Proposal / Contract no.: 248488
Project start: February 1, 2010
Project end: January 31, 2013

UncertWeb

The *Uncert*ainty Enabled Model *Web*

SEVENTH FRAMEWORK PROGRAMME

THEME FP7-ICT-2009-4

ICT for Environmental Services and Climate Change Adaptation

Deliverable 8.2

Report on simple and publicly accessible UncertWeb examples

Title of Deliverable	Report on simple and publicly accessible UncertWeb examples
Deliverable reference number	D8.2
Related WP and Tasks	WP1 (Task 1.3), WP2 (Task 2.2), WP3 (Task 3.2, 3.3)
Type of Document	Public Report
Authors	Lydia Gerharz, Christoph Stasch, Benjamin Pross, Edzer Pebesma, Richard Jones, Matthew Williams, Dan Cornford
Date	30/09/2011
Version	1.0

Project coordinator

Dr. Dan Cornford

Aston University, United Kingdom

E-mail: d.cornford@aston.ac.uk

http://www.uncertweb.org

Revision History

Version	Date	Changes	Authors
0.0	13/07/2011	Initial template	Edzer Pebesma, Benjamin Pross, Christoph Stasch, Lydia Gerharz
0.1	24/08/2011	Filled content on encoding, processing, visualisation, case study 1 description	Benjamin Pross, Christoph Stasch, Lydia Gerharz
0.2	01/09/2011	Updated visualisation and case study description	Lydia Gerharz
0.3	14/09/2011	Added FERA chain	Richard Jones, Jill Johnson
0.4	14/09/2011	Added STAS and Discovery/Chaining descriptions	Christoph Stasch
1.0	30/09	Final draft Completed text, corrected minor errors	Edzer Pebesma, Christoph Stasch, Lydia Gerharz
1.1	07/10	Final version, internal review comments included	Edzer Pebesma, Christoph Stasch, Lydia Gerharz, Benjamin Proß

Related task(s):

Task 1.3 UncertML API

Provide a translation API that supports and permits conversion between all UncertML types where this is appropriate. This will involve creating software with a clear, simple interface (an API) that will permit other developers and users to easily employ UncertML. This API will include code to convert between all UncertML types where this is possible. Conversion between most types will be appropriate, though some conversions are non-trivial, for example from a set of samples to a Gaussian mixture model. We will provide a fully working system for common uncertainty representations (moments and other widely used statistics, histograms, realisations and distributions) and release the code on Sourceforge to permit the easy extension of the API by any developers.

Active partners: AST, UOM

Task 2.2 Discovery and access methodology and services

Design the methodology and the service specifications for model resources discovery and access: discovery and access mechanisms for the uncertainty-enabled model Web to allow client side chaining, with a software tool. The information, computational and engineering aspects of Web service specifications for stochastic model discovery and access will be addressed. For the information view, the WP1 outputs will be considered. For the computational and engineering views, this task will consider the following experiences and technologies: GEOSS AIP-2 Transverse Technology WGs experiences (i.e. Metadata, Clearinghouse Catalogue, Access, Mediation WGs), IP3 for Climate Change and Biodiversity, the INSPIRE Implementing Rules on Metadata, Discovery, Download and Transformation services, the ESA HMA catalogue extensions, and the FP7 GIGAS project interoperability guidelines for discovery. Moreover, the FP7 EuroGEOSS project (in the negotiation phase) technologies will be considered. The task identifies components that can be re-used and those that must be developed. This task will contribute to ongoing GEOSS initiatives such as: AIP and the Model Web. These service specifications will be consolidated in a document on: “Service frameworks for modelling resources”.

Active partners: CNR, AST, UOM

Task 3.2 Spatio-temporal aggregation of uncertainties

This task develops and implements the tools for aggregation of data, encoded as UncertML and GML, over space and time. Aggregation over space and/or time is often needed because models produce outputs at much finer scales than is of interest to end users and decision makers. For instance, terrestrial greenhouse gas emission or groundwater quality may be predicted for small grid cells or ‘points’ over instants in time, whereas international agreements and European legislation are based on annual averages over entire countries or NUTS regions. This service can also be used in composing (uncertain) model chains where the output of one model is required at a larger spatial or temporal resolution for input to another model. We will focus on the “upscaling” issue (aggregation). However, we also consider how simple methods for “downscaling” (disaggregation) could be included in the service. Aggregation is done numerically using Monte Carlo simulation (e.g. Heuvelink and Pebesma 1999), thus allowing maximum flexibility with respect to aggregation domain (regular as well as irregular) and type of aggregation (linear as well as non-linear). The tool will take the form of an UncertWeb service.

Active partners: WU, UOM

Task 3.3 Tools for communicating and visualising uncertainties

To maximise the uptake of the technology developed in UncertWeb we will implement open source tools to enable the visualisation and communication of the uncertain output of the service chains. Communication and visualisation of uncertainty requires intuitively appealing representations. In fact, humans can deal with uncertainty very well, as demonstrated in every-day life where we must cope with obvious uncertainties such as tomorrow’s weather or the value of stock market shares. Among the many alternatives, approaches that will be implemented and tested are dynamic visualisation of possible realities in animation mode, confidence limits, and whiteness blurring. As with expert elicitation, this will be done using established, appropriate methods and quality standards. The tool developed will be a thick client which integrates seamlessly into the UncertWeb infrastructure.

Active partners: UOM, WU, AST, NILU, CNR

Task 8.2 Integration applications – testing on the uncertainty-enabled model Web

Simple space-time averaging functions as developed in W3 will be applied to complex, uncertain data, simple but complete and publicly publishable example case studies will be prepared, and prototypical software solutions will be developed for (i) encoding and converting spatio-temporal information with probability distributions, (ii) chaining simple services that pass probability distribution information, and (iii) summarising and visualising the resulting information in a simple form.

Active partners: UOM, AST

Legal Notices

Executive Summary

This report is an UncertWeb project deliverable (D8.2: Report on simple and publicly accessible UncertWeb examples) summarising the development of components and tools realising the uncertainty-enabled model web. The UncertWeb components cover the areas of information encoding, processing, discovery & access and visualisation. Use of the different products is demonstrated in two use cases.

Section 2 describes the UncertWeb component development so far. In Section 2.1 the information encoding for uncertainties and spatial (raster and vector) data is addressed. Section 2.2 covers the processing components. Here OGC based components as well as implementations based on SOAP/WSDL technology are presented. Next, Section 2.3 briefly describes the Service Discovery and Access. This topic is discussed in more detail in deliverable D2.2 (Service frameworks for modelling resources: Discovery and Chaining Services specifications). In section 2.4 an overview of the visualisation tool is given.

Section 3 contains two example case studies demonstrating the use of the UncertWeb components. The first case study includes air quality modelling on an urban scale and utilises OGC web-based geoprocessing components. Case study two is based on the WP5 scenario on modelling land use change. The prototypes are realised using SOAP/WSDL technology in contrast to case study one. The first case study is completely based upon data, models and software that are publicly available, and that can be obtained by request from the authors.

Section 4 gives a conclusion on the presented examples, the feasibility of the developed concepts and the problems occurring with the implemented case studies. The two approaches of OGC and SOAP/WSDL implementations are compared and evaluated, and future work for the upcoming project time is discussed.

Contents

1 Introduction

The European Project UncertWeb aims to realise the uncertainty enabled model web. Therefore existing tools and technologies for the Model Web need to be extended to support the processing and communication of uncertainties associated with spatio-temporal as well as non-spatial and non-temporal information, e.g. input parameters. In the UncertWeb project, standards to encode, process and visualise uncertain information are developed and tested. The integration and evaluation of these technologies takes place in WP8 and are presented as examples in this report.

This report on simple and publicly accessible UncertWeb examples provides integration applications testing the uncertainty enabled Model Web components. These components were developed in the technical work packages WP1, WP2 and WP3. Uncertainty enabled model web services will be tested using prototypical software solutions developed for (i) encoding and converting spatio-temporal information with probability distributions, (ii) chaining simple services that pass probability distribution information, and (iii) summarising and visualising the resulting information in a simple form. As test beds, two use cases are taken into account using different implementations of the components. Examples are published in the web and accessible to the public.

All software developed for this deliverable is available under an open source licence (GPL v2[1]), managed by 52°North. It can be found at
https://svn.52north.org/svn/geostatistics/main/uncertweb/

1.1 Definitions

In order to ensure that a standard set of terms is used throughout this document, the following definitions are adhered to.

Model web / Model chain: a chain of model components connected through web service interfaces.

Model component: a representation of a process or series of processes implemented as computer code, often called a simulator.

Model input: a value, or series of values (if a field), that must be provided to the model component to evaluate the model (likely to include parameters in the model component, and initial and boundary conditions for the model component).

Model output: a value, or series of values (if a field), that is produced by the model when it is evaluated.

1.2 Abbreviations

EBP	Executable Business Process
GDAL	Geographic Data Abstraction Layer
GML	Geography Markup Language
JSON	JavaScript Object Notation

[1] http://www.gnu.org/licenses/old-licenses/gpl-2.0.txt

NetCDF	Network Common Data Format
OGC	Open Geospatial Consortium
O&M	Observations & Measurements
PDF	Probability Distribution Function
SOAP	Simple Object Access Protocol
UncertML	Uncertainty Markup Language
UPS	Uncertainty Proxy Service
UTS	Uncertainty Transformation Service
WSDL	Web Service Definition Language
WMS	Web Mapping Service
WPS	Web Processing Service

2 UncertWeb components

2.1 Data encodings

In order to exchange spatio-temporal data with quantified uncertainties in the Model Web, common data models and encodings for uncertain spatio-temporal data are needed. These are described in this subsection. Based on the requirements from the application scenarios identified in D8.1 we introduce our models and encodings for uncertainties, spatio-temporal vector data, spatio-temporal raster data, and how to add uncertainty to the data.

2.1.1 Uncertainy Markup Language API

The Uncertainty Markup Language (UncertML) is used in the UncertWeb project to encode uncertainties. After UncertML 1.0 [Williams 2009] was defined by an XML schema in its first version within the INTAMAP project [Pebesma 2011], UncertML has been updated in the UncertWeb project to version 2.0 by developing a conceptual model for uncertainties and defining additional XML and JSON encodings. For the XML encoding, the UncertML XML schema has been simplified by removing dependencies to other XML schema (e.g. OGC SweCommon) and by changing it from a soft-typed to a hard-typed schema. Furthermore, UncertML has been extended in the UncertWeb project to support additional types of uncertainties as defined in the requirements in UncertWeb Deliverable 8.1. Figure 1 shows the basic model of uncertainties as defined in UncertML 2.0. All uncertainties are subtypes of the `AbstractUncertainty` type defined in the main UncertML packages. The subtypes are then grouped into three major packages: `Samples` contains samples which can be ensembles or realisations. `SummaryStatistics` contains statistical measures for a dataset such as `Mean` or `Median`. The third major group defines different `Distributions` such as the `Normal` or the `StudentT` distribution.

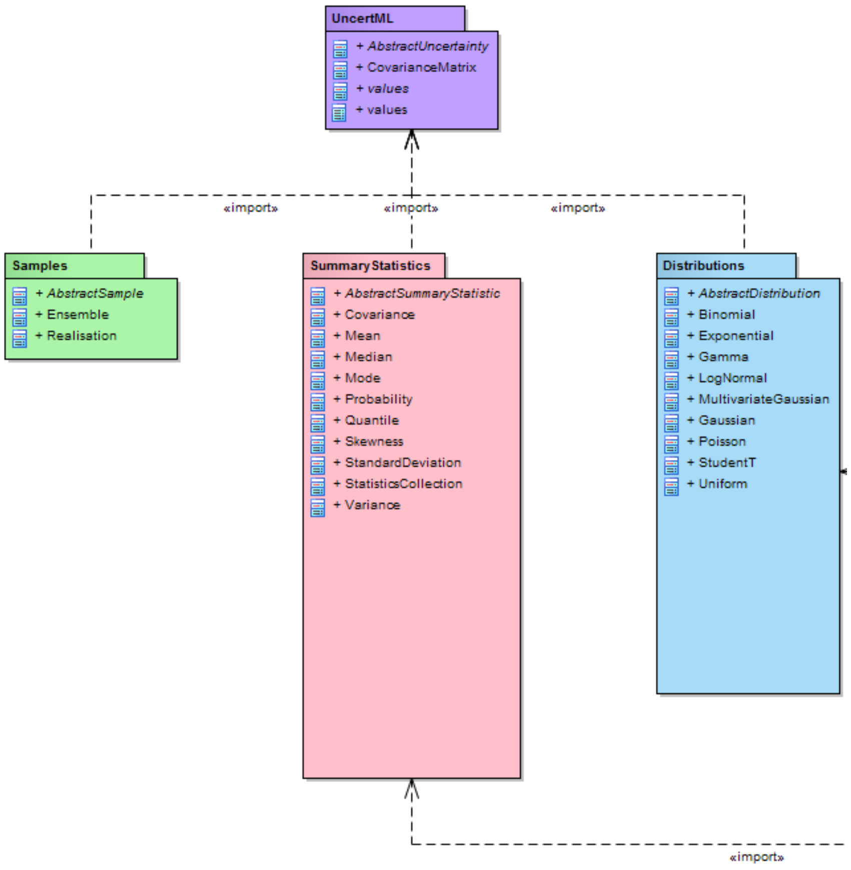
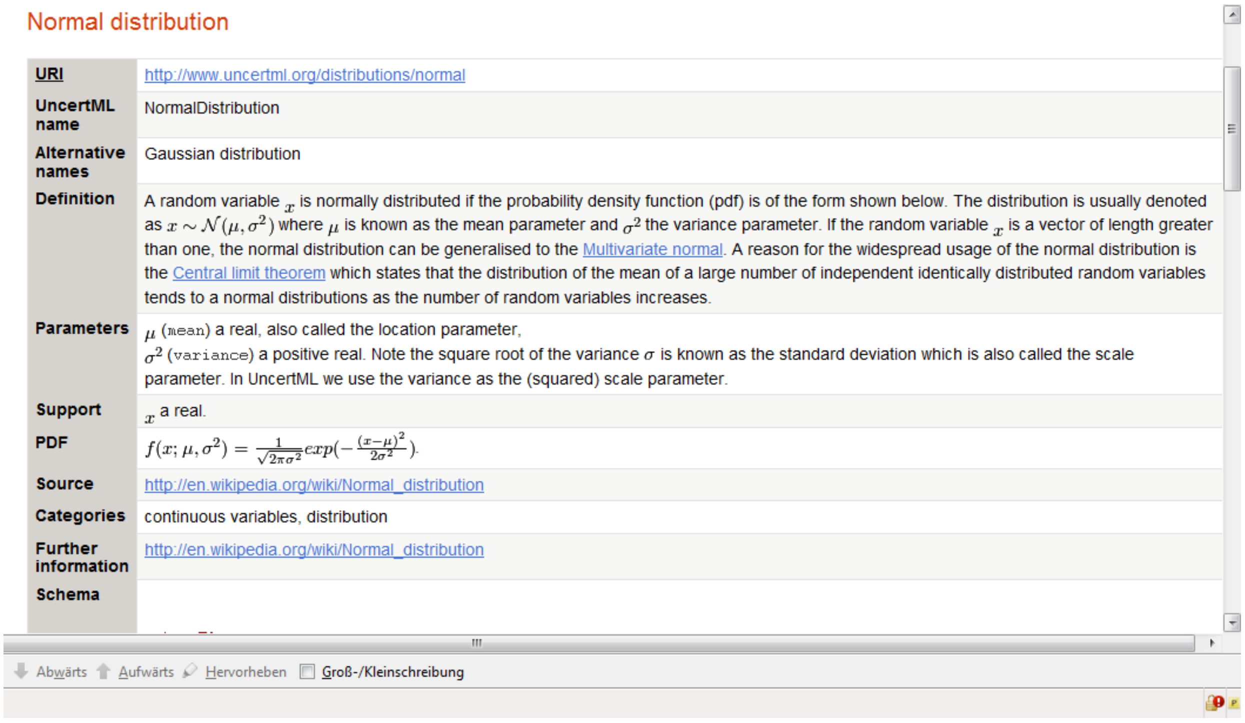


