/her needs in a graphical or natural language based way.

For stating the problem during user interaction, different levels of abstraction are possible. The appropriate level does not only depend on the skill level of the user, but also on the complexity of the problem that needs to be described. More specific requests that involve many aspects and also requests that differ greatly from the default configuration, need a lower level of abstraction. The lowest level of abstraction is the direct control of the single values of the PDL. Therefore, we suggest three options for formulating queries ordered from abstract to concrete: (controlled) natural language, graphical widgets or wizards, and direct control of the PDL structure.

Natural language is a powerful way of describing a problem in an abstract way. This can be suitable in particular for novice users due to the familiarity of natural language. For example, it is possible to state: "*I want to travel from A to B, and I have exercise-induced asthma*". The user does not need to specify that "travel" is his activity and that "A" and "B" are locations. Furthermore, he does not have to search for the concrete objects in a map or list.

The natural language mode of query formulation can be based on a limited thesaurus of inquiry types, actions, etc. defined in the PESCaDO ontologies. In the process of user's writing of the inquiry, the items of the inquiry are matched against the thesaurus. If an item does not match, it is highlighted and further supportive actions are offered. In case the system identifies the required type of input, a list with possible options for selection is offered; otherwise, the user is prompted for the selection of the type of input such that more targeted support can be provided. Underspecified queries can be made more concrete by requesting additional information from the user after sending the query to the system.

However this way of problem description depends on the availability of appropriate standard values, e.g., for the routing criteria or which environmental information is of interest. For stating many different aspects of a query, a graphical query formulation is more adequate. Here, different practices for supporting the user input are possible. A summary of the current configuration of the inquiry can be used as an overview of the query. From this summary, the user can choose to deliver input for certain aspects of the query in different forms, each specialized to display the available values of its associated aspect. For example, the location of interest for the inquiry would be selected by interacting with a map, or to select a timeframe of interest the user would brush it in a timeline display. A mapping from PDL elements to a suitable view would be supported also by an appropriate formalization of the elements in the PESCaDO ontologies. To further simplify the input, each aspect could have a preset value that the user can just confirm or adjust. For the aforementioned examples, this would be the current location or home location (if available) in case of locations, or the current date in case of the timeframe.

The more information about the request is available, the better the system can support the user in formulating it further or request additional information. For example, if the user already has chosen the activity "travel", the map module knows that at least two

locations are necessary as input; or choosing the activity biking should exclude motorways during the route calculation. This relates to the adaptation of configuration management solutions to the user interface design. If the computational complexity of the underlying algorithms allows it, even parts of intermediate responses of the system can be integrated into the user interface. For example, after the user states that he is allergic to pollen X, a visualization of the pollen concentration could be laid over the map display for instant decision support when choosing the area for an activity.

Given a defined mapping of graphical components to instances of the PDL, the graphical query formulation is less abstract. But still there can be the need of combining aspects that are not represented by graphical components. Therefore, a direct manipulation of the PDL should also be supported. Here, the only available abstraction is hiding of the OWL/XML syntax.

Navigation between different abstraction layers will be supported. Navigating from abstract to concrete does not lose information and has to be done in order to deliver the final inquiry to the reasoning system. It is also necessary to adjust details of the inquiry that might not be available at one abstraction layer. Navigation from concrete to abstract, however, might result in information loss due to missing expressiveness of the graphical or natural language formulation.

A further aspect of the possible interaction between users and the PDL concerns the support of experts in extending the ontology modules which define the PDL, and in particular, adding new activities or types of requests. Although this can, of course, be performed off-line and outside the system, using a state-of-the-art ontology editor that supports the extension within the system with an adequate user interface would definitely improve the extensibility of the tool, and its applicability to other scenarios. However, to support expert users in adding new activities or types of requests is challenging because, in general, other elements and information related to the new activity or type have to be formally defined – as, for instance, the relevant environmental aspects to be computed by the system if a request with the “new” activity is submitted. Furthermore, the system should know what to do in order to reply to a new kind of activity. For example, if the activity is travelling, the system has to know that it has to plan the computing of a route. We will investigate whether it is feasible (and to what extent we will be able) to support the extension/revision of the PDL ontologies by experts directly within the PESCaDO system.

6. Outline of the performance assessment criteria for PDL

The criteria to be used for the assessment of the PDL must cover the potential of the PDL to express the problem of the user, its usability and its intuitiveness.

