

# UncertWeb

The *Uncertainty Enabled Model Web*

SEVENTH FRAMEWORK PROGRAMME

THEME FP7-ICT-2009-4

ICT for Environmental Services and Climate Change Adaptation

## Deliverable 1.1

UncertML requirements

Title of Deliverable	UncertML requirements
Deliverable reference number	D1.1
Related WP and Tasks	WP1, Task 1.1
Type of Document	Public
Authors	Lucy Bastin, Matthew Williams
Date	26 Nov 2010
Version	1.0

Project coordinator

Dr. Dan Cornford

Aston University, United Kingdom

E-mail: [d.cornford@aston.ac.uk](mailto:d.cornford@aston.ac.uk)

<http://www.uncertweb.org/>

## Revision History

Version	Date	Changes	Authors
0.1	04-10-2010	Initial draft	Lucy Bastin
0.2	18-10-2010	Secondary draft	Matthew Williams
0.3	30-10-2010	Review	Stefano Nativi
0.4	07-11-2010	Third draft - more formal requirements added	Lucy Bastin
1.0	26-11-2010	Final revisions - minor modifications	Dan Cornford

## Related task

### Task 1.1 UncertML requirements

A state-of-the-art workshop will be organised within the project to get everyone familiar with UncertML, uncertainty representation through probability distributions, risk assessment, and the distinction between uncertainty and variability. The workshop will be internal, but the training material will be made public, and representatives from related projects such as Envision and Tadoo will be invited. Formalise requirements for the extended UncertML standard from use cases (WP4–7), working with partners and the advisory panel. The requirements will be gathered by formal and informal interviews, and through the discussions of the use cases developed in WP4–7. A forum will be used to enable discussion of the developing requirements, which will be documented on a publicly readable wiki as they mature. The Project Advisory Board will be given the opportunity to review and provide feedback on the requirements. The requirements will be frozen at Month 9 within the project to allow development of the schema and API. However it is true of all systems that requirements change. Thus we will review and revise the requirements to produce new versions if the need arises.

Active partners: AST, UOM, WU

### **Legal Notices**

The information in this document is subject to change without notice. The Members of the UncertWeb Consortium make no warranty of any kind with regard to this document, including, but not limited to, the implied warranties of merchantability and fitness for a particular purpose. The Members of the UncertWeb Consortium shall not be held liable for errors contained herein or direct, indirect, special, incidental or consequential damages in connection with the furnishing, performance, or use of this material

## Executive Summary

This document consolidates the model requirements described in deliverables 4.1, 5.1, 6.1 & 7.1 that are relevant to UncertML - these requirements will provide the basis for a new (second) version of UncertML.

The main aim of this document is to identify the uncertainty types (i.e. statistics and distributions) that UncertML must provide in order to satisfy the needs of the UncertWeb project, and support the uncertainty-enabled model web as it develops. Thus many of the functional requirements documented here may be referred to as ‘type requirements’. The first version of UncertML relied on weak-typing of elements to allow flexibility. In response to this requirements-gathering activity, the new version will be a hard-typed model with explicit data types for each of a defined suite of statistics, distributions and other methods of quantifying uncertainty. Many of these types have been extracted from the specific UncertWeb model requirements. Further uncertainty types that have been identified as ‘commonly used’ are also included in this document to ensure that UncertML maintains applicability over a wide array of domains.

This document also describes a number of non-functional requirements for UncertML. These requirements are not explicit types of uncertainty representation, but more general ‘features’ that UncertML should provide, including multiple encodings. The results are presented as a textual description, or as a list of terms where applicable. A summary of all requirements is given in the conclusions (page 16).

The analysis and implementation of these requirements will be undertaken in subsequent deliverables, specifically deliverables 1.2 & 1.3. The aim of this document is to provide a clear set of requirements to allow UncertML to satisfy a broad range of use cases, both within UncertWeb but also outside the project.