Figure 1. Basic model of UncertML 2.0.

In order to add uncertainties to other encodings and to help with understanding of the concepts provided in UncertML, an UncertML dictionary has been developed that allows the user to identify concepts defined in UncertML by a URL and to retrieve a description of the concept when resolving the URL. Figure 2 shows a screenshot of the UncertML dictionary entry for the concept of `NormalDistribution`. It contains the URI definition as well as the UncertML name, alternative names, a textual definition and the parameters of the distribution. Furthermore, the mathematical formulae are provided along with links to additional descriptions (e.g. Wikipedia).



Normal distribution

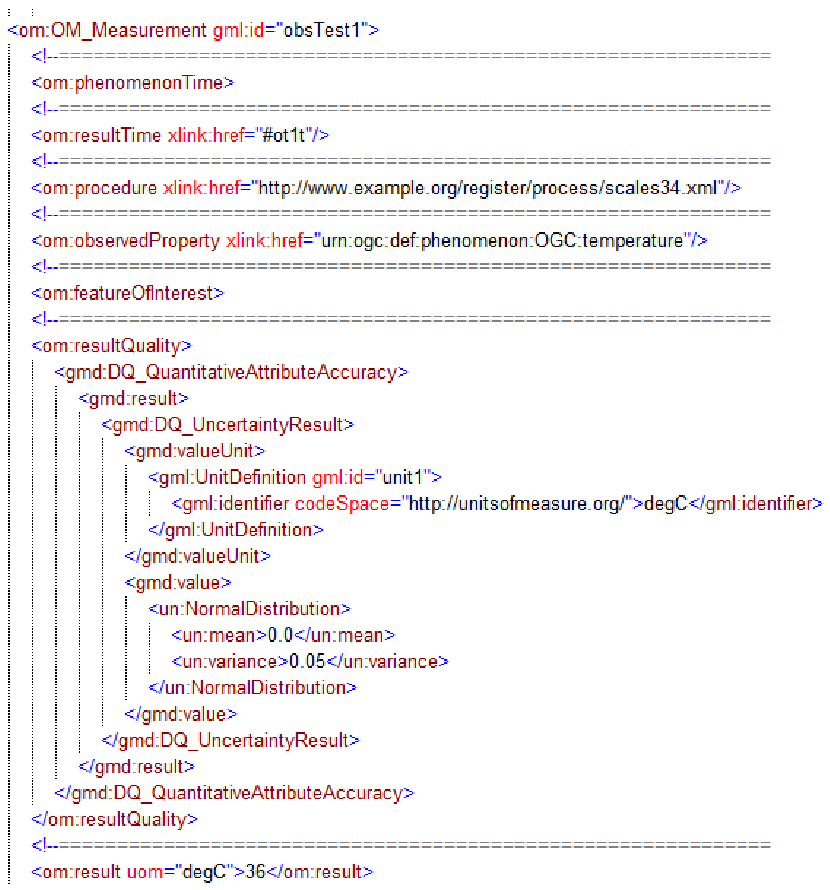
URI	http://www.uncertml.org/distributions/normal
UncertML name	NormalDistribution
Alternative names	Gaussian distribution
Definition	A random variable x is normally distributed if the probability density function (pdf) is of the form shown below. The distribution is usually denoted as $x \sim \mathcal{N}(\mu, \sigma^2)$ where μ is known as the mean parameter and σ^2 the variance parameter. If the random variable x is a vector of length greater than one, the normal distribution can be generalised to the Multivariate normal. A reason for the widespread usage of the normal distribution is the Central limit theorem which states that the distribution of the mean of a large number of independent identically distributed random variables tends to a normal distributions as the number of random variables increases.
Parameters	μ (mean) a real, also called the location parameter, σ^2 (variance) a positive real. Note the square root of the variance σ is known as the standard deviation which is also called the scale parameter. In UncertML we use the variance as the (squared) scale parameter.
Support	x a real.
PDF	$f(x;\mu,\sigma^2) = \frac{1}{\sqrt{2\pi\sigma^2}} exp(-\frac{(x-\mu)^2}{2\sigma^2})$.
Source	http://en.wikipedia.org/wiki/Normal_distribution
Categories	continuous variables, distribution
Further information	http://en.wikipedia.org/wiki/Normal_distribution
Schema	

Figure 2. Screenshot of the UncertML dictionary entry for normal distribution.

2.1.2 O&M profiles

The Open Geospatial Consortium (OGC) has already defined a model and encoding for spatio-temporal vector based observations, the Observations & Measurements format [Cox 2007]. In the UncertWeb project, O&M is restricted in order to fulfill the requirements identified in the UncertWeb Deliverable 8.1 and to reduce complexity when supporting this format. The UncertWeb O&M profile consists of three different parts: First, a subset of the large number of geometries provided in the Geography Markup Language (GML) [Portele 2007] . Second, different observation subtypes for different variable types have been defined according to the O&M specification, e.g. `BooleanObservation`. Third, two different ways to add uncertainty in observations have been developed. The result value of an observation can either be an uncertainty value, e.g. the resulting value of an observation is a probability distribution, or additional uncertainty information about the value can be added in the `resultQuality` element of an observation.

Two different encodings have been developed for the UncertWeb O&M profile, an XML encoding and a JSON encoding. The XML encoding is defined by XML schema and restricts or extends the schemas provided by OGC. The JSON encoding follows the GeoJSON conventions and defines a JSON encoding for O&M observations in a similar way. For encoding of uncertainties, the definitions from the UncertML JSON encodings are used. Additionally, the encoding allows for annotating O&M observations by using XML links to point to external uncertainties. Figure 3 shows an example where the observation value in the `result` element is the temperature average over 1 min (36 ° Celcius) and the `resultQuality` element contains information about the measurement error in the form of a normal probability distribution with zero mean and a variance of 0.05 ° Celcius².



```
<om:OM_Measurement gml:id="obsTest1">
  <!--==============================================================
  <om:phenomenonTime>
  <!--==============================================================
  <om:resultTime xlink:href="#ot1t"/>
  <!--==============================================================
  <om:procedure xlink:href="http://www.example.org/register/process/scales34.xml"/>
  <!--==============================================================
  <om:observedProperty xlink:href="urn:ogc:def:phenomenon:OGC:temperature"/>
  <!--==============================================================
  <om:featureOfInterest>
  <!--==============================================================
  <om:resultQuality>
    <gmd:DQ_QuantitativeAttributeAccuracy>
      <gmd:result>
        <gmd:DQ_UncertaintyResult>
          <gmd:valueUnit>
            <gml:UnitDefinition gml:id="unit1">
              <gml:identifier codeSpace="http://unitsofmeasure.org/">degC</gml:identifier>
            </gml:UnitDefinition>
          </gmd:valueUnit>
          <gmd:value>
            <un:NormalDistribution>
              <un:mean>0.0</un:mean>
              <un:variance>0.05</un:variance>
            </un:NormalDistribution>
          </gmd:value>
        </gmd:DQ_UncertaintyResult>
      </gmd:result>
    </gmd:DQ_QuantitativeAttributeAccuracy>
  </om:resultQuality>
  <!--==============================================================
  <om:result uom="degC">36</om:result>
```

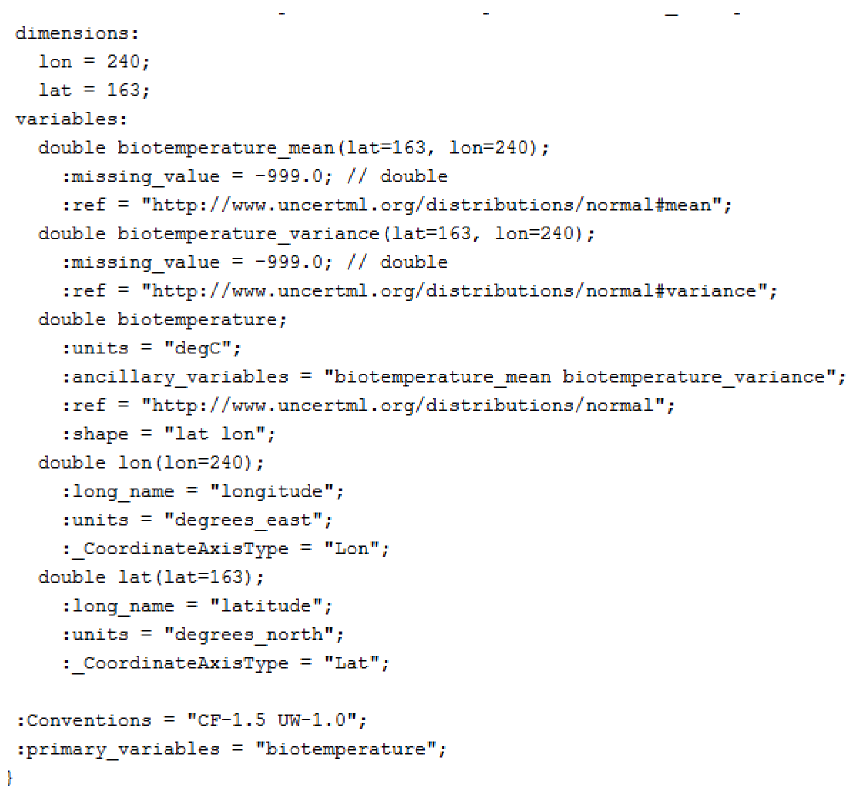
Figure 3. Measurement observation with a normal distribution for the measurement error as the uncertainty in the resultQuality element.

2.1.3 NetCDF-U extension

The Network Common Data Format (NetCDF)[2] is a well-established standard for exchanging scientific multi-dimensional datasets in the form of spatially gridded time series. Due to its binary encoding, it offers an efficient way to exchange large scientific datasets in the web. In order to integrate uncertainties in NetCDF files, UncertWeb has developed an annotation approach that uses the URLs defined in the UncertML dictionary to add variables containing uncertainty information to a NetCDF file. Figure 4 shows a header of an uncertainty-enabled NetCDF file[3] containing a normal distribution. The `biotemperature` variable is associated with additional uncertainty information indicated by the `ref` attribute whose value contains the URL for the normal distribution pointing to the UncertML dictionary. As described in the dictionary, the normal distribution contains two parameters, the `mean` and the `variance`. Therefore, two additional variables are defined (`biotemperature_mean` and `biotemperature_variance`) which are referenced in the declaration of the `biotemperature` variable by the `ancillary_variables` attribute. Both ancillary variables contain links to the UncertML dictionary again in order to identify the UncertML type that is represented by the ancillary variable.

[2] http://www.unidata.ucar.edu/software/netcdf/

[3] Example NetCDF files are available upon request.



```
dimensions:
  lon = 240;
  lat = 163;
variables:
  double biotemperature_mean(lat=163, lon=240);
    :missing_value = -999.0; // double
    :ref = "http://www.uncertml.org/distributions/normal#mean";
  double biotemperature_variance(lat=163, lon=240);
    :missing_value = -999.0; // double
    :ref = "http://www.uncertml.org/distributions/normal#variance";
  double biotemperature;
    :units = "degC";
    :ancillary_variables = "biotemperature_mean biotemperature_variance";
    :ref = "http://www.uncertml.org/distributions/normal";
    :shape = "lat lon";
  double lon(lon=240);
    :long_name = "longitude";
    :units = "degrees_east";
    :_CoordinateAxisType = "Lon";
  double lat(lat=163);
    :long_name = "latitude";
    :units = "degrees_north";
    :_CoordinateAxisType = "Lat";

:Conventions = "CF-1.5 UW-1.0";
:primary_variables = "biotemperature";
}
```

Figure 4. Exemplary header of NetCDF-U containing a normal distribution as the uncertainty.