In order to assess the performance and sufficiency of the PDL for the formulation of user requests in the framework of PESCaDO, qualitative and also partly quantitative evaluation will be carried out. The PDL performance assessment procedure will consists of the following three main steps:

1. *Preparation of the assessment*: Compilation of a representative set of user problems (covering all scenarios that can be addressed in PESCaDO). Care must be taken that not only simple problems whose answers match facts in the PESCaDO Knowledge Base one-to-one are captured, but also more complex problems.
2. *Experiment*:
    a. Subjects (laymen not familiar with PESCaDO) are given a manual of the PDL – describing the main components of a problem description (activities, request, users) and their properties – and are asked to reformulate all problems collected in step 1 in terms of the components of the PDL. The subjects will not write the problem directly in OWL, but use the corresponding visual component of the PESCaDO platform – if this component is already operational – or an equivalent natural language based format (e.g. “Problem XYZ involves two activities and one user. The first activity is a `jogging` activity to be performed at […]. The request associated to the problem is an `AnyHealthIssue` request.”)
    b. After the evaluation of the correctness of the obtained PDL statements (see step 3) and their potential correction, PESCaDO personnel translate the statements into OWL – in case the corresponding visual component is not operational – and retrieve the corresponding facts from the KB.
3. *Evaluation*: The evaluation consists of several parts, involving both quantitative analyses and qualitative questionnaire:
    a. Satisfaction of the subjects with the PDL: feedback from the subjects on the intuitiveness and comprehensiveness of the PDL (via qualitative questionnaire).
    b. Intuitiveness and usability of the PDL: analyses of the correctness of PDL statements including analyses of the frequency and type of errors made by the subjects (including statistics on complete failure to produce a PDL-equivalent for a given problem (see item d below).
    c. Complexity of the PDL: how complex is the “wording” of the problems in PDL (measuring how many statements it takes to express a problem as a function of problem complexity), and how much time-consuming is to formulate a problem with the PDL (measuring how much time it takes to a user to formulate a problem with the PDL, depending on the complexity of the problem to be described).
    d. Coverage of PDL: synthesis of layman and expert ability to formulate given problems in PDL (via quantitative analyses).

The subjects will be recruited from different communities accessible by the different partners of the Consortium. A particularly relevant community is constituted by the citizens of the Helsinki Metropolitan Area.

References

[Fun00] EvoCAD: EVOLUTION-ASSISTED DESIGN - Pablo Funes, Louis Lapat and Jordan B. Pollack (2000)[3]

[Joh86] Intention-Based Diagnosis of Novice Programming Errors - Lewis Johnson (1986) Morgan Kaufman Publisher, ISBN 978-0934613194

[Bie92] Customization of network surveillance expert systems - Andrzej Bieszczad and Tony White (1992)[4]

[Fic87] Development tools for rule-based systems - Stephen Fickas (in "Expert systems: the user interface" (1987)[5])

[Mon01a] IST-2001-34145, see MONET homepage[6] for more details.

[Mon01b] MONET Deliverable D14: Mathematical Service Description Language: Final Version[7]

[3] http://www.demo.cs.brandeis.edu/pr/buildable/evocad/aid00/
[4] http://ieeexplore.ieee.org/stamp/stamp.jsp?tp=&arnumber=276592&userType=inst
[5] http://portal.acm.org/citation.cfm?id=40529
[6] http://monet.nag.co.uk/monet/
[7] http://monet.nag.co.uk/monet/publicdocs/monet-msdl-final.pdf

APPENDIX A: PESCaDO PDL Related Ontology Modules

Problem

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY problem "http://www.pescado.org/ontologies/problem.owl#" >
   <!ENTITY userProfile "http://www.pescado.org/ontologies/userProfile.owl#" >
   <!ENTITY userActivities "http://www.pescado.org/ontologies/userActivities.owl#" >
   <!ENTITY userRequests "http://www.pescado.org/ontologies/userRequests.owl#" >
]>

<rdf:RDF xmlns="http://www.pescado.org/ontologies/problem.owl#"
    xml:base="http://www.pescado.org/ontologies/problem.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:userActivities="http://www.pescado.org/ontologies/userActivities.owl#"
    xmlns:problem="http://www.pescado.org/ontologies/problem.owl#"
    xmlns:userProfile="http://www.pescado.org/ontologies/userProfile.owl#"
    xmlns:userRequests="http://www.pescado.org/ontologies/userRequests.owl#">

   <owl:Ontology rdf:about="http://www.pescado.org/ontologies/problem.owl"/>

   <owl:ObjectProperty rdf:about="#hasProblemActivity">
      <rdfs:range rdf:resource="&userActivities;Activity"/>
      <rdfs:domain rdf:resource="#Problem"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="#hasProblemRequest">
      <rdfs:domain rdf:resource="#Problem"/>
      <rdfs:range rdf:resource="&userRequests;Request"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="#hasProblemUser">
      <rdfs:domain rdf:resource="#Problem"/>
      <rdfs:range rdf:resource="&userProfile;User"/>
   </owl:ObjectProperty>