# Contents

1	Introduction . . . . .	1
1.1	UncertML version 1.0 . . . . .	1
1.2	Explicit types supported in UncertML 1.0 . . . . .	1
1.3	Requirements for UncertML 2.0 . . . . .	2
2	UncertWeb requirements . . . . .	3
2.1	Deliverable 4.1 . . . . .	3
2.2	Deliverable 5.1 . . . . .	5
2.3	Deliverable 6.1 . . . . .	8
2.4	Deliverable 7.1 . . . . .	10
2.5	Additional Type Requirements . . . . .	11
2.6	Overarching non-functional requirements . . . . .	14
2.7	Other non-functional requirements . . . . .	15
3	Summary . . . . .	16
3.1	Summary of UncertML requirements . . . . .	16

# 1 Introduction

This deliverable documents the requirements which have been collected from UncertWeb partners as to the modelling and representation of uncertainty in the data and models which constitute their specific use cases. It considers each of these requirements as they apply to the design of version 2.0 of the UncertML language. The listed requirements are categorised by their nature, and drive the development and use of UncertML 2.0 within the UncertWeb project and beyond. The expected outcome of this exercise is a clear conceptual model, agreed by all partners, and an explicit list of uncertainty types which will be supported by UncertML 2.0. This will fulfil several broader requirements identified in Deliverable 8.1, in particular *RQ5: A variety of different statistical measures are used for uncertainty representation in UncertWeb and should be considered for support.*

## 1.1 UncertML version 1.0

The original version of UncertML (version 1.0) took the form of a single XML schema which was submitted to OGC for consideration as a standard<sup>1</sup>. It was designed to allow the interoperable encoding of statistical summaries of uncertainty based on probabilistic approaches.

The design of this schema was weak-typed: that is, it addressed the wide variety of uncertainty representations which might be required by allowing users to define their own statistics and distributions. This approach aimed to allow flexibility by, for example, allowing a user to characterise their uncertainty with a ‘Distribution’ type, whose specific description and details (including mathematical characteristics) might be defined in their own dictionary. For this reason, the original set of elements defined within UncertML was relatively small, and grouped into three major classes: Statistics, Distributions and Realisations. Many commonly-used uncertainty representations could be supported by the use of a Statistic or Distribution element in combination with a reference to the UncertML dictionary<sup>2</sup> However, some commonly-used statistics and distributions were defined as explicit elements, and these are described below.

## 1.2 Explicit types supported in UncertML 1.0

### 1.2.1 Statistics

**Quantile** — points taken at regular intervals from the cumulative distribution function (CDF) of a random variable.

**Moment** — a measure of the shape of the probability distribution of a real-valued random variable, such as skewness or kurtosis.

**DiscreteProbability** — the probability that a variable has a specific discrete value.

**Probability** — the probability that a variable exceeds (or does not exceed) a certain threshold.

---

<sup>1</sup>[http://portal.opengeospatial.org/files/?artifact\\_id=33234](http://portal.opengeospatial.org/files/?artifact_id=33234)

<sup>2</sup><http://dictionary.uncertml.org/dictionary.php>

UncertML 1.0 encoded other common statistics such as mean, median and standard deviation as generic Statistics with appropriate dictionary references. Elements were provided for collecting together statistical summary data, namely the StatisticsSet and the StatisticsArray (the latter used for multiple observations).

### 1.2.2 Distributions

All distributions in UncertML 1.0 were encoded as a Distribution type with appropriate dictionary references. Elements were provided for collecting together distribution information in useful ways, namely the MixtureModel and the DistributionArray (the latter used for multiple observations).

### 1.2.3 Realisations

The Realisations type allowed a user to collect a set of samples from a random quantity, allowing uncertainty to be described implicitly.

## 1.3 Requirements for UncertML 2.0

The design briefly described above was flexible and allowed the construction of many complex and informative uncertainty encodings. However, in the course of requirements gathering for the UncertWeb project, this design has been reformulated, with several major changes which are detailed in the following document. Some of the requirements identified below relate to specific data types and model inputs/outputs. Other relate to the more generic issues associated with automatic discovery, use and chaining of models in the context of uncertainty. The following sections detail the specific case studies of the UncertWeb project, and notes the specific requirements for each.

Three different types of requirements are identified, and these are numbered and labelled as follows:

**Functional Type Requirements.** Some case studies involve very specific uncertainty types which need to be encoded and communicated. These are labelled with the prefix ‘FTR’, as they are a special subset of the functional requirements.