2.1.4 UncertWeb Information Model Java API

To ease the development of components using the models and encodings as defined above, the UncertWeb project has implemented a Java API that provides parsers and encoders for the different data models and XML/JSON encodings. Figure 5 shows the four components that make up the API. The UncertML API provides Java classes that represent the different uncertainty types defined in the UncertML conceptual model as well as parsers and encoders for XML and JSON. The UncertWeb GML API provides parsers and encoders for the geometries as defined in the requirements. It uses and extends the Java Topology Suite[4] to represent and parse/encode the geometries. The GML API is used by the O&M API which provides classes and parsers/encoders for the vector-based spatio-temporal data with uncertainties. For the uncertainties, the UncertML API is used. Similar to the O&M API, the NetCDF API provides a Java API for representing and encoding/decoding uncertainty enabled NetCDF files. It relies upon the already existing NetCDF Java API and adds methods for defining and managing the uncertainty variables by using the UncertML Java API to represent the uncertainty in the variables.

The different Java APIs allow us to easily add the support for the data models and encodings used in UncertWeb to already existing implementations. For example, a module is developed that adds support for the UncertWeb data types as input or output variables to the 52°North Web Processing Service[5]. A module for the 52°North Sensor Observation Service is currently under development in order to provide web-based storage and query of uncertainty-enabled scientific observation sets in the Web. For gridded data, the NetCDF-U profile can be used by WCS servers to provide uncertainty enabled NetCDF layers.

[4] http://www.vividsolutions.com/jts/JTSHome.htm

[5] http://52north.org/communities/geoprocessing/wps/index.html

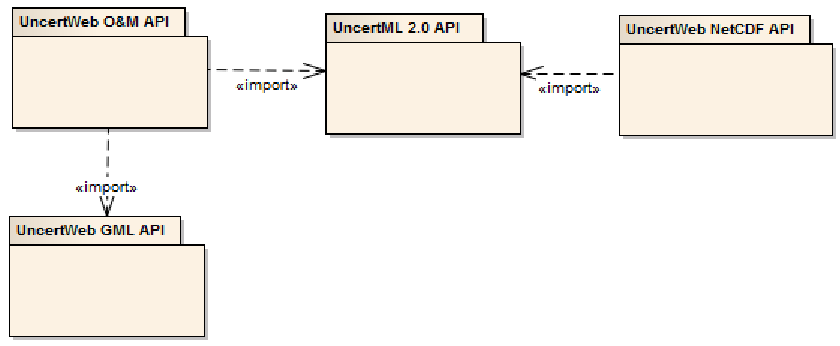


Figure 5. Components of the UncertWeb Information Model API.

2.2 Processing

Besides the data encodings described in the previous section, different service interfaces have been developed to accomplish the goal of exposing models in the web. The developed services can be divided up into those coming from the geospatial domain, those represented through the OGC WPS [Schut 2007], and domain independent services based on SOAP/WSDL. In this section, we firstly introduce the services developed within UncertWeb based on the OGC WPS specification, and then we discuss the SOAP/WSDL based services.

2.2.1 OGC WPS based service interfaces

As the WPS standard is kept generic, application profiles should be developed to ensure interoperability. The services introduced in this section were developed as application profiles of the WPS standard. Like [Baranski 2009] we do not define a specific common WPS profile, but instead define process descriptions for specific processes and restrict the supported input and output types. This should make it easier to connect to the services in a generic way which the UncertWeb architecture heavily relies on. All OGC WPS implementations shown here are based on the 52°North WPS implementation[6].

2.2.1.1 Uncertainty Transformation Service (UTS)

This service, exposed as a WPS or SOAP/WSDL interface, will allow the conversion between different UncertML types. These can be simple transformations from distributions to samples or from samples to statistics like the mean and standard deviation. In particular the UTS will also allow conversions that require further assumptions, for example going from a set of realisations to a distribution. The UTS will also create spatial and temporal realisations which extends the capabilities of the UncertML API. The UTS will use the UncertWeb Information Model API for the parsing and writing of UncertML, NetCDF-U and O&M, and it will link to R[7] or Matlab[8] for the conversion of uncertainty types.

Implementation

[6] http://52north.org/communities/geoprocessing/wps/index.html

[7] http://www.r-project.org/

[8] http://www.mathworks.de/products/matlab/

The UTS is realised as a profile of the OGC WPS based on the 52°North WPS implementation. It uses Rserve[9] to link to R where the actual conversion is performed. In Table 1 the supported abstract processes are described.

Table 1. Input and output descriptions of the example UTS process.

Process	Description	Supported input types	Supported output types
Gaussian2Samples	Creates realisations from a normal distribution.	NetCDF UncertML	NetCDF UncertML
MultiGaussian2Sample	Creates realisations from a multivariate normal distribution.	NetCDF UncertML	NetCDF UncertML
Lognormal2Sample	Creates realisations from a lognormal distribution.	NetCDF UncertML	NetCDF UncertML
Gaussian2Quantiles	Computes quantiles of a normal distribution.	NetCDF UncertML	NetCDF UncertML
Gaussian2Probabilities	Computes probabilities of a normal distribution.	NetCDF UncertML	NetCDF UncertML
Samples2Statistics	Computes statistics (e.g. mean and variance) from a given set of samples.	NetCDF UncertML	NetCDF UncertML

A demonstration instance of the UTS with a set of example requests using NetCDF-U inputs and outputs is running at [10].

In the following we will describe one of the example processes to draw samples from a normal distribution raster. Figure 6 shows a process description for the process "org.uncertweb.wps.Gaussian2Samples".

Table 2 shows a description of the input and output types.

[9] http://rosuda.org/Rserve/

[10] http://giv-uw2.uni-muenster.de:8080/uts/test.html

```xml
<?xml version="1.0" encoding="UTF-8"?>
<ns:ProcessDescriptions xmlns:ns="http://www.opengis.net/wps/1.0.0" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://schemas.opengis.net/wps/1.0.0/wpsDescribeProcess_response.xsd" xml:lang="en-US" service="WPS" version="1.0.0">
  <ProcessDescription xmlns:wps="http://www.opengis.net/wps/1.0.0" xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink" wps:processVersion="2" storeSupported="true" statusSupported="false">
    <ows:Identifier>org.uncertweb.wps.Gaussian2Samples</ows:Identifier>
    <ows:Title>Process takes a number of random samples from a Gaussian Distribution</ows:Title>
    <ows:Abstract>Computes a specified number of samples for a given gaussian distribution.</ows:Abstract>
    <wps:Profile>http://www.opengis.net/spec/WPS_application-profile_uncertainty_transformation_service/1.0</wps:Profile>
    <DataInputs>
      <Input minOccurs="1" maxOccurs="1">
        <ows:Identifier>Distribution</ows:Identifier>
        <ows:Title>Gaussian distribution</ows:Title>
        <ows:Abstract>Gaussian distribution</ows:Abstract>
        <ComplexData>
          <Default>
            <Format>
              <MimeType>application/x-netcdf</MimeType>
            </Format>
          </Default>
          <Supported>
            <Format>
              <MimeType>text/xml</MimeType>
              <Schema>http://giv-uw.uni-muenster.de:8080/uts/schemas/uncertml2.0.0/GaussianDistribution.xsd</Schema>
            </Format>
          </Supported>
        </ComplexData>
      </Input>
      <Input minOccurs="1" maxOccurs="1">
        <ows:Identifier>NumberOfRealisations</ows:Identifier>
        <ows:Title>number of realisations</ows:Title>
        <ows:Abstract>number of realisations</ows:Abstract>
        <LiteralData>
          <ows:DataType ows:reference="xs:integer"/>
          <ows:AllowedValues>
            <ows:Value/>
          </ows:AllowedValues>
          <DefaultValue>10</DefaultValue>
        </LiteralData>
      </Input>
    </DataInputs>
    <ProcessOutputs>
      <Output>
        <ows:Identifier>Samples</ows:Identifier>
        <ows:Title>samples</ows:Title>
        <ows:Abstract>samples</ows:Abstract>
        <ComplexOutput>
          <Default>
            <Format>
              <MimeType>application/x-netcdf</MimeType>
            </Format>
          </Default>
          <Supported>
            <Format>
              <MimeType>text/xml</MimeType>
              <Schema>http://giv-uw.uni-muenster.de:8080/uts/schemas/uncertml2.0.0/Realisation.xsd</Schema>
            </Format>
          </Supported>
        </ComplexOutput>
      </Output>
    </ProcessOutputs>
  </ProcessDescription>
</ns:ProcessDescriptions>
```

Figure 6. Example UTS process description for a conversion from a normal distribution to realisations.

Table 2. Input and output descriptions of the example UTS process.

Identifier	Description
Distribution	The Normal distribution encoded in NetCDF-U or UncertML
NumberOfRealisations	The desired number of realisations
Samples	The resulting samples encoded in NetCDF-U or UncertML