   <owl:Class rdf:about="#Problem">
      <owl:equivalentClass>
         <owl:Restriction>
            <owl:onProperty rdf:resource="#hasProblemUser"/>
            <owl:someValuesFrom rdf:resource="&userProfile;User"/>
         </owl:Restriction>
      </owl:equivalentClass>
      <owl:equivalentClass>
```

```
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasProblemActivity"/>
          <owl:someValuesFrom rdf:resource="&userActivities;Activity"/>
        </owl:Restriction>
      </owl:equivalentClass>
      <owl:equivalentClass>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasProblemRequest"/>
          <owl:onClass rdf:resource="&userRequests;Request"/>
          <owl:qualifiedCardinality
rdf:datatype="&xsd;nonNegativeInteger">1</owl:qualifiedCardinality>
        </owl:Restriction>
      </owl:equivalentClass>
    </owl:Class>

</rdf:RDF>
```

User Activities

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY location "http://www.pescado.org/ontologies/location.owl#" >
   <!ENTITY transportation "http://www.pescado.org/ontologies/transportation.owl#" >
   <!ENTITY userActivities "http://www.pescado.org/ontologies/userActivities.owl#" >
]>

<rdf:RDF xmlns="http://www.pescado.org/ontologies/userActivities.owl#"
     xml:base="http://www.pescado.org/ontologies/userActivities.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:location="http://www.pescado.org/ontologies/location.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
     xmlns:userActivities="http://www.pescado.org/ontologies/userActivities.owl#"
     xmlns:transportation="http://www.pescado.org/ontologies/transportation.owl#">
    <owl:Ontology rdf:about="http://www.pescado.org/ontologies/userActivities.owl"/>

    <owl:ObjectProperty rdf:about="#hasActivityFromLocation">
        <rdfs:range rdf:resource="&location;Location"/>
        <rdfs:domain rdf:resource="#Travelling"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="#hasActivityLocation">
        <rdfs:range rdf:resource="&location;Location"/>
        <rdfs:domain>
            <owl:Class>
                <owl:unionOf rdf:parseType="Collection">
                    <rdf:Description rdf:about="#AttendingOpenAirEvent"/>
                    <rdf:Description rdf:about="#None"/>
                    <rdf:Description rdf:about="#PhysicalOutdoorActivity"/>
                </owl:unionOf>
            </owl:Class>
        </rdfs:domain>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="#hasActivityToLocation">
        <rdfs:range rdf:resource="&location;Location"/>
        <rdfs:domain rdf:resource="#Travelling"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="#hasActivityTransportation">
        <rdfs:range rdf:resource="&transportation;Transportation"/>
        <rdfs:domain rdf:resource="#Travelling"/>
    </owl:ObjectProperty>
```

```
<owl:DatatypeProperty rdf:about="#hasActivityDateTime">
   <rdfs:domain rdf:resource="#Activity"/>
   <rdfs:range rdf:resource="&xsd;dateTime"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#hasActivityHowLong">
   <rdfs:range rdf:resource="&xsd;duration"/>
   <rdfs:domain>
      <owl:Class>
         <owl:unionOf rdf:parseType="Collection">
            <rdf:Description rdf:about="#AttendingOpenAirEvent"/>
            <rdf:Description rdf:about="#None"/>
            <rdf:Description rdf:about="#PhysicalOutdoorActivity"/>
         </owl:unionOf>
      </owl:Class>
   </rdfs:domain>
</owl:DatatypeProperty>

<owl:Class rdf:about="#AttendingOpenAirConcert">
   <rdfs:subClassOf rdf:resource="#AttendingOpenAirEvent"/>
</owl:Class>

<owl:Class rdf:about="#AttendingOpenAirEvent">
   <rdfs:subClassOf rdf:resource="#Activity"/>
</owl:Class>

<owl:Class rdf:about="#AttendingOpenAirPublicDemonstration">
   <rdfs:subClassOf rdf:resource="#AttendingOpenAirEvent"/>
</owl:Class>

<owl:Class rdf:about="#AttendingOpenAirSportCompetition">
   <rdfs:subClassOf rdf:resource="#AttendingOpenAirEvent"/>
</owl:Class>

<owl:Class rdf:about="#BusinessTravelling">
   <rdfs:subClassOf rdf:resource="#Travelling"/>
</owl:Class>

<owl:Class rdf:about="#CityBicycling">
   <rdfs:subClassOf rdf:resource="#ModeratePhysicalOutdoorActivity"/>
</owl:Class>

<owl:Class rdf:about="#Hiking">
   <rdfs:subClassOf rdf:resource="#IntensivePhysicalOutdoorActivity"/>
</owl:Class>

<owl:Class rdf:about="#IntensivePhysicalOutdoorActivity">
   <rdfs:subClassOf rdf:resource="#PhysicalOutdoorActivity"/>
</owl:Class>

<owl:Class rdf:about="#Jogging">
   <rdfs:subClassOf rdf:resource="#IntensivePhysicalOutdoorActivity"/>
</owl:Class>

<owl:Class rdf:about="#LeisureTravelling">
```

```
    <rdfs:subClassOf rdf:resource="#Travelling"/>
  </owl:Class>

  <owl:Class rdf:about="#ModeratePhysicalOutdoorActivity">
    <rdfs:subClassOf rdf:resource="#PhysicalOutdoorActivity"/>
  </owl:Class>

  <owl:Class rdf:about="#MountainBiking">
    <rdfs:subClassOf rdf:resource="#IntensivePhysicalOutdoorActivity"/>
  </owl:Class>

  <owl:Class rdf:about="#None">
    <rdfs:subClassOf rdf:resource="#Activity"/>
  </owl:Class>

  <owl:Class rdf:about="#NordicWalking">
    <rdfs:subClassOf rdf:resource="#IntensivePhysicalOutdoorActivity"/>
  </owl:Class>

  <owl:Class rdf:about="#PhysicalOutdoorActivity">
    <rdfs:subClassOf rdf:resource="#Activity"/>
  </owl:Class>

  <owl:Class rdf:about="#PlayingTennis">
    <rdfs:subClassOf rdf:resource="#IntensivePhysicalOutdoorActivity"/>
  </owl:Class>

  <owl:Class rdf:about="#Skiing">
    <rdfs:subClassOf rdf:resource="#IntensivePhysicalOutdoorActivity"/>
  </owl:Class>

  <owl:Class rdf:about="#Travelling">
    <rdfs:subClassOf rdf:resource="#Activity"/>
  </owl:Class>

  <owl:Class rdf:about="#Walking">
    <rdfs:subClassOf rdf:resource="#ModeratePhysicalOutdoorActivity"/>
  </owl:Class>

</rdf:RDF>
```