**Functional Requirements.** Other functional requirements are labelled with the prefix ‘FR’.

**Non-functional Requirements.** Non-functional requirements are labelled with the prefix ‘NFR’.

There is one fundamental non-functional requirement which is met by UncertML 1.0, and will be addressed by all subsequent version, as follows:

**NFR1: UncertML will permit the communication of uncertainty information through a standardised and universally-accessible conceptual model.**

While the conceptual model will change in version 2.0, the priorities of interoperability and open access remain. The conceptual model should also be stable in this version.

The requirements identified and collected in the following sections are summarised in the Conclusions.

## 2 UncertWeb requirements

The primary source of requirements for this iteration of UncertML is the set of use cases described in UncertWeb deliverables 4.1–7.1. UncertML is integrated into many aspects of the UncertWeb project; consequently all requirements defined by these deliverables must be met. This section describes each deliverable (4.1–7.1) in detail and establishes which inputs and outputs are uncertain, how their uncertainty is quantified and what requirements this places on UncertML.

### 2.1 Deliverable 4.1

Deliverable 4.1 focuses on defining the user requirements necessary to define the means to quickly visually validate high resolution multi-temporal satellite data (Landsat images) using ground truth data within and around African Protected areas. Within D4.1 there is only a single model, e-habitat. The inputs and outputs of this model are defined below.

#### 2.1.1 e-Habitat

The e-habitat model looks at the habitat of a species or a group of species. This habitat is currently considered within defined protected areas. It is assumed that a species has the potential to persist in areas where the physical environment is optimal for the specific requirements of that species. Using the Mahalanobis distance as a measure of similarity, the model identifies and ranks other areas in the region that could possibly replace the habitat if it is threatened by climate change, deforestation, competition for land, population pressure, poaching etc.

The inputs to the e-habitat model have been simplified in table 1. While many of the uncertainties are *unknown*, a few requirements can be elicited:

**Proportional statistics** the pTreeCover input contains an uncertainty that is typically quantified using a standard deviation that is proportional to the mean value.

**FTR1: UncertML should provide a mechanism for quantifying statistics using proportions where applicable.**

**Standard deviation** a ‘standard deviation’ statistic is required, as stipulated by the elevation input. There is an argument that either Standard Deviation *or* Variance should be supported, given the ease of conversion between the two. However, an explicit differentiation between the two will help to ensure the correct characterisation of the inputs and outputs for pre-existing models which are incorporated into the uncertainty-enabled ModelWeb in future.

**FTR2: UncertML should include explicit types for recording standard deviation and variance.**

## Inputs

Name	Type	Spatial Type	Uncertainty Type	PDF
SelectedArea	Categorical	Vector	Unknown	Unknown
pTreeCover	Continuous	Raster	Standard deviation (proportional)	Unknown
elevation	Continuous	Raster	Standard deviation	Gaussian
slope	Continuous	Raster	N/A	N/A
climate	Continuous	Raster	Realisations	N/A
NDVI	Continuous	Raster	N/A	N/A
waterBodies	Continuous	Raster	Unknown	Unknown
landcover	Categorical	Raster	Probability	Unknown
E00-maps	Categorical	Vector/Raster	Unknown	Unknown

Table 1: Inputs to the e-habitat model (deliverable 4.1)

**Realisations** a method for encoding a series of realisations is required, specified by the `climate` input. Realisations are also identified as an important requirement in RQ3 of Deliverable 8.1.

**FTR3: UncertML should retain the Realisations element type included in version 1.0.**

**Gaussian distribution** The Gaussian distribution is used to describe uncertainty in the `elevation` input, where error around the mean reported value is expected to be normally distributed. This type of distribution is one of the most commonly-cited in the elicitation of uncertainty from users and experts, and arises naturally as a consequence of the central limit theorem in several settings.

**FTR4: UncertML should include an explicit GaussianDistribution type.**

**Probabilities** UncertML should provide a method for encoding probabilities for categorical data, i.e. discrete probabilities. This is deduced from the `landcover` input which is categorical data that may have a probability of misclassification.