Figure 7 shows an execute process request for this process. The input is a NetCDF-U file containing a raster with a mean and a variance value for every raster cell.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0" xmlns:ows="
http://www.opengis.net/ows/1.1" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="
http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
 http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
  <ows:Identifier>org.uncertweb.wps.Gaussian2Samples</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier xmlns:ns1="http://www.opengis.net/ows/1.1">Distribution</ows:Identifier>
      <wps:Reference xlink:href="http://v-mars.uni-muenster.de/uncertweb/data/biotemperature_normalDistr.nc"
mimeType="application/x-netcdf"/>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>NumberOfRealisations</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>10</wps:LiteralData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:RawDataOutput mimeType="application/x-netcdf">
      <ows:Identifier>Samples</ows:Identifier>
    </wps:RawDataOutput>
  </wps:ResponseForm>
</wps:Execute>
```

Figure 7. Execute request for example UTS process.

The UTS parses the NetCDF-U file and passes the mean and variance values via Rserve to R. There the specified number of realisations is generated by drawing random samples from a normal distribution using the `rnorm`[11] command. In this example, the samples are drawn independently for each raster cell. Information about the spatial covariance structure (e.g. represented by a variogram) could be used to draw spatially dependent realisations. The resulting sample values per cell are gathered and written to a new NetCDF-U file along with the appropriate header information (see Figure 4 in Section 2.1.3).

2.2.1.2 Uncertainty Proxy Service (UPS)

The UPS serves as a proxy service for a WPS that encapsulates a model but is not able to use uncertainty information for the model execution. This is usually the case if a model is brought to the web but the model software itself is not able to use uncertainties, e.g. distributions, in its input information. To account for the error propagation from the inputs to the output the Monte Carlo simulation approach is chosen here. Thereby, a large number of samples are drawn from the input distributions and each sample is used for a single model run. The

[11] http://stat.ethz.ch/R-manual/R-patched/library/stats/html/Normal.html

outputs can be summarised again as either a distribution or as statistics. Therefore the UPS must be able to parse and generate the encodings described in Section 2.1. This task of running the Monte Carlo simulations over the underlying Model WPS can be executed either by the UncertWeb architecture framework (see Section 2.3) or the UPS, allowing for more flexibility in uncertainty enabled service chains.

Profile
The UPS offers one process named "ExecuteMonteCarloSimulation" that has a fixed set of inputs and outputs. In the following table (Table 3) the input and output types are described:

Table 3. Input and output descriptions of the "ExecuteMonteCarloSimulation" process from the UPS.

Identifier	Description
IdentifierSimulatedProcess	The identifier of the process with which the Monte Carlo simulation will be run.
UncertainProcessInputs	Inputs for the process that will change with every Monte Carlo run.
StaticProcessInputs	Inputs for the process that will not change during the Monte Carlo simulation.
ServiceURL	The url of the WPS that exposes the model process.
OutputUncertaintyType	Besides the output format we need to specify the output uncertainty type. For example realisations or normal distribution.
NumberOfRealisations	The desired number of realisations, i.e. Monte Carlo runs.
UncertainProcessOutputs	The result with uncertainty encoded in the specified format. For example UncertML or O&M.

General execution patterns
There are two general patterns for the “ExecuteMonteCarloSimulation” process: In the first one (the simple UPS workflow - see Figure 8) a set of samples serves as the uncertain process input for the process. The UPS executes the Model WPS in a loop iterating over the input samples. When finished the resulting output realisations are put together and sent back to the user.

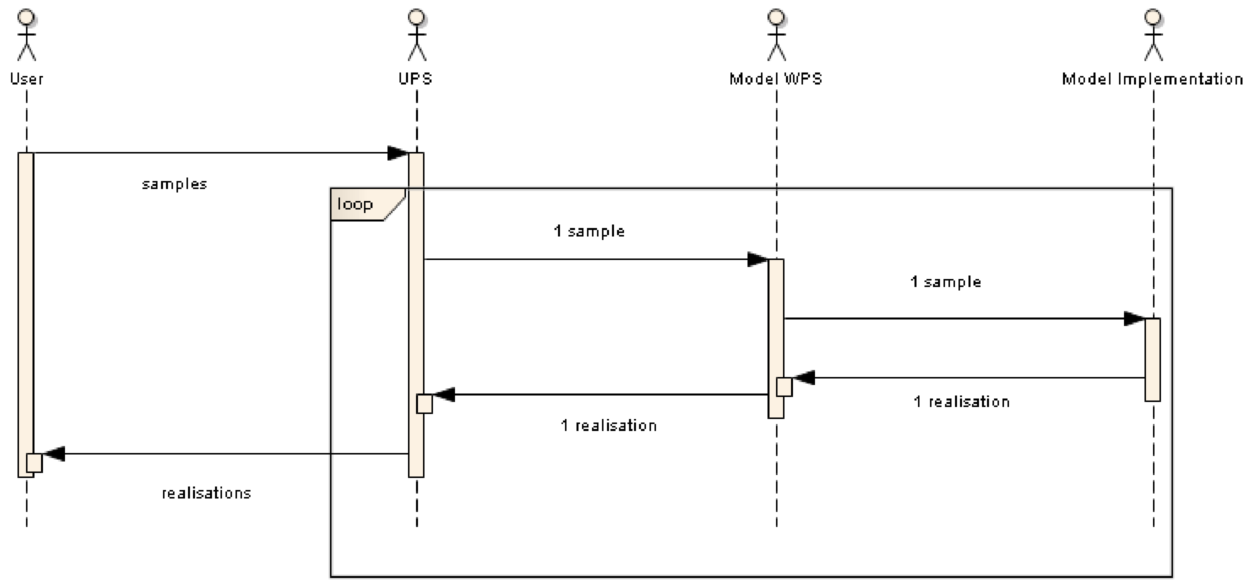


Figure 8. Simple UPS workflow.

In the second pattern (the extended UPS workflow - see Figure 9) the uncertain input is encoded as a probability distribution function (PDF). In an additional step during the process, the PDF is transformed into a set of samples using the UTS (see Section 2.2.1.1). The UPS will iterate over these samples as in the simple UPS workflow pattern. The resulting output realisations are transformed into a PDF via the UTS again and sent back to the user.

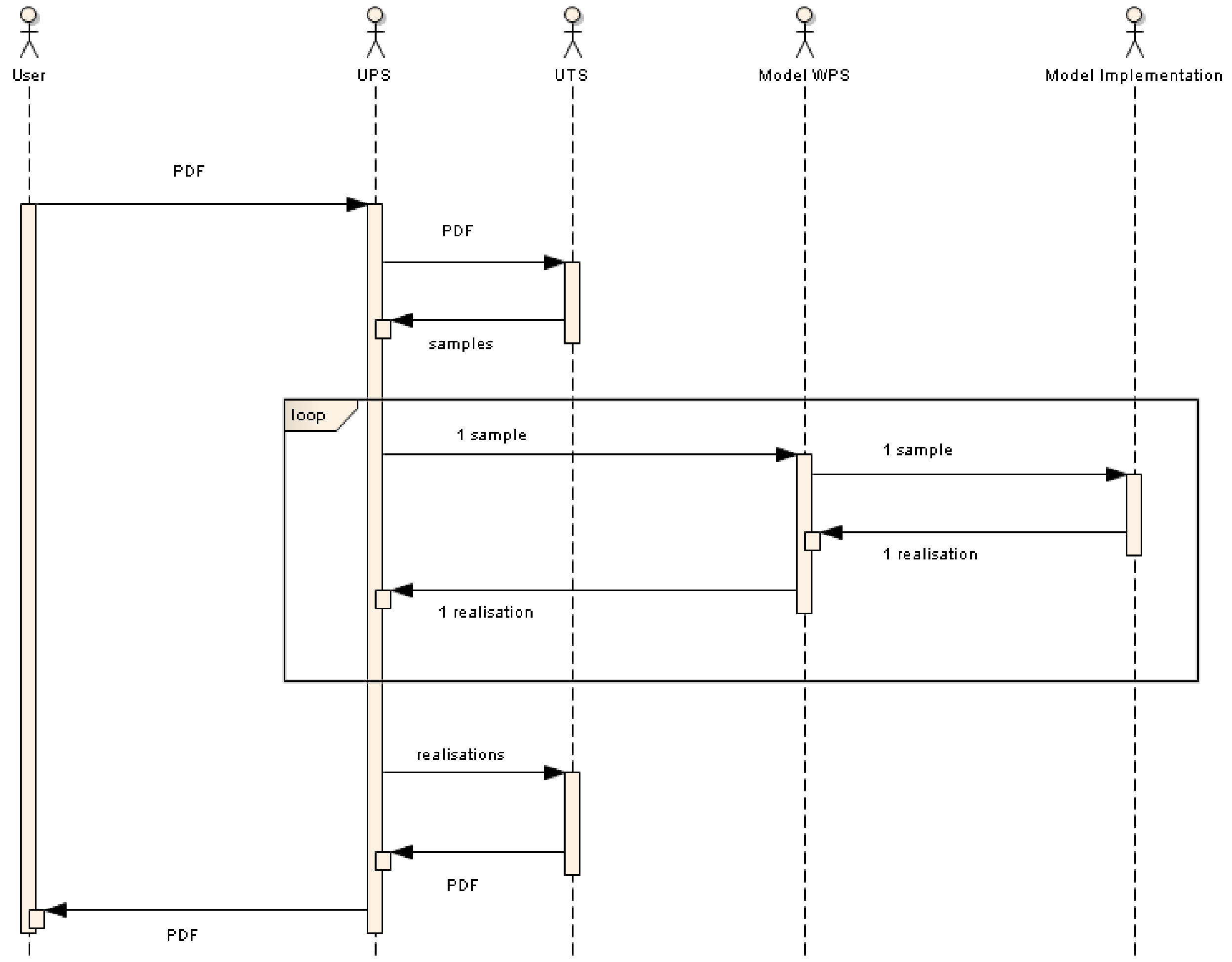


Figure 9. Extended UPS workflow.

The application of these general patterns (i.e. the internal workflow of the "ExecuteMonteCarloSimulation" process) depends on the model WPS request. An example for the extended workflow is given in the case study in Section 3.1.

Implementation

We developed a prototype UPS based on a 52°North WPS implemented in Java. The modular framework of the WPS made it easy to integrate the encodings. The implementation was successfully tested with a simple model that required no pre-processing of the input data. The internal workflow of the process follows the extended workflow pattern (see Figure 9). Figure 10 shows an execute request for the process.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<wps:Execute service="WPS" version="1.0.0" xmlns:wps="http://www.opengis.net/wps/1.0.0" xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
  http://schemas.opengis.net/wps/1.0.0/wpsExecute_request.xsd">
  <ows:Identifier>ExecuteMonteCarloSimulation</ows:Identifier>
  <wps:DataInputs>
    <wps:Input>
      <ows:Identifier>IdentifierSimulatedProcess</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>org.uncertweb.wps.SimpleRegressionModelProcess</wps:LiteralData>
      </wps:Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>UncertainProcessInputs</ows:Identifier>
      <wps:Data>
        <wps:ComplexData mimeType="text/xml" schema="http://giv-uw.uni-muenster.de:8080/uts/schemas/UncertainInputType.xsd">
          <uw:UncertaintInput xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:wps="http://www.opengis.net/wps/1.0.0"
          xmlns:uw="http://www.uncertweb.org" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.uncertweb.org http://v-mars.uni-muenster.de/uncertweb/schema/UncertainInput.xsd">
            <ows:Identifier>Test</ows:Identifier>
            <Data>
              <ComplexData mimeType="text/xml" schema="http://giv-uw.uni-muenster.de:8080/uts/schemas/uncertml2.0.0/GaussianDistribution.xsd">
                <un:GaussianDistribution xmlns:un="http://www.uncertml.org/2.0">
                  <un:mean>20</un:mean>
                  <un:variance>5</un:variance>
                </un:GaussianDistribution>
              </ComplexData>
            </Data>
          </uw:UncertaintInput>
        </wps:ComplexData>
      </wps:Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>StaticProcessInputs</ows:Identifier>
      <wps:Data>
        <wps:ComplexData mimeType="text/xml" schema="http://giv-uw.uni-muenster.de:8080/uts/schemas/StaticInputType.xsd">
          <un:StatictInput xmlns:ows="http://www.opengis.net/ows/1.1" xmlns:wps="http://www.opengis.net/wps/1.0.0"
          xmlns:un="http://www.uncertweb.org" xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xsi:schemaLocation="http://www.uncertweb.org http://v-mars.uni-muenster.de/uncertweb/schema/StaticInput.xsd">
            <ows:Identifier>input</ows:Identifier>
            <wps:Data>
              <wps:LiteralData>20</wps:LiteralData>
            </wps:Data>
          </un:StatictInput>
        </wps:ComplexData>
      </wps:Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>ServiceURL</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>http://localhost:8080/uts/WebProcessingService</wps:LiteralData>
      </wps:Data>
    </wps:Input>
    <wps:Input>
      <ows:Identifier>NumberOfRealisations</ows:Identifier>
      <wps:Data>
        <wps:LiteralData>10000</wps:LiteralData>
      </wps:Data>
    </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm>
    <wps:ResponseDocument>
      <wps:Output mimeType="text/xml" schema="http://giv-uw.uni-muenster.de:8080/uts/schemas/uncertml2.0.0/GaussianDistribution.xsd">
        <ows:Identifier>UncertainProcessOutputs</ows:Identifier>
      </wps:Output>
    </wps:ResponseDocument>
  </wps:ResponseForm>
</wps:Execute>
```

Figure 10. Example UPS execute request.

2.2.1.3 Spatio-Temporal Aggregation Service (STAS)

The Spatio-Temporal Aggregation Service (STAS) is a web service that allows converting observations and model outputs from one spatio-temporal resolution to another by means of spatio-temporal aggregation processes. It is used in the UncertWeb project to mediate between services with differing spatio-temporal resolutions of in- and outputs. The STAS supports spatial, temporal or spatio-temporal aggregation. Similar to the services described before, the STAS is defined as a profile of the WPS. Therefore, a model for defining aggregation processes has been developed that allows the definition of input sources, grouping predicates, aggregation functions that should be applied and additional parameters of such aggregation functions. The supported aggregation functions are simple statistics such as MEAN, MIN, etc. Currently, the implementation of the STAS is extended to support the rescaling of coverage data encoded as NetCDF and to allow for complex aggregation procedures including interpolation and classification steps.

2.2.2 SOAP/WSDL

In addition to using the WPS standard, a number of processes have been exposed using a custom SOAP/WSDL framework. SOAP is a lightweight protocol that allows messages to be exchanged independent of programming language or other implementation detail [W3C 2000], and WSDL is an XML specification for describing network services [W3C 2001]. Both SOAP and WSDL are mature web service technologies, and have a wide array of support in client and server tools.
Inadequate support for WSDL within the WPS 1.0.0 specification provided motivation for developing this framework. While it is possible to provide a WSDL document for a WPS instance [OGC 2007], request and response messages for a process can only be described in a weak, abstract manner. This is due to the generic nature of the specification, as it is intended the user issues a `DescribeProcess` request to discover more about a process. For a WSDL document to be usable within tools and software, and understandable for the user, message descriptions need to be concrete.
The framework is built on Java Servlet technology, and allows a developer to expose a process on the web by extending a single abstract class, with message serialisation and de-serialisation being performed automatically. Process requests and responses follow a standard pattern (see Figure 11), in a similar manner to the WPS. However, request and response messages are fully described in the WSDL document, complete with references to any XML schema elements for input and output types (e.g. an O&M measurement, or a GML point).

```
<?xml version="1.0" encoding="UTF-8"?>
<soap:Envelope xmlns:soap="http://schemas.xmlsoap.org/soap/envelope/" xmlns:ps="
http://www.uncertweb.org/ProcessingService">
  <soap:Header/>
  <soap:Body>
    <ps:ExampleProcessRequest>
      <ps:A>0.444</ps:A>
      <ps:B>
        <gml:Point xmlns:gml="http://www.opengis.net/gml/3.2" gml:id="point1">
          <gml:pos srsName="http://www.opengis.net/def/crs/EPSG/0/4326">52.87 7.78</gml:pos>
        </gml:Point>
      </ps:B>
      <ps:C>
        <ps:DataReference href="http://uncertws.aston.ac.uk/data/example_point.xml" mimeType="text/xml"/>
      </ps:C>
      <ps:RequestedOutputs>
        <ps:O reference="true"/>
      </ps:RequestedOutputs>
    </ps:ExampleProcessRequest>
  </soap:Body>
</soap:Envelope>
```
Figure 11. Request message for ‘ExampleProcess’ with inputs ‘A’, ‘B’, and ‘C’

The concrete WSDL documents served by the framework can be used in client code generation tools such as Apache Axis, and workflow creation software such as Taverna and

Kepler. In addition to SOAP/WSDL, a JSON interface is also available, allowing JavaScript web applications to be created with ease.

2.3 Discovery and Access

This section is taken from the UncertWeb Deliverable 2.2 [UW-SF 2011] (L. Bigagli, M. Santoro, V. Angelini, P. Mazzetti, S. Nativi).

In UncertWeb, the discovery and chaining of modelling resources, and processing resources in general, are implemented by the Composition-as-a-Service (CaaS) system, providing the necessary interoperability service framework for adaptation, reuse and complementation of existing resources (including geospatial services, as well as uncertainty-aware services) in the form of executable workflows.
.