User Requests

```
<?xml version="1.0"?>
<!DOCTYPE rdf:RDF [
   <!ENTITY owl "http://www.w3.org/2002/07/owl#" >
   <!ENTITY xsd "http://www.w3.org/2001/XMLSchema#" >
   <!ENTITY owl2xml "http://www.w3.org/2006/12/owl2-xml#" >
   <!ENTITY rdfs "http://www.w3.org/2000/01/rdf-schema#" >
   <!ENTITY rdf "http://www.w3.org/1999/02/22-rdf-syntax-ns#" >
   <!ENTITY userRequests "http://www.pescado.org/ontologies/userRequests.owl#" >
   <!ENTITY sweetOntologyMod
"http://www.pescado.org/ontologies/sweetOntologyMod.owl#" >
]>

<rdf:RDF xmlns="http://www.pescado.org/ontologies/userRequests.owl#"
     xml:base="http://www.pescado.org/ontologies/userRequests.owl"
     xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
     xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
     xmlns:userRequests="http://www.pescado.org/ontologies/userRequests.owl#"
     xmlns:sweetOntologyMod="http://www.pescado.org/ontologies/sweetOntologyMod.owl#"
     xmlns:owl="http://www.w3.org/2002/07/owl#"
     xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
     xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#">
    <owl:Ontology rdf:about=""/>

    <owl:ObjectProperty rdf:about="#hasRequestSpecificAirPollutant">
       <rdfs:domain rdf:resource="#CheckSpecificAirPollutantSituation"/>
       <rdfs:range rdf:resource="&sweetOntologyMod;AirPollutant"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="#hasRequestSpecificPollen">
       <rdfs:domain rdf:resource="#CheckSpecificPollenConcentration"/>
       <rdfs:range rdf:resource="&sweetOntologyMod;Pollen"/>
    </owl:ObjectProperty>

    <owl:ObjectProperty rdf:about="#hasRequestSpecificWeatherCondition">
       <rdfs:range rdf:resource="&sweetOntologyMod;WeatherCondition"/>
       <rdfs:domain>
          <owl:Class>
             <owl:unionOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#CheckSpecificWeatherConditionSituation"/>
                <rdf:Description rdf:about="#SymptomsDueToSpecificWeatherCondition"/>
             </owl:unionOf>
          </owl:Class>
       </rdfs:domain>
    </owl:ObjectProperty>

    <owl:Class rdf:about="#AnyHealthDrivenSuggestion">
       <rdfs:subClassOf rdf:resource="#InstructionRequest"/>
    </owl:Class>

    <owl:Class rdf:about="#AnyHealthIssue">
       <rdfs:subClassOf rdf:resource="#WarningRequest"/>
    </owl:Class>
```

```
<owl:Class rdf:about="#CheckAirQuality">
   <rdfs:subClassOf rdf:resource="#ReportRequest"/>
</owl:Class>

<owl:Class rdf:about="#CheckPollensSituation">
   <rdfs:subClassOf rdf:resource="#ReportRequest"/>
</owl:Class>

<owl:Class rdf:about="#CheckSpecificAirPollutantSituation">
   <rdfs:subClassOf rdf:resource="#CheckAirQuality"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#hasRequestSpecificAirPollutant"/>
         <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#CheckSpecificPollenConcentration">
   <rdfs:subClassOf rdf:resource="#CheckPollensSituation"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#hasRequestSpecificPollen"/>
         <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#CheckSpecificWeatherConditionSituation">
   <rdfs:subClassOf rdf:resource="#CheckWeatherCondition"/>
   <rdfs:subClassOf>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#hasRequestSpecificWeatherCondition"/>
         <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
      </owl:Restriction>
   </rdfs:subClassOf>
</owl:Class>

<owl:Class rdf:about="#CheckTrafficSituation">
   <rdfs:subClassOf rdf:resource="#ReportRequest"/>
</owl:Class>

<owl:Class rdf:about="#CheckWeatherCondition">
   <rdfs:subClassOf rdf:resource="#ReportRequest"/>
</owl:Class>

<owl:Class rdf:about="#EcoFriendliestTransportation">
   <rdfs:subClassOf rdf:resource="#InstructionRequest"/>
</owl:Class>

<owl:Class rdf:about="#HealthConditionDrivenSuggestedTransportation">
   <rdfs:subClassOf rdf:resource="#InstructionRequest"/>
</owl:Class>
```

```
    <owl:Class rdf:about="#InstructionRequest">
      <rdfs:subClassOf rdf:resource="#Request"/>
    </owl:Class>

    <owl:Class rdf:about="#MostCostEffectiveTransportation">
      <rdfs:subClassOf rdf:resource="#InstructionRequest"/>
    </owl:Class>

    <owl:Class rdf:about="#ReportRequest">
      <rdfs:subClassOf rdf:resource="#Request"/>
    </owl:Class>

    <owl:Class rdf:about="#Request"/>

    <owl:Class rdf:about="#SymptomsDueToEnvironmentalConditions">
      <rdfs:subClassOf rdf:resource="#WarningRequest"/>
    </owl:Class>

    <owl:Class rdf:about="#SymptomsDueToSpecificWeatherCondition">
      <rdfs:subClassOf rdf:resource="#SymptomsDueToEnvironmentalConditions"/>
      <rdfs:subClassOf>
        <owl:Restriction>
          <owl:onProperty rdf:resource="#hasRequestSpecificWeatherCondition"/>
          <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">1</owl:cardinality>
        </owl:Restriction>
      </rdfs:subClassOf>
    </owl:Class>

    <owl:Class rdf:about="#WarningRequest">
      <rdfs:subClassOf rdf:resource="#Request"/>
    </owl:Class>

</rdf:RDF>
```