**FTR5: UncertML should retain the explicit DiscreteProbability statistic type.**

**Standard Error** The `pTreeCover` input is itself an mean estimated output from a regression tree, and for certain ranges of the variable, uncertainty can be described as a standard error which critically depends on sample size. (This feature of the use case is also discussed, along with a variety of issues raised by all four use cases, in UncertWeb Deliverable 8.1). It is important to either provide an explicit `StandardError` type, or to standardise the encoding of the information which a user can employ to calculate this standard error on the fly.

**FTR6: UncertML should allow the communication of standard errors.**

There are only two outputs from the e-habitat model — the probability of habitat similarity (PoHS) and the habitat irreplaceability index (HRI) (Table 2). Both of these outputs have *unknown* uncertainties and therefore no further requirements for UncertML can be deduced, although it seems likely that these will require realisations for support given the diverse input uncertainties.

## Outputs

Name	Type	Spatial Type	Uncertainty Type	PDF
PoHS	Continuous	Raster	Unknown	Unknown
HRI	Continuous	N/A	Unknown	Unknown

Table 2: Outputs of the e-habitat model (deliverable 4.1)

## Inputs

Name	Type	Spatial Type	Uncertainty Type	PDF
IACS	Categorical, Discrete & Continuous	Vector	Certain	N/A
Geophysical	Categorical & Continuous	Vector	Certain	N/A
CurWeather	Continuous	Raster	Unknown	Unknown
rules	Categorical	N/A	Unknown	N/A

Table 3: Inputs to the LCCS model (deliverable 5.1)

## 2.2 Deliverable 5.1

Deliverable 5.1 documents the models to be considered in work package 5. This includes a description of each model’s inputs and outputs, the computational complexity of the model and how the models link together. The interaction of socio-economic and policy models with the dynamic land-use model is of greatest importance. Work package 5 contains four models: the Land Capability Classification System, the Markovian Crop Allocation Mechanism, the Field Use Simulator and the YIELD models. The description of these models, including any uncertain inputs and outputs are discussed in the sections below.

### 2.2.1 Land Capability Classification System (LCCS)

The LCCS model takes climate data (either current or some simulated future climate scenario gained from the climate model), geophysical data (such as soil type and field drainage) and the Integrated Administration and Control System (IACS) historical crop rotation data for the fields in the region under study to group the field parcels by land capability and hence determine which types of crops can be grown in each field.

The model outputs the historical crop rotation data in sets corresponding to the different land capabilities of interest, which are chosen by the user.

Table 3 lists the inputs to the LCCS model. The first two inputs (IACS and Geophysical) are considered ‘certain’ inputs, i.e. they contain no uncertainty. The uncertainty of the CurWeather input is unknown so we cannot infer any further requirements, although typically this would be represented statistically. The rules input contains a set of user chosen rules, subsequently they have no spatial context. The only uncertainty associated with this input comes from the possible confusion of the user over the classification rules.

There are three outputs from the LCCS model, listed in Table 4. The LCdatasets output contains the crop rotation data corresponding to the different land capabilities under consideration. The uncertainty of this output is unknown. A secondary output of the LCCS model is the LCclass output, containing a list of the land classifications that correspond to

## Outputs

Name	Type	Spatial Type	Uncertainty Type	PDF
LCdatasets	Continuous, Discrete & Categorical	Vector	Unknown	N/A
LCclass	Categorical	N/A	N/A	N/A
LCfieldalloc	Categorical	Vector	Unknown	N/A

Table 4: Outputs from the LCCS model (deliverable 5.1)

## Inputs

Name	Type	Spatial Type	Uncertainty Type	PDF
LCdatasets	Continuous, Discrete & Categorical	Vector	Unknown	N/A

Table 5: Inputs to the Markovian Crop Allocation Mechanism (CAM) model (deliverable 5.1)

the sets of data in the `LCdatasets` output. These classifications are fixed and therefore do not contain any uncertainty. The final output of the LCCS model is the `LCfieldalloc` output. This contains a table of the IACS reference number and the allocated land capability classification corresponding to that field. The uncertainty of the `LCfieldalloc` is currently unknown, but it is logical that realisations will be the only plausible representation for all these outputs.