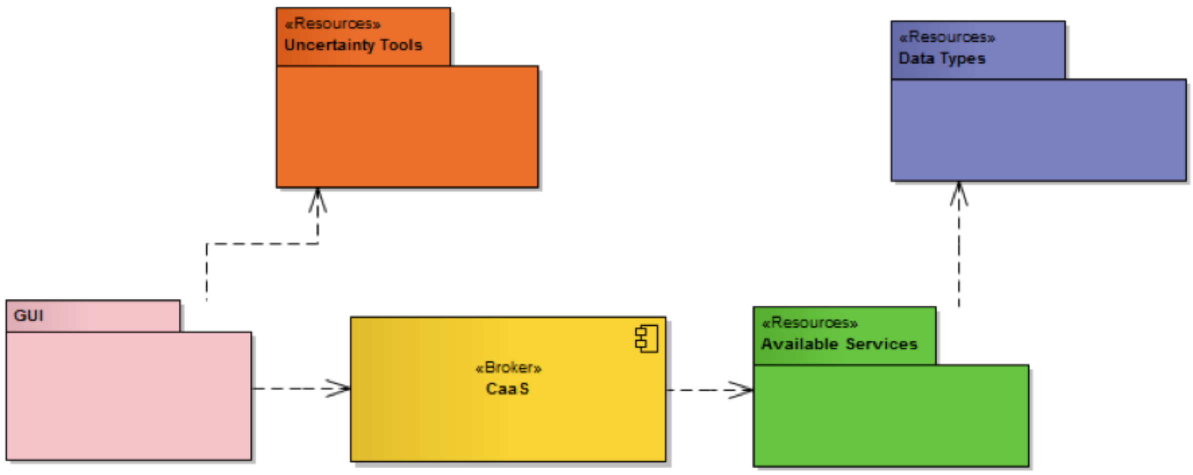


Figure 12. General view of the UncertWeb architecture with CaaS component

As depicted in the general view represented in Figure 12, the CaaS component is at the core of the UncertWeb architecture, playing two main roles:

- As a mediator, the CaaS is able to access the available services handling mismatches in the interface and (meta)data models;
- As an orchestrator, the CaaS is able to invoke the necessary services in a correct order to execute a complex EBPs (Executable Business Processes)

The current implementation of the CaaS is based on JBoss jBPM and has been successfully tested and evaluated in two use scenarios identified in previous phases of the project. The details of the CaaS are described in the UncertWeb Deliverable 2.2 [UW-SF 2011].

2.4 Visualisation

In UncertWeb, visualisation of model results, input data or intermediate results plays an important role for communicating the associated uncertainties. From the requirements in the application scenarios as collected in D8.1, specific functionalities for an interactive web-based visualisation tool could be derived as shown below. Following the paradigm of UncertWeb the tool is focused on the visualisation of quantified probabilistic uncertainty.

Specific requirements for the visualisation tool:

- Support non-spatial and non-temporal visualisation (e.g. summary statistics, PDF, charts)
- Support spatial, temporal and spatio-temporal data visualisation
- Support raster (NetCDF) and vector data (O&M, JSON) visualisation
- Support visualisation of continuous and categorical data
- Support web-based, interactive visualisation of UncertWeb component inputs, outputs and intermediate results

2.4.1 Implementation

The visualisation client developed in UncertWeb is based on the OpenLayers JavaScript API[12]. The first prototype supports visualisation of mean, standard deviation/variance, PDF, error bars, and cumulative/interval/exceedance probability for temporal, continuous vector and raster data. Interaction as well as the visualisation of the data plots is realised using JavaScript.

Vector data

In the visualisation client, support for vector data as OGC O&M 1.0 as well as for JSON, following the encoding conventions in Section 2.1, is implemented. Figure 13 shows the visualisation prototype for a time series at point locations with uncertainties. The 95 % confidence intervals are given for the time series at one particular location and the PDF is shown for the selected observation at 00:00. For each feature, the time series of observations as well as the PDF for each observation can be displayed in additional windows. The visualisation of the uncertainty in the time series and in the map can be realised as confidence intervals, error bars or exceedance probabilities for a defined threshold. The JavaScript libraries flot[13] and jstat[14] are used for time series and PDF plotting respectively. The jstat library was developed in UncertWeb for providing statistical operations like PDF definitions, plotting and transformation via JavaScript. Phenomenon and uncertainty can be visualised together with base map layers for the area of interest.

A demonstration prototype of the client using JSON encoding can be found at[15]

[12] http://openlayers.org

[13] http://code.google.com/p/flot/

[14] http:// www.jstat.org

[15] http://giv-uw.uni-muenster.de/openlayers/?json=json/gaussian.json#

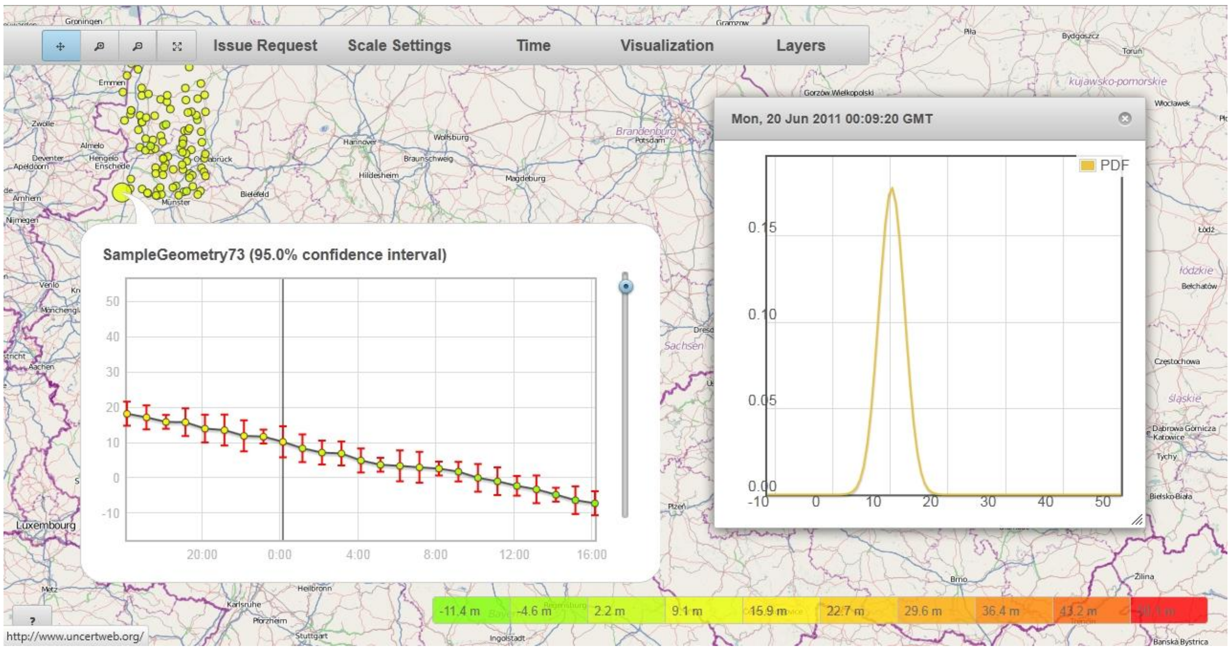


Figure 13. Uncertainty visualisation client showing a point measurement time series with error bars of 95 % confidence intervals and the probability distribution function for the respective measurement point.

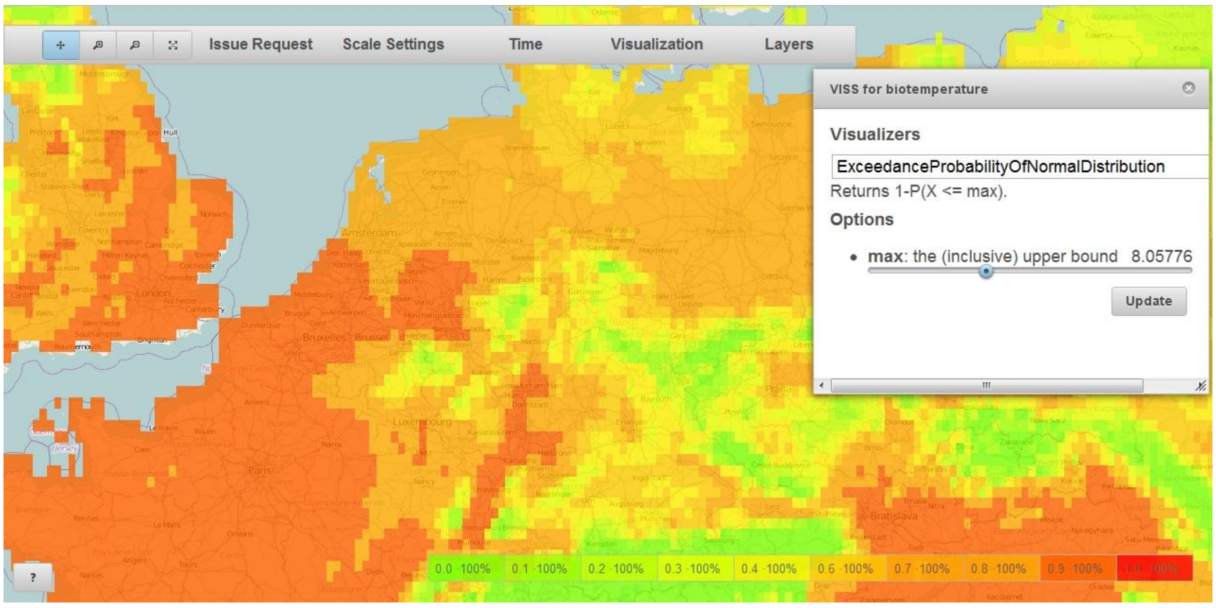
Raster data

For raster data an additional server component is used to render the raster visualisations. The Visualisation server is based on the Geoserver[16] implementation of the OGC Web Mapping Service. The user can choose between different visualisation methods (mean, variance, standard deviation, cumulative/interval/exceedance probabilities) and thresholds if applicable. In Figure 14 an example for exceedance probability for biotemperature over Europe is given. The visualisation request is sent to the Visualisation server which checks if the resource is already available (e.g. from an earlier request). If not, the requested layer is produced by the Geoserver WMS and sent back to the client in JSON format. The JSON file contains minimum and maximum values (required for visualisation), the WMS-URL and the WMS layer name for the OpenLayers client to create the layer. To allow a smooth interaction of the user with the client, a number of realisations, e.g. for each percentile, can be pre-processed and stored as layers on the server.

A demonstration prototype of the client using NetCDF-U encoding can be found at[17]

[16] http://geoserver.org

[17] http://giv-uw.uni-muenster.de/vis/?netcdf=http://giv-uw.uni-muenster.de/vis/raster/biotemp.nc#



Visualizers

ExceedanceProbabilityOfNormalDistribution

Returns 1-P(X <= max).

Options

- **max**: the (inclusive) upper bound 8.05776

Figure 14. Uncertainty visualisation client showing a raster with Exceedance Probabilities of crossing a user specified threshold in temperature (°C). .

3 Example case studies

3.1 Uncertainty-enabled local air quality model chain for Germany

This first case study intends to test the developed UncertWeb components in a simple model scenario. This scenario involves the modelling of air quality parameters, with associated uncertainty due to uncertain inputs, in an urban area. Therefore, a dispersion air quality model was i) brought to the web as a model service and was ii) uncertainty-enabled using the developed UncertWeb architecture. A simple chain of necessary components was implemented to test the practicality of the components within a “real-world” scenario. An evaluation of the resulting architecture shows the benefits and draw backs of the developed components in the test case.

3.1.1 Scenario

Analysing air quality information on an urban level is necessary to identify hazards and exposure levels that can cause negative health effects for individuals in the urban population. Modelling air quality using emission dispersion models enables the estimation of the concentration of a pollutant at un-sampled locations if continuous measurements are not available at a high spatial density. The modelling of air quality information on an urban level introduces a number of simplifications and errors leading to uncertainties in the model results. These need to be quantified and communicated to the users.

The scenario comprises the implementation of a prototypical system using a pre-configured web service chain for estimating and presenting PM_{10} (particulate matter with aerodynamic diameter smaller 10 µm) concentration in an urban area. The PM_{10} concentration produced by local urban emissions like traffic is modelled using the Lagrangian dispersion model AUSTAL2000[18]. AUSTAL2000 takes emission strengths of local sources, meteorology and

[18] http://www.austal2000.de

land use as inputs to estimate the dispersion of particles over the study area. Input uncertainties are quantified for street traffic emissions and for wind speed and direction using sample measurements. Concentration outputs can be requested as hourly time series for a number of receptor points.

The system will be able to model PM_{10} concentrations with uncertainties for point locations, e.g. tracked via GPS. As the model itself is deterministic and does not account for model error, the quantified uncertainties in the output will be caused by propagation from uncertain input. The estimates of local concentration can be combined with background concentration by adding other data sources like a SOS providing background measurements to the chain.

3.1.2 Architecture

An overview of the air quality use case architecture is shown in Figure 15. The UncertWeb components used in this scenario are the UPS and UTS as processing components, the UncertML API, the O&M profile, the UncertWeb information model API and the visualisation tool. The workflow as shown in Figure 16 is managed by the UPS which takes the uncertain inputs (emissions and meteorology) as PDFs encoded in O&M and UncertML from the workflow client. The inputs are sent to the UTS to draw samples which are used to execute the air quality model in a loop. The AUSTAL2000 model software is accessed through an OGC WPS interface which takes the O&M emission and meteorology samples as inputs (without uncertainties), converts them to the AUSTAL2000 specific format, executes AUSTAL2000 and parses the output into O&M samples at the GPS point locations. These point locations were sent to the Austal WPS as static inputs. Due to the model run time only a small set of samples is executed to yield a set of output realisations. The UPS collects the samples and sends them as UncertML realisations to the visualisation tool.