User Profile

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.pescado.org/ontologies/userProfile.owl#"
    xml:base="http://www.pescado.org/ontologies/userProfile.owl"
    xmlns:userProfile="http://www.pescado.org/ontologies/userProfile.owl#"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
    xmlns:location="http://www.pescado.org/ontologies/location.owl#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:transportation="http://www.pescado.org/ontologies/transportation.owl#"
    xmlns:diseases="http://www.pescado.org/ontologies/diseases.owl#">
   <owl:Ontology rdf:about=""/>

  <owl:ObjectProperty rdf:about="#hasAreaOfExpertise">
      <rdfs:range rdf:resource="#AreaOfExpertise"/>
      <rdfs:domain rdf:resource="#User"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="#hasDefaultLocation">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:range rdf:resource="&location;Location"/>
      <rdfs:domain rdf:resource="#User"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="#hasGender">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:range rdf:resource="#Gender"/>
      <rdfs:domain rdf:resource="#User"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="#hasPhysicalExerciseLevel">
      <rdf:type rdf:resource="&owl;FunctionalProperty"/>
      <rdfs:range rdf:resource="#PhysicalExerciseLevel"/>
      <rdfs:domain rdf:resource="#User"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="#hasPreferedTransportation">
      <rdfs:range rdf:resource="&transportation;Transportation"/>
      <rdfs:domain rdf:resource="#User"/>
   </owl:ObjectProperty>

  <owl:ObjectProperty rdf:about="#hasPreferredLanguage">
      <rdfs:range rdf:resource="#Language"/>
      <rdfs:domain rdf:resource="#User"/>
   </owl:ObjectProperty>

   <owl:ObjectProperty rdf:about="#mayDoActivitiesWithUser">
      <rdfs:domain rdf:resource="#User"/>
      <rdfs:range rdf:resource="#User"/>
   </owl:ObjectProperty>
```

```
<owl:ObjectProperty rdf:about="#suffersOf">
   <rdfs:range rdf:resource="&diseases;Disease"/>
   <rdfs:domain rdf:resource="#User"/>
</owl:ObjectProperty>