We cannot infer any further requirements for UncertML from the outputs of the LCCS model.

### 2.2.2 Markovian Crop Allocation Mechanism (CAM)

The Markovian Crop Allocation Mechanism (CAM) model uses the historical crop rotation data grouped by land capability, and possible economic constraints, to determine a matrix of transition probabilities for crop rotation under each land capability, for the field parcels in an area under study. It is a statistical methodology, based on transition matrices in a Markov chain. Different climate and economic scenarios can affect the model.

There is only a single input to the CAM model, the `LCdatasets`, which were outputted from the LCCS model (Table 4). As stated in Section 2.2.1, the uncertainty of this input is unknown and therefore no further requirements were deduced.

There is also only a single output from the CAM model, the Crop Transition Matrices (CTM). This output is a list containing crop transition matrices for each of the land capabilities. For each set of data in `LCdatasets`, and therefore each land capability in `LCCS:LCclass`, CTM holds a corresponding matrix of transition probabilities for crop rotation – for moving from the crop type of the row to the crop type of the column. The uncertainty of this output is currently unknown but it is anticipated that the data may be modelled using a set of Dirichlet distributions.

**FTR7: UncertML should provide the necessary type to encode a Dirichlet distribution.**

## Outputs

Name	Type	Spatial Type	Uncertainty Type	PDF
CTM	Continuous	N/A	Unknown	Dirichlet

Table 6: Outputs of the Markovian Crop Allocation Mechanism (CAM) model (deliverable 5.1)

## Inputs

Name	Type	Spatial Type	Uncertainty Type	PDF
Fclass	Categorical	Vector	Unknown	N/A
Tmats	Continuous	N/A	Unknown	Dirichlet
Inicrop	Categorical	Vector	Unknown	Unknown

Table 7: Inputs for the Land Use Simulator (LandSFACTS) model (deliverable 5.1)

### 2.2.3 Field Use Simulator (LandSFACTS)

The Field Use Simulator (LandSFACTS) model takes information about possible crop rotations, land capability, current field usage and temporal/spatial constraints on cropping and simulates cropping patterns across each field for a given number of years. This is achieved through stochastic simulation of field usage using different transition matrices for different land capabilities (all within the specified constraints).

The LandSFACTS model has three possible uncertain inputs: **Fclass**, **Tmats** and **Inicrop**. **Fclass** input describes the land capability of each field parcel within the region of interest, it corresponds to the **LCfieldalloc** output of the LCCS model (Table 4). The uncertainty of the **Fclass** input is currently unknown.

The second input of the LandSFACTS model is the **Tmats** input. This input refers to a set of transition matrices that govern the stochastic part of the crop allocation within the model. There is a set of matrices as a matrix is needed for each class of field. The **Tmats** input corresponds to the **CTM** output of the CAM model (Table 6). The requirement of a Dirichlet distribution has already been addressed in as **FTR7** of Section 2.2.2.

The final input of the LandSFACTS model is a list of crops that are present in the field at the beginning of the simulation period: **Inicrop**. The input comes from the IACS crop rotation data and as such is regarded as being certainty.

The LandSFACTS model produces three outputs: **FArea**, **CropArea** and **SimRot**. **FArea** is the area of each field and is considered as a ‘certain’ input. The other two outputs, **CropArea** and **SimRot**, represent the area of land used for each crop, along with the proportion of land for each crop and a cropping choice for each field for every simulated year, respectively. Both of these inputs do not quantify the uncertainty explicitly, however, they are both returned as a series of *realisations*, which quantifies the uncertainty implicitly. The inclusion of realisations within UncertML has already been covered in Section 2.1.1, and recorded as **FTR3**.

### 2.2.4 YIELD

The YIELD model takes the simulated area for each crop from the output of the LandSFACTS field use simulator model, **CropArea**, and multiplies this by the average expected

### Outputs

Name	Type	Spatial Type	Uncertainty Type	PDF
FArea	Continuous	N/A	Certain	N/A
CropArea	Continuous	Vector	Realisations	N/A
SimRot	Categorical	Vector	Realisations	N/A

Table 8: Outputs of the Land Use Simulator (LandSFACTS) model (deliverable 5.1)

### Inputs

Name	Type	Spatial Type	Uncertainty Type	PDF
CropAreas	Continuous	Vector	Realisations	N/A

Table 9: Inputs to the YIELD model (deliverable 5.1)

yield from the expected yield data set.