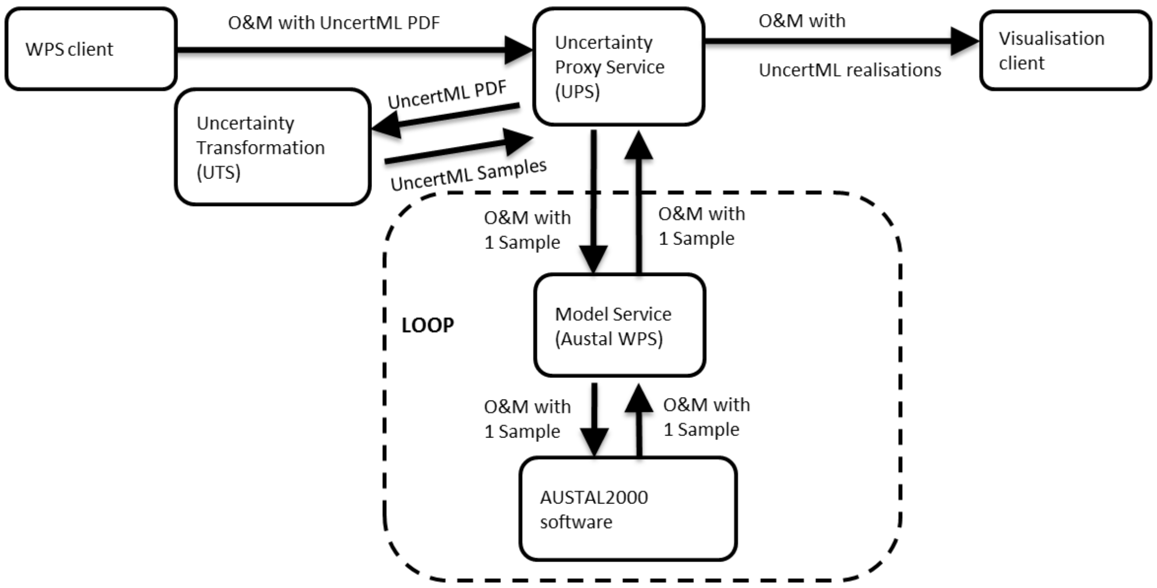


Figure 15. Architecture overview of the processing components in the scenario showing uncertain inputs only.

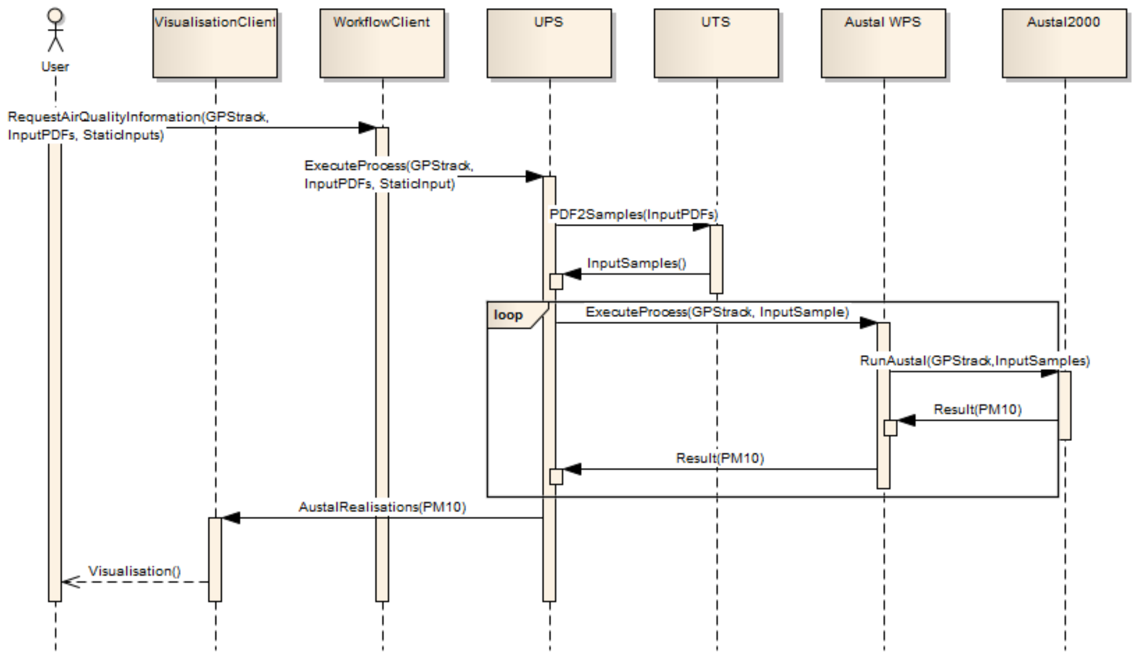

User
VisualisationClient
WorkflowClient
UPS
UTS
Austal WPS
Austal2000
RequestAirQualityInformation(GPStrack, InputPDFs, StaticInputs)
ExecuteProcess(GPStrack, InputPDFs, StaticInput)
PDF2Samples(InputPDFs)
InputSamples()
loop
ExecuteProcess(GPStrack, InputSample)
RunAustal(GPStrack,InputSamples)
Result(PM10)
Result(PM10)
AustalRealisations(PM10)
Visualisation()

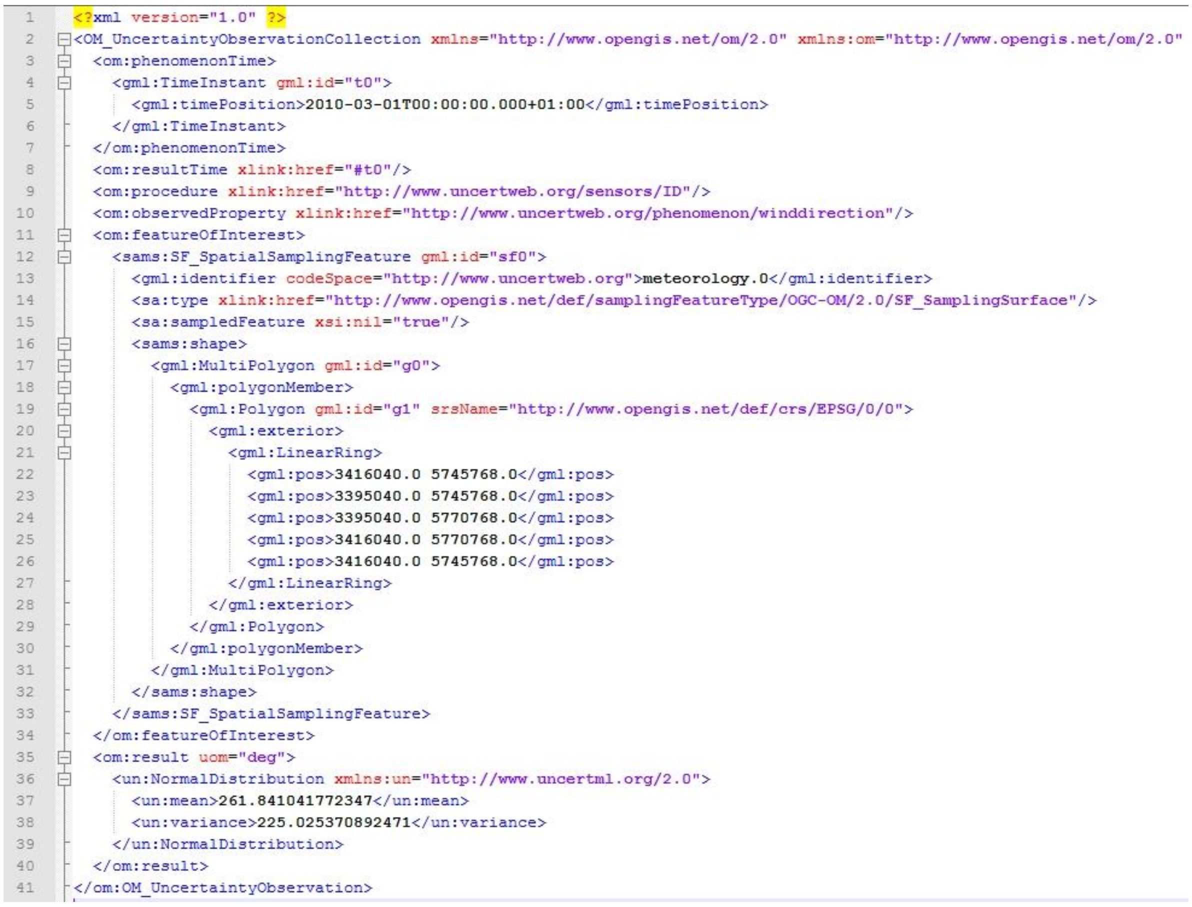
Figure 16. Scenario workflow.

3.1.3 Implementation

All processing components used in this scenario are based on the 52°North WPS implementation.

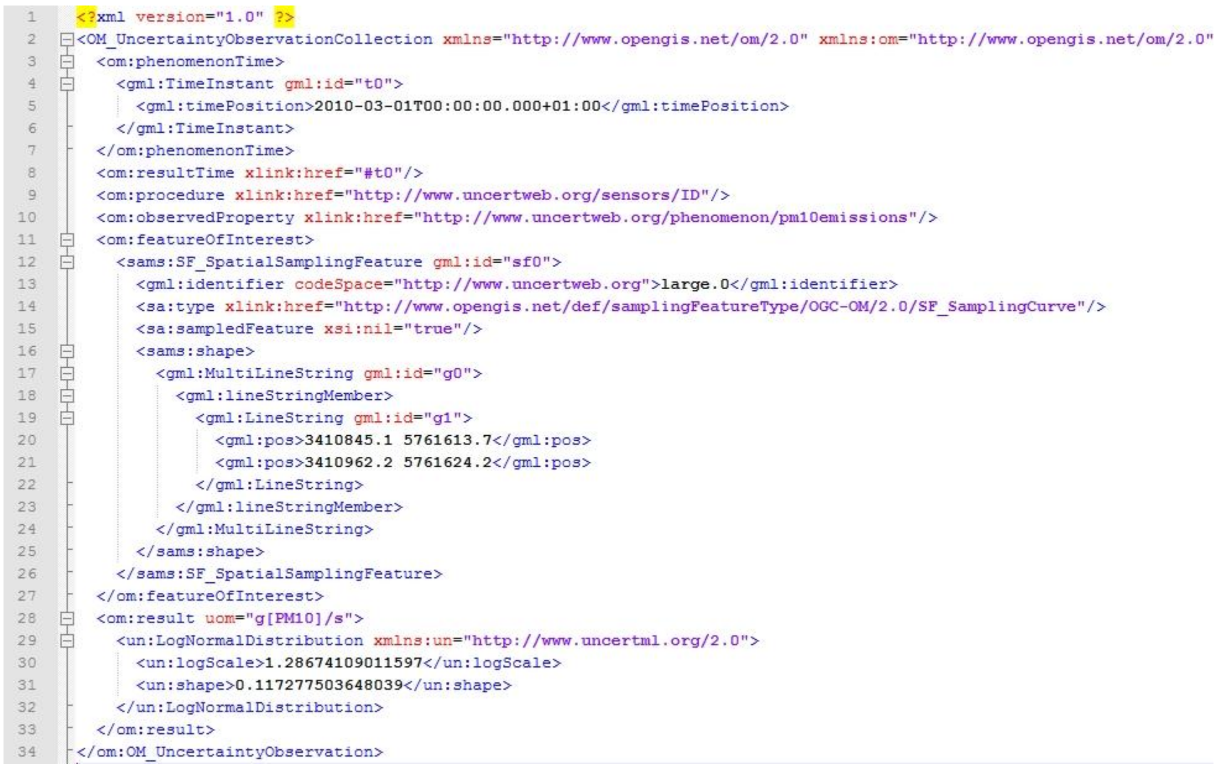
Uncertain Inputs

The inputs with uncertainty information were the meteorology data (wind speed and wind direction) and the PM_{10} emission rates for street traffic in Münster. The wind data uncertainty was quantified using a comparison of two measurement stations in and close to the study area and were described using a normal distribution in UncertML (see excerpt in Figure 17). The street traffic was estimated using hourly traffic counts at different days and point locations within the study area. Lognormal distributions were fitted to the average daily emissions and multivariate normal distributions were used to represent the percentages of the daily emissions occurring per hour of the day. Both street traffic data sets were encoded in O&M combined with UncertML lognormal and multivariate normal distributions (see the example for the daily sum of emissions in Figure 18).



```xml
<?xml version="1.0" ?>
<OM_UncertaintyObservationCollection xmlns="http://www.opengis.net/om/2.0" xmlns:om="http://www.opengis.net/om/2.0"
  <om:phenomenonTime>
    <gml:TimeInstant gml:id="t0">
      <gml:timePosition>2010-03-01T00:00:00.000+01:00</gml:timePosition>
    </gml:TimeInstant>
  </om:phenomenonTime>
  <om:resultTime xlink:href="#t0"/>
  <om:procedure xlink:href="http://www.uncertweb.org/sensors/ID"/>
  <om:observedProperty xlink:href="http://www.uncertweb.org/phenomenon/winddirection"/>
  <om:featureOfInterest>
    <sams:SF_SpatialSamplingFeature gml:id="sf0">
      <gml:identifier codeSpace="http://www.uncertweb.org">meteorology.0</gml:identifier>
      <sa:type xlink:href="http://www.opengis.net/def/samplingFeatureType/OGC-OM/2.0/SF_SamplingSurface"/>
      <sa:sampledFeature xsi:nil="true"/>
      <sams:shape>
        <gml:MultiPolygon gml:id="g0">
          <gml:polygonMember>
            <gml:Polygon gml:id="g1" srsName="http://www.opengis.net/def/crs/EPSG/0/0">
              <gml:exterior>
                <gml:LinearRing>
                  <gml:pos>3416040.0 5745768.0</gml:pos>
                  <gml:pos>3395040.0 5745768.0</gml:pos>
                  <gml:pos>3395040.0 5770768.0</gml:pos>
                  <gml:pos>3416040.0 5770768.0</gml:pos>
                  <gml:pos>3416040.0 5745768.0</gml:pos>
                </gml:LinearRing>
              </gml:exterior>
            </gml:Polygon>
          </gml:polygonMember>
        </gml:MultiPolygon>
      </sams:shape>
    </sams:SF_SpatialSamplingFeature>
  </om:featureOfInterest>
  <om:result uom="deg">
    <un:NormalDistribution xmlns:un="http://www.uncertml.org/2.0">
      <un:mean>261.841041772347</un:mean>
      <un:variance>225.025370892471</un:variance>
    </un:NormalDistribution>
  </om:result>
</om:OM_UncertaintyObservation>
```

Figure 17. Example O&M document containing the normal distribution for the wind direction



```xml
<?xml version="1.0" ?>
<OM_UncertaintyObservationCollection xmlns="http://www.opengis.net/om/2.0" xmlns:om="http://www.opengis.net/om/2.0"
  <om:phenomenonTime>
    <gml:TimeInstant gml:id="t0">
      <gml:timePosition>2010-03-01T00:00:00.000+01:00</gml:timePosition>
    </gml:TimeInstant>
  </om:phenomenonTime>
  <om:resultTime xlink:href="#t0"/>
  <om:procedure xlink:href="http://www.uncertweb.org/sensors/ID"/>
  <om:observedProperty xlink:href="http://www.uncertweb.org/phenomenon/pm10emissions"/>
  <om:featureOfInterest>
    <sams:SF_SpatialSamplingFeature gml:id="sf0">
      <gml:identifier codeSpace="http://www.uncertweb.org">large.0</gml:identifier>
      <sa:type xlink:href="http://www.opengis.net/def/samplingFeatureType/OGC-OM/2.0/SF_SamplingCurve"/>
      <sa:sampledFeature xsi:nil="true"/>
      <sams:shape>
        <gml:MultiLineString gml:id="g0">
          <gml:lineStringMember>
            <gml:LineString gml:id="g1">
              <gml:pos>3410845.1 5761613.7</gml:pos>
              <gml:pos>3410962.2 5761624.2</gml:pos>
            </gml:LineString>
          </gml:lineStringMember>
        </gml:MultiLineString>
      </sams:shape>
    </sams:SF_SpatialSamplingFeature>
  </om:featureOfInterest>
  <om:result uom="g[PM10]/s">
    <un:LogNormalDistribution xmlns:un="http://www.uncertml.org/2.0">
      <un:logScale>1.28674109011597</un:logScale>
      <un:shape>0.117277503648039</un:shape>
    </un:LogNormalDistribution>
  </om:result>
</om:OM_UncertaintyObservation>
```

Figure 18. Example O&M document containing the lognormal distribution for the daily average PM_{10} emissions.