<owl:DatatypeProperty rdf:about="#hasAge">
   <rdf:type rdf:resource="&owl;FunctionalProperty"/>
   <rdfs:domain rdf:resource="#User"/>
   <rdfs:range rdf:resource="&xsd;integer"/>
</owl:DatatypeProperty>

<owl:DatatypeProperty rdf:about="#isPregnant">
   <rdfs:range rdf:resource="&xsd;boolean"/>
   <rdfs:domain>
      <owl:Class>
         <owl:intersectionOf rdf:parseType="Collection">
            <rdf:Description rdf:about="#User"/>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasGender"/>
               <owl:someValuesFrom>
                  <owl:Class>
                     <owl:oneOf rdf:parseType="Collection">
                        <rdf:Description rdf:about="#Female"/>
                     </owl:oneOf>
                  </owl:Class>
               </owl:someValuesFrom>
            </owl:Restriction>
         </owl:intersectionOf>
      </owl:Class>
   </rdfs:domain>
</owl:DatatypeProperty>

<owl:Class rdf:about="#AdditionalData"/>

<owl:Class rdf:about="#AdministrativeUser">
   <owl:equivalentClass>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#hasAreaOfExpertise"/>
         <owl:someValuesFrom>
            <owl:Class>
               <owl:oneOf rdf:parseType="Collection">
                  <rdf:Description rdf:about="#Traffic"/>
               </owl:oneOf>
            </owl:Class>
         </owl:someValuesFrom>
      </owl:Restriction>
   </owl:equivalentClass>
   <rdfs:subClassOf rdf:resource="#User"/>
</owl:Class>

<owl:Class rdf:about="#AirQualityExpert">
   <owl:equivalentClass>
      <owl:Restriction>
         <owl:onProperty rdf:resource="#hasAreaOfExpertise"/>
         <owl:someValuesFrom>
```

```
            <owl:Class>
              <owl:oneOf rdf:parseType="Collection">
                <rdf:Description rdf:about="#AirQuality"/>
              </owl:oneOf>
            </owl:Class>
          </owl:someValuesFrom>
        </owl:Restriction>
      </owl:equivalentClass>
      <rdfs:subClassOf rdf:resource="#User"/>
    </owl:Class>

    <owl:Class rdf:about="#AreaOfExpertise">
      <owl:equivalentClass>
        <owl:Class>
          <owl:oneOf rdf:parseType="Collection">
            <rdf:Description rdf:about="#Traffic"/>
            <rdf:Description rdf:about="#Meteorology"/>
            <rdf:Description rdf:about="#AirQuality"/>
          </owl:oneOf>
        </owl:Class>
      </owl:equivalentClass>
      <rdfs:subClassOf rdf:resource="#AdditionalData"/>
    </owl:Class>

    <owl:Class rdf:about="#Gender">
      <owl:equivalentClass>
        <owl:Class>
          <owl:oneOf rdf:parseType="Collection">
            <rdf:Description rdf:about="#Female"/>
            <rdf:Description rdf:about="#Male"/>
          </owl:oneOf>
        </owl:Class>
      </owl:equivalentClass>
      <rdfs:subClassOf rdf:resource="#AdditionalData"/>
    </owl:Class>

    <owl:Class rdf:about="#HealthRisksExposedUser">
      <owl:equivalentClass>
        <owl:Class>
          <owl:unionOf rdf:parseType="Collection">
            <owl:Restriction>
              <owl:onProperty rdf:resource="#hasAge"/>
              <owl:someValuesFrom>
                <rdf:Description>
                  <rdf:type rdf:resource="&rdfs;Datatype"/>
                  <owl:onDatatype rdf:resource="&xsd;integer"/>
                  <owl:withRestrictions rdf:parseType="Collection">
                    <rdf:Description>
                      <xsd:minInclusive
rdf:datatype="&xsd;integer">65</xsd:minInclusive>
                    </rdf:Description>
                  </owl:withRestrictions>
                </rdf:Description>
              </owl:someValuesFrom>
            </owl:Restriction>
```

```
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasAge"/>
               <owl:someValuesFrom>
                  <rdf:Description>
                     <rdf:type rdf:resource="&rdfs;Datatype"/>
                     <owl:onDatatype rdf:resource="&xsd;integer"/>
                     <owl:withRestrictions rdf:parseType="Collection">
                        <rdf:Description>
                           <xsd:maxInclusive
rdf:datatype="&xsd;integer">2</xsd:maxInclusive>
                        </rdf:Description>
                     </owl:withRestrictions>
                  </rdf:Description>
               </owl:someValuesFrom>
            </owl:Restriction>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#isPregnant"/>
               <owl:hasValue rdf:datatype="&xsd;boolean">true</owl:hasValue>
            </owl:Restriction>
         </owl:unionOf>
      </owl:Class>
   </owl:equivalentClass>
   <rdfs:subClassOf rdf:resource="#User"/>
</owl:Class>