The solitary uncertain input to the YIELD model is the **CropAreas** input (Table 9). This input contains the area of land that has been allocated to each of the 15 aggregated crop (land use) types, for each year of the simulation from the LandSFACTS Model. This input corresponds to the **CropArea** output of the LandSFACTS model. The uncertainty of this is implied implicitly as a series of realisations, the inclusion of which has already been discussed.

The solitary output of the YIELD model is the **RegCropEsts** which is an estimate of the production / yield of each crop over the region of interest, for each simulated year from the field use simulator model, LandSFACTS. The uncertainty of this output is currently unknown.

## 2.3 Deliverable 6.1

Deliverable 6.1 interacts directly with WP1-3 and WP8 in order to effectively implement the protocols and standards for use in the uncertainty-enabled model Web. Current GMES services will be the main source of data for driving the ensemble forecasts. These source data will include a) ensemble weather forecasts available from ECMWF through GMES; b) ensemble regional scale air quality forecasts available from 7 models through the GEMS project; c) local near-real time monitoring data, currently available at NILU. The output of the local scale ensemble air quality forecasts will also be addressed using the same standards and methodologies.

Work package 6 contains two models: The Air Pollution Model (TAPM) and an urban and local scale air quality model (EPISODE). A description of these models and their uncertain inputs/outputs is given in the following sections.

### Outputs

Name	Type	Spatial Type	Uncertainty Type	PDF
RegCropEsts	Continuous	Vector	Unknown	N/A

Table 10: Outputs from the YIELD model (deliverable 5.1)

### Inputs

Name	Type	Spatial Type	Uncertainty Type	PDF
ECMWF ENSEMBLE	Continuous	Raster	Ensemble	Unknown
SURFACE PARAMETERS	Continuous	Raster	Unknown	N/A

Table 11: Inputs to the TAPM model (deliverable 6.1)

### Outputs

Name	Type	Spatial Type	Uncertainty Type	PDF
ENSEMBLE OUTPUT	Continuous	Raster	Ensemble	N/A

Table 12: Outputs of the TAPM model (deliverable 6.1)

## 2.3.1 TAPM

The TAPM model is used as a meteorological model. It takes synoptic analysis at 125 -50 km resolution and nests these down (3-4 nestings) to 1 km resolution in the area of interest. There are two inputs to the TAPM model: **ECMWF ENSEMBLE** and **SURFACE PARAMETERS** (Table 11).

The inputs to the TAPM model are relatively complex. The **ECMWF ENSEMBLE** input includes wind, temperature, water vapour and pressure as 3D fields. The data ranges from ‘quite’ to ‘very’ uncertain, yet these uncertainties are not currently quantified by their complete probability distributions. The uncertainty is currently implicit within a series of ensembles, as is very common in applications where the models are very complex but also very large in terms of the numbers of variables that need to be considered, for example as seen in weather forecasting.

**FTR8: UncertML should provide a type to represent Ensembles.**

The **SURFACE PARAMETERS** input is a number of parameters linked to the land use data which need to be provided as initial conditions to the TAPM model before a calculation can be made. These parameters may include soil moisture content, soil temperature, sea surface temperature and snow and ice cover. The uncertainty for this input is unknown, however, it can mainly be considered a ‘certain’ input.

The TAPM model has a single output: **ENSEMBLE OUTPUT**. Like the **ECMWF ENSEMBLE** input, the **ENSEMBLE OUTPUT** is complex in nature: an ensemble of hourly 3D meteorological fields for the 2 day forecast period. The uncertainty of the TAPM output is represented within the ensemble, this has been discussed previously.

## 2.3.2 EPISODE

EPISODE is an air quality model that takes in gridded meteorological data (in this case from TAPM) and combines these with a range of emissions sources and models to produce concentrations in fields and at ‘receptor’ points. In this application the 2 models are used together. Hourly meteorological fields from TAPM are given to EPISODE and hourly mean concentrations are calculated.