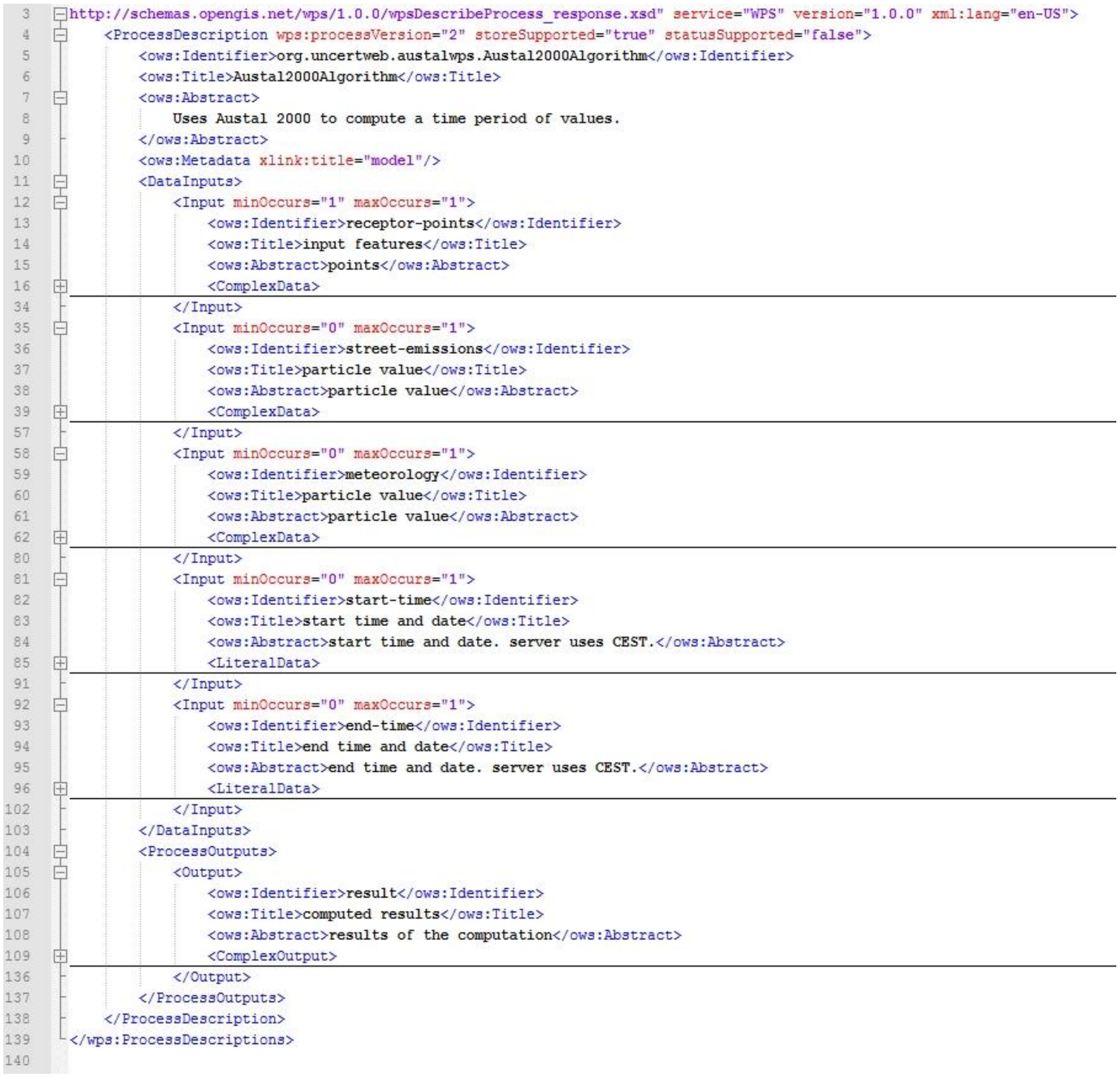
UPS
The UPS offers a “MonteCarloSimulation” process as shown in Figure 10 which takes the static and uncertain process inputs as well as an execute request and the service URL for the Austal WPS. In the following the workflow of an “ExecuteMonteCarloSimulation” process as shown in Figure 16, including interaction with a UTS, is described in detail.

Call UPS MonteCarloSimulation process

- Get NumberOfRealisations input
- Get UncertainProcessInputs
    - Get streetEmission input
        - Extract UncertML from O&M objects (lines and polygons)
            - For each street object
                - Call UTS Lognormal2Samples for daily emissions
                - Call UTS Multivariate2Samples for hourly fractions of emissions
                - Multiply samples to get hourly sums of emissions
    - Get meteorology input
        - Extract UncertML from O&M objects (polygon for area of Münster)
            - Call UTS Gaussian2Samples for wind speed and wind direction
- Get StaticProcessInputs
    - Get receptor point input (GPS track)
- Get ServiceURL and IdentifierSimulatedProcess
    - For each NumberOfRealisations
        - Call Austal2000 WPS
            - Run Austal2000 with one UTS sample for each input
    - Collect results per receptor point
- Get UncertainProcessOutputs
    - If necessary call UTS again to transform samples to desired output
- Call Visualisation client to visualise point output

UTS
The UTS required three processes for this use case: Multivariate2Samples, Lognormal2Samples and Gaussian2Samples. As no spatial sampling was required, the UTS gets the UncertML documents containing the distribution parameter but without the spatial and temporal information of the O&M encoding and uses simple sampling methods in R based on the distribution parameters.

Model service
AUSTAL2000 is available as a Windows executable or as a Linux distribution and needs the input data in the form of two text files. The model service is based on the 52°North WPS implementation. This WPS implementation offers the “Austal2000algorithm” process to change the input files and execute the AUSTAL2000 software. A process description is given in Figure 19. Mandatory inputs are the receptor points, i.e. the locations that the PM_{10} concentration should be estimated for. Optional inputs are meteorology, street emission and start/end time. If no meteorological or emission inputs are given, the Austal WPS will use the default input data which is stored in the server. Input data is available for the period of July 2008 to June 2010. If input data is given with the ExecuteProcess request, the WPS will substitute only the respective data, e.g. street emissions. After AUSTAL2000 finishes the model run, the WPS parses the results for the receptor points and sends them back as O&M encoded observations.



```xml
http://schemas.opengis.net/wps/1.0.0/wpsDescribeProcess_response.xsd" service="WPS" version="1.0.0" xml:lang="en-US">
    <ProcessDescription wps:processVersion="2" storeSupported="true" statusSupported="false">
        <ows:Identifier>org.uncertweb.austalwps.Austal2000Algorithm</ows:Identifier>
        <ows:Title>Austal2000Algorithm</ows:Title>
        <ows:Abstract>
            Uses Austal 2000 to compute a time period of values.
        </ows:Abstract>
        <ows:Metadata xlink:title="model"/>
        <DataInputs>
            <Input minOccurs="1" maxOccurs="1">
                <ows:Identifier>receptor-points</ows:Identifier>
                <ows:Title>input features</ows:Title>
                <ows:Abstract>points</ows:Abstract>
                <ComplexData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>street-emissions</ows:Identifier>
                <ows:Title>particle value</ows:Title>
                <ows:Abstract>particle value</ows:Abstract>
                <ComplexData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>meteorology</ows:Identifier>
                <ows:Title>particle value</ows:Title>
                <ows:Abstract>particle value</ows:Abstract>
                <ComplexData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>start-time</ows:Identifier>
                <ows:Title>start time and date</ows:Title>
                <ows:Abstract>start time and date. server uses CEST.</ows:Abstract>
                <LiteralData>
            </Input>
            <Input minOccurs="0" maxOccurs="1">
                <ows:Identifier>end-time</ows:Identifier>
                <ows:Title>end time and date</ows:Title>
                <ows:Abstract>end time and date. server uses CEST.</ows:Abstract>
                <LiteralData>
            </Input>
        </DataInputs>
        <ProcessOutputs>
            <Output>
                <ows:Identifier>result</ows:Identifier>
                <ows:Title>computed results</ows:Title>
                <ows:Abstract>results of the computation</ows:Abstract>
                <ComplexOutput>
            </Output>
        </ProcessOutputs>
    </ProcessDescription>
</wps:ProcessDescriptions>
```

Figure 19. Process description for the Austal2000algorithm process.

3.1.4 Evaluation

Using the UPS as a “proxy service” enabled us to reuse an existing implementation of the Austal model WPS. The only additional implementation done for the model service was to allow further inputs, i.e. meteorology and emissions. All uncertainties are processed by the UPS and hidden from the model service.

Although not implemented here, this concept also allows the parallelisation of Austal model runs as the UPS manages the loop. Therefore, the UPS can distribute the different input samples which can be run independently on different model service implementations. This has a special advantage as environmental models might have a considerable running time (in this case several minutes) which makes a parallel execution more appealing.

One disadvantage discovered during implementation of this case study was the communication overhead produced by the components. Using UncertML with O&M for hundreds of emission sources lead to considerably large XML documents when encoding several days. One reason was the bulky encoding of multivariate distributions including a 24 x 24 matrix containing covariances. Another reason was the separate encoding of each observation as a new UncertML observation in the `om:result` tag. Using time series encodings like the `swe:DataArray` could reduce the overhead, especially for long time series, but only for a small number of features.

3.2 Modelling land-use change in the UK

3.2.1 Scenario

This scenario has been developed within WP5, and is composed of a set of models for predicting land-use and crop yield response to climatic and economic change. In future years, the capability of the UK's agricultural land may change. Such change can be driven by many factors, including government policy and weather and climatic conditions. For example, the climate predictions from UKCP09 show marked changes in England's climate over the next century. Temperatures and CO_2 levels are expected to increase, and rainfall patterns are expected to become more variable with the possibility of drought events (especially in summer periods) occurring more frequently. One possible impact here is that these trends might lead to a decrease in water availability during crucial crop growth stages and more crops experiencing water stresses, which would have a negative impact on both crop yields and quality.

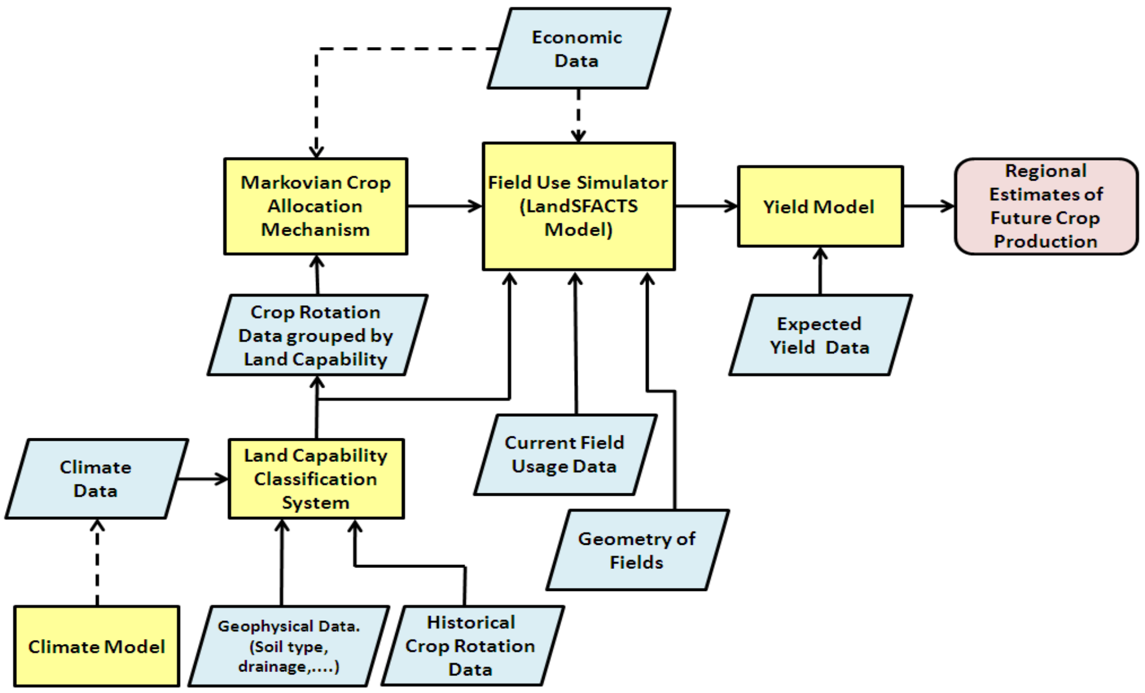

Economic Data
Markovian Crop Allocation Mechanism
Field Use Simulator (LandSFACTS Model)
Yield Model
Regional Estimates of Future Crop Production
Crop Rotation Data grouped by Land Capability
Expected Yield Data
Current Field Usage Data
Climate Data
Land Capability Classification System
Geometry of Fields
Climate Model
Geophysical Data. (Soil type, drainage,….)
Historical Crop Rotation Data

Figure 20. An overview of the land-use and crop yield response workflow.