<owl:Class rdf:about="#HealthyAdult">
   <owl:equivalentClass>
      <owl:Class>
         <owl:intersectionOf rdf:parseType="Collection">
            <owl:Restriction>
               <owl:onProperty rdf:resource="#suffersOf"/>
               <owl:cardinality rdf:datatype="&xsd;nonNegativeInteger">0</owl:cardinality>
            </owl:Restriction>
            <owl:Restriction>
               <owl:onProperty rdf:resource="#hasAge"/>
               <owl:someValuesFrom>
                  <rdf:Description>
                     <rdf:type rdf:resource="&rdfs;Datatype"/>
                     <owl:onDatatype rdf:resource="&xsd;integer"/>
                     <owl:withRestrictions rdf:parseType="Collection">
                        <rdf:Description>
                           <xsd:minInclusive
rdf:datatype="&xsd;integer">18</xsd:minInclusive>
                        </rdf:Description>
                     </owl:withRestrictions>
                  </rdf:Description>
               </owl:someValuesFrom>
            </owl:Restriction>
         </owl:intersectionOf>
      </owl:Class>
   </owl:equivalentClass>
   <rdfs:subClassOf rdf:resource="#User"/>
</owl:Class>

<owl:Class rdf:about="#Language">
```

```
    <owl:equivalentClass>
      <owl:Class>
        <owl:oneOf rdf:parseType="Collection">
          <rdf:Description rdf:about="#Finnish"/>
          <rdf:Description rdf:about="#English"/>
          <rdf:Description rdf:about="#Swedish"/>
        </owl:oneOf>
      </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#AdditionalData"/>
  </owl:Class>

<owl:Class rdf:about="#PhysicalExerciseLevel">
    <owl:equivalentClass>
      <owl:Class>
        <owl:oneOf rdf:parseType="Collection">
          <rdf:Description rdf:about="#AmateurSportPracticing"/>
          <rdf:Description rdf:about="#ProfessionalSportPracticing"/>
          <rdf:Description rdf:about="#ModerateActivity"/>
          <rdf:Description rdf:about="#LightActivity"/>
          <rdf:Description rdf:about="#IntenseActivity"/>
          <rdf:Description rdf:about="#NoActivity"/>
        </owl:oneOf>
      </owl:Class>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#AdditionalData"/>
  </owl:Class>

  <owl:Class rdf:about="#SportyUser">
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#hasPhysicalExerciseLevel"/>
        <owl:someValuesFrom>
          <owl:Class>
            <owl:oneOf rdf:parseType="Collection">
              <rdf:Description rdf:about="#AmateurSportPracticing"/>
              <rdf:Description rdf:about="#ProfessionalSportPracticing"/>
              <rdf:Description rdf:about="#ModerateActivity"/>
              <rdf:Description rdf:about="#IntenseActivity"/>
            </owl:oneOf>
          </owl:Class>
        </owl:someValuesFrom>
      </owl:Restriction>
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#User"/>
  </owl:Class>

  <owl:Class rdf:about="#User"/>

  <owl:Class rdf:about="#UserSufferingOfRespiratoryOrHeartDiseases">
    <owl:equivalentClass>
      <owl:Restriction>
        <owl:onProperty rdf:resource="#suffersOf"/>
        <owl:someValuesFrom rdf:resource="&diseases;Disease"/>
      </owl:Restriction>
```

```
    </owl:equivalentClass>
    <rdfs:subClassOf rdf:resource="#User"/>
  </owl:Class>

  <AreaOfExpertise rdf:about="#AirQuality">
    <rdf:type rdf:resource="&owl;Thing"/>
  </AreaOfExpertise>

  <owl:Thing rdf:about="#AmateurSportPracticing">
    <rdf:type rdf:resource="#PhysicalExerciseLevel"/>
  </owl:Thing>

  <Language rdf:about="#English">
    <rdf:type rdf:resource="&owl;Thing"/>
  </Language>

  <Language rdf:about="#Finnish">
    <rdf:type rdf:resource="&owl;Thing"/>
  </Language>

  <Language rdf:about="#Swedish">
    <rdf:type rdf:resource="&owl;Thing"/>
  </Language>

  <Gender rdf:about="#Female">
    <rdf:type rdf:resource="&owl;Thing"/>
  </Gender>

  <PhysicalExerciseLevel rdf:about="#IntenseActivity">
    <rdf:type rdf:resource="&owl;Thing"/>
  </PhysicalExerciseLevel>

  <PhysicalExerciseLevel rdf:about="#LightActivity">
    <rdf:type rdf:resource="&owl;Thing"/>
  </PhysicalExerciseLevel>