### Inputs

Name	Type	Spatial Type	Uncertainty Type	PDF
REGIONAL BACKGROUND	Continuous	Raster	Ensemble	N/A

Table 13: Inputs of the EPISODE model (deliverable 6.1)

### Outputs

Name	Type	Spatial Type	Uncertainty Type	PDF
ENSEMBLE OUTPUT	Continuous	Raster	Ensemble	N/A

Table 14: Outputs of the EPISODE model (deliverable 6.1)

The REGIONAL BACKGROUND input, listed in Table 13 provides boundary conditions of concentrations for the EPISODE model. This is typically an ensemble of 7-10 model calculations. The uncertainty of this input is characterised within the ensemble, as discussed in the delineation of requirement **FTR8**.

The output of the EPISODE model is also an ensemble, ENSEMBLE OUTPUT (Table 14). The uncertainty of this output is also characterised via the ensemble and is not fully quantified by a probability distribution. This reinforces the importance of requirement **FTR8**.

## 2.4 Deliverable 7.1

Deliverable 7.1 describes the Albatross model, and associated transport and emissions models. The core model is the very complex Albatross model, an agent based simulator of individual travel behaviour. The core aim of this work package is to produce a simulation environment for developing emissions scenarios and examining the factors that critically affect emissions.

### 2.4.1 Albatross

The Albatross model is based on the learning of decision trees for individual activity behaviours from activity diaries. These diaries also link to socio-economic data about the individuals and their households and by learning the decision trees for groups of the population an agent based simulation environment is created that can be used to simulate the activities of individuals within a given geographic region.

The inputs to the Albatross model are very complex. For the purposes of UncertWeb the

### Inputs

Name	Type	Spatial Type	Uncertainty Type	PDF
Socio-economic data	Categorical	Vector	Unknown	N/A
Activity diary data	Categorical	Vector	Unknown	N/A
Physical data	Categorical and Continuous	Vector	Unknown	N/A

Table 15: Inputs to the Albatross model (deliverable 7.1)

Outputs				
Name	Type	Spatial Type	Uncertainty Type	PDF
Time dependent activity schedule	Continuous	Space-time trajectory	Confusion Matrix	N/A

Table 16: Outputs from the Albatross model (deliverable 7.1)

activity diaries will be considered certain, although of course they only represent a sample of the underlying population. Socio-economic data is uncertain due to finite sample size effects to again the ability to represent standard errors as given in **FTR6** is important. The physical data, which includes things such as office hours, locations of places of employment is considered largely certain. The main uncertainty arises in the travel times within the transport network, which are likely to be represented by statistics of the observed travel times.

The main outputs of the Albatross mode will be a realisation of space-time trajectories for a collection of individuals. This will again require the encoding of realisations.

## 2.5 Additional Type Requirements

Deliverable 8.1 defines a series of requirements for the UncertWeb model architecture. Section 2.1.3 of deliverable 8.1 outlines a list of uncertainty types that are needed within the framework. While most of these requirements have been captured from deliverables 4.1–7.1, some have not been included; for example UncertML should be capable of describing a confusion matrix and a covariance matrix.

Sections 2.1–2.4 provided a definite list of requirements that UncertML should achieve to satisfy all current use cases within the UncertWeb project. However, it is clear from the summary tables that that the uncertainty on many model inputs and outputs is currently unknown, and that as the work evolves, the process of elicitation within the course of the project may generate uncertainty representations of types which have not yet been specifically identified. In addition, we must consider that UncertML exists within a broader spectrum than UncertWeb and should provide statistics and distributions that satisfy a wider range of users.

**FTR9: UncertML should encode common statistics and distributions sufficient to allow the handling of all reasonable sources of uncertainty which will arise during the course of the project.**

While it is important that this wide range of representations should be available, there is a smaller subset which will satisfy the requirements for the majority of use cases. In this sense UncertML has similarities to GML and other schemas which can be unwieldy and intimidating when viewed as a whole. In order to make the UncertML vocabulary and encodings