The workflow designed in WP5 will be used to investigate the potential impacts of changes in climate and policy on future agricultural production. As illustrated by Figure 20the workflow here consists of the following four models:

1. a Climate Model.
2. a Land Classification Model (Following discussions regarding model granularity, the Markovian Crop Allocation Mechanism was integrated into the Land Capability Classification System to form one Land Classification Model (LCM).)
3. a Field Use Simulator Model, and
4. a Yield Model.

For the field use simulator model, we will use LandSFACTS[19] - a model developed by the Macaulay Land Use Research Institute. At the present time, the climate and yield models are still under consideration, and are therefore not yet integrated into the current workflow. For the yield model, we are likely to use the Aquacrop model[20] – a model that has been developed by the FAO. This crop yield model is transpiration-driven, and therefore suited to the kinds of issues (such as climate impacts) that we would like to explore.

3.2.2 Architecture

At the time of development, only two models were available for use: the LCM and LandSFACTS. The final proposed workflow will produce crop yield estimates, but at this stage the output is the simulated crop rotations for each field. The workflow is as follows:

- Call LCM
    - Parse historical crop data and soil data data
    - Execute LCM
    - Output crop transition distributions
- Call LandSFACTS
    - Parse crop transition distributions
    - Parse soil and area data
    - Sample field transition matrices
        - Extract Dirichlet distributions from distribution arrays
        - Generate required number of samples from each distribution
    - For each sample, execute LandSFACTS model
    - Output simulated crop rotations

Each model is exposed as a process using the SOAP/WSDL and JSON web service framework. With the automatically generated WSDL document, it is possible to orchestrate the workflow using a BPEL engine, or software such as Taverna or Kepler. To enable interoperability, standard data types defined by the UncertWeb GML and O&M profiles, are used where appropriate.

Due to data licensing issues, data services will not be used at this stage. Once these issues are resolved, historical crop data, soil data, and area data will be hosted on a service such as an OGC Sensor Observation Service (SOS). For this example the data will be explicitly embedded in the request, rather than linking to a data service.

3.2.3 Implementation

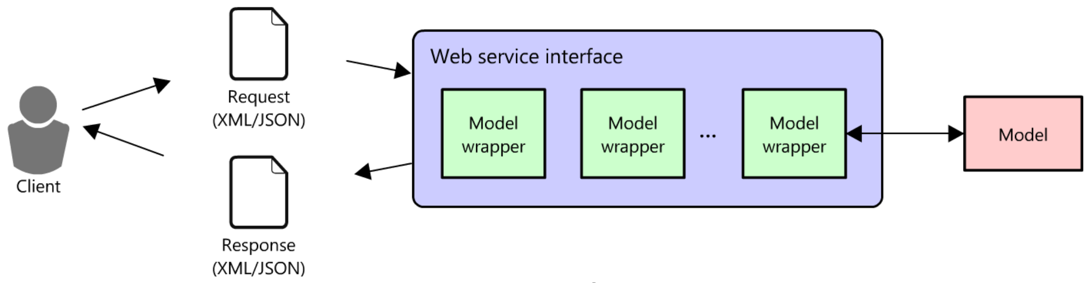


Figure 21, Model wrapping inside the web service interface

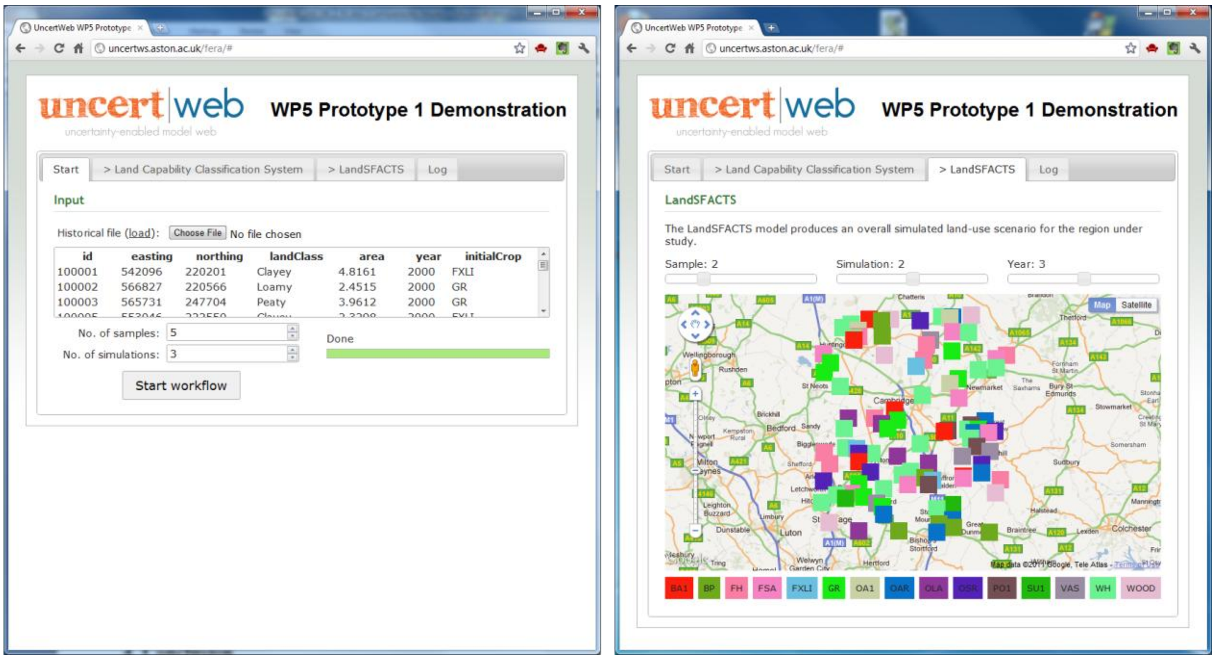
[19] http://www.macaulay.ac.uk/LandSFACTS/

[20] http://www.fao.org/nr/water/aquacrop.html

As we are using existing models, written in various languages, it is not feasible to place the model functionality directly in the process class on the web service interface. Instead we implement the process class as a wrapper to the underlying models. These wrappers are responsible for understanding the input on the request, translating it to a format the model understands, calling the model, and converting the output from the model to the required format for the web service interface.

The LCM was developed using R. Historical crop data and soil data is read by the model from one file, and the generated parameters of the crop transition distributions are written to another file in the form of a matrix. As O&M ObservationCollection elements are used as inputs to the wrapper process, and the format of these files is simply tabulated plain text, O&M data from a request must be converted to the tabulated format before running the R script. Once the model has been run, the tabulated output is read and converted to the appropriate transition matrix data type. Rserve is used for interaction between the Java process class and a remote instance of R.

Initially, the LandSFACTS model was only provided as a Windows executable with graphical user interface, making automated calls from a web service interface challenging. However, a standalone command line version with source code was obtained from the authors of the software. As our services run on a Linux based server, some modifications to the source code were required in order to compile a working binary. Before executing the model, samples are generated from the corresponding Dirichlet distributionfor each input transition matrix row. The model is run for each set of transition matrix samples. In a similar manner to the LCM, the wrapper process for this model generates a set of text files from O&M and transition matrix inputs. After the model has been executed, the process class reads the output files and generates a series of O&M observations, one for each simulated year for each field.



Input

Historical file (load): No file chosen

id	easting	northing	landClass	area	year	initialCrop
100001	542096	220201	Clayey	4.8161	2000	FXLI
100002	566827	220566	Loamy	2.4515	2000	GR
100003	565731	247704	Peaty	3.9612	2000	GR

No. of samples: 5
No. of simulations: 3
Done

LandSFACTS

The LandSFACTS model produces an overall simulated land-use scenario for the region under study.

Sample: 2　Simulation: 2　Year: 3

Figure 22, Screenshots from the workflow demonstration web client

To demonstrate the workflow, a simple JavaScript web client was developed[21]. For ease of development, the JSON service interface was used, as JavaScript can natively parse JSON to objects. For demonstration purposed faked crop and soil data is used which is not derived

[21] http://uncertws.aston.ac.uk/fera/

from real field data. The client is responsible for loading the historical crop data and soil data, and for orchestrating the workflow, calling each service in turn. A user is able select the number of samples to take from the transition matrix distributions, and also the number of crop allocation simulations to run using LandSFACTS.

3.2.4 Evaluation

This workflow serves to illustrate the potential of the uncertainty-enabled model web. Standardised web service interfaces have been developed for software which was previously only accessible on a local machine, and could only understand bespoke data types. Both components in the current workflow have been uncertainty enabled, with uncertainty propagated from the first model output to the second model input. The SOAP/WSDL and JSON web service framework enabled the exposed models to be used with generic workflow software and tools, and also to rapidly create a demonstration web client.

Some practical issues were experienced when connecting existing model code to web service interfaces. Our approach of creating wrapper processes for each model was very time consuming and required an understanding of both the web service side and also the underlying models, possibly presenting an entry barrier to current model owners. The propagation of uncertainty within the workflow is currently very basic, as no input uncertainty is taken into account. Ideally the transition matrices sampling in the LandSFACTS process would be undertaken by the UncertML Translation Service (UTS). This would mean that the model could be run using data without any uncertainty information, and therefore it would be more generalised and have wider applicability.

4 Conclusion

The present document describes simple and publicly accessible UncertWeb examples available as open source implementations. The tools presented are developed in the technical work packages WP1, WP2 and WP3. The solutions allow for (i) encoding and converting spatio-temporal information with probability distributions, (ii) chaining simple services that pass probability distribution information, and (iii) summarising and visualising the resulting information in a simple form. The tools have been successfully deployed in two use cases, an uncertainty-enabled local air quality prediction system and a system for predicting land-use and crop yield response to climatic and economic change. Service examples presented in this report are published in the web and accessible to the public.

The encoding profiles developed in UncertWeb allow for encoding spatio-temporal uncertainties in a standardised way and their usability has been demonstrated in the case studies. However, one problem that occurred is handling and encoding large datasets. For example, a solution for a more compact encoding of U-O&M as XML might be added in the future. The developed processing service profiles offered utilities like uncertainty transformation and aggregation for flexible usage within the model service chains.

We have intentionally considered two different approaches to implement uncertainty-enabled Web services in UncertWeb. In one approach, profiles of the WPS defined by the OGC have been developed. In the other approach, plain SOAP/WSDL services have been implemented to expose the models in the Web. Both approaches have advantages and disadvantages. As SOAP/WSDL services are widely adopted in the Web, the tools support for generating code is rather sophisticated. Furthermore, the integration in workflow software and orchestration engines is easier than using the WPS interface. However, the WPS is well defined for supporting spatial (and temporal) data and also eases the integration of models in spatial data infrastructures by relying on the common service model defined by the OGC. Harmonising the two approaches has already started at OGC and needs to be continued to ease the development and integration of environmental tools in the Web.

References

[Baranski 2009]
Baranski, B. (Ed.) (2009). OWS-6 WPS Grid Processing Profile Engineering Report. OGC 09-041r3.

[Cox 2007]
Cox, S. (Ed.), 2007. Observation and Measurements - Part 1 - Observation Schema. OGC 07-022r1. Open Geospatial Consortium, Inc., 86 pp, http://portal.opengeospatial.org/_les/?artifact id=22466, [accessed 22 March, 2011].

[OGC, 2007]
Open Geospatial Consortium, 2007: OpenGIS® Web Processing Service, http://portal.opengeospatial.org/files/?artifact_id=24151

[Pebesma 2011]
Pebesma, E., Cornford, D., Dubois, G., Heuvelink, G.B., Hristopulos, D., Pilz, J., Sthlker, U., Morin, G., Skien, J.O., 2011. INTAMAP: The design and implementation of an interoperable automated interpolation web service. Computers & Geosciences 37, 343 – 352.

[Portele 2007]
Portele, C. (Ed.), 2007 OpenGIS® Geography Markup Language (GML) Encoding Standard. OGC 07-036. http://portal.opengeospatial.org/files/?artifactid=20509 [accessed 14 September, 2011].

[Schut 2007]
Schut, P. (Ed.), 2007. OpenGIS Web Processing Service. OGC 05-007r7. Open Geospatial Consortium, Inc., 87pp.http://portal.opengeospatial.org/les/?artifact id=24151 [accessed 453 22 March, 2011].

[UW-SF 2011]
UncertWeb Consortium, “Service frameworks for modelling resources”, Deliverable 2.2, 2011.

[Williams 2009]
Williams, M.; Cornford, D.; Bastin, L., Pebesma, E. (Ed.) Uncertainty Markup Language. OGC 08-122r2. Open Geospatial Consortium, Inc., 61pp. http://portal.opengeospatial.org/files/?artifactid=33234 [accessed 22 March, 2011]

[W3C 2000].
W3C, 2000: Simple Object Access Protocol (SOAP) 1.1, http://www.w3.org/TR/soap11/

[W3C 2001].
W3C, 2001: Web Services Description Language (WSDL) 1.1, http://www.w3.org/TR/wsdl