  <Gender rdf:about="#Male">
    <rdf:type rdf:resource="&owl;Thing"/>
  </Gender>

  <owl:Thing rdf:about="#Meteorology">
    <rdf:type rdf:resource="#AreaOfExpertise"/>
  </owl:Thing>

  <owl:Thing rdf:about="#ModerateActivity">
    <rdf:type rdf:resource="#PhysicalExerciseLevel"/>
  </owl:Thing>

  <PhysicalExerciseLevel rdf:about="#NoActivity">
    <rdf:type rdf:resource="&owl;Thing"/>
  </PhysicalExerciseLevel>

  <owl:Thing rdf:about="#ProfessionalSportPracticing">
    <rdf:type rdf:resource="#PhysicalExerciseLevel"/>
  </owl:Thing>
```

```
    <AreaOfExpertise rdf:about="#Traffic">
        <rdf:type rdf:resource="&owl;Thing"/>
    </AreaOfExpertise>

</rdf:RDF>
```

Example A: Any health issue related to a travelling activity

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.pescado.org/ontologies/ProblemExample.owl#"
    xml:base="http://www.pescado.org/ontologies/ProblemExample.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ProblemExample="http://www.pescado.org/ontologies/ProblemExample.owl#"
    xmlns:problem="http://www.pescado.org/ontologies/problem.owl#"
    xmlns:location="http://www.pescado.org/ontologies/location.owl#"
    xmlns:userActivities="http://www.pescado.org/ontologies/userActivities.owl#"
    xmlns:transportation="http://www.pescado.org/ontologies/transportation.owl#"
    xmlns:userProfile="http://www.pescado.org/ontologies/userProfile.owl#"
    xmlns:userRequests="http://www.pescado.org/ontologies/userRequests.owl#">

  <problem:Problem rdf:about="#problemABC">
     <problem:hasProblemActivity rdf:resource="#activityABC"/>
     <problem:hasProblemRequest rdf:resource="#requestABC"/>
     <problem:hasProblemUser rdf:resource="#user1845"/>
     <problem:hasProblemUser rdf:resource="#user3415"/>
  </problem:Problem>

  <userActivities:Travelling rdf:about="#activityABC">
     <userActivities:hasActivityDateTime rdf:datatype="&xsd;dateTime"
        >2010-07-26T09:30:10</userActivities:hasActivityDateTime>
     <userActivities:hasActivityToLocation rdf:resource="#locationA"/>
     <userActivities:hasActivityFromLocation rdf:resource="#locationB"/>
     <userActivities:hasActivityTransportation rdf:resource="#car"/>
  </userActivities:Travelling>

  <userRequests:AnyHealthIssue rdf:about="#requestABC"/>

</rdf:RDF>
```

Example B: Forecasted weather conditions related to a bike tour

```
<?xml version="1.0"?>
<rdf:RDF xmlns="http://www.pescado.org/ontologies/ProblemExample.owl#"
    xml:base="http://www.pescado.org/ontologies/ProblemExample.owl"
    xmlns:rdfs="http://www.w3.org/2000/01/rdf-schema#"
    xmlns:owl2xml="http://www.w3.org/2006/12/owl2-xml#"
    xmlns:owl="http://www.w3.org/2002/07/owl#"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema#"
    xmlns:rdf="http://www.w3.org/1999/02/22-rdf-syntax-ns#"
    xmlns:ProblemExample="http://www.pescado.org/ontologies/ProblemExample.owl#"
    xmlns:problem="http://www.pescado.org/ontologies/problem.owl#"
    xmlns:location="http://www.pescado.org/ontologies/location.owl#"
    xmlns:userActivities="http://www.pescado.org/ontologies/userActivities.owl#"
    xmlns:transportation="http://www.pescado.org/ontologies/transportation.owl#"
    xmlns:userProfile="http://www.pescado.org/ontologies/userProfile.owl#"
    xmlns:userRequests="http://www.pescado.org/ontologies/userRequests.owl#">

<problem:Problem rdf:about="#problemXYZ">
      <problem:hasProblemActivity rdf:resource="#activityXYZ"/>
      <problem:hasProblemRequest rdf:resource="#requestXYZ"/>
      <problem:hasProblemUser rdf:resource="#user35"/>
   </problem:Problem>

   <userActivities:MountainBiking rdf:about="#activityXYZ">
      <userActivities:hasActivityDateTime rdf:datatype="&xsd;dateTime"
         >2010-08-01T09:00:00</userActivities:hasActivityDateTime>
      <userActivities:hasActivityLocation rdf:resource="#locationX"/>
      <userActivities:hasActivityHowLong rdf:datatype="&xsd;duration"
         >12h</userActivities: hasActivityHowLong >
   </userActivities:MountainBiking>

   <userRequests:CheckWeatherCondition rdf:about="#requestXYZ"/>

</rdf:RDF>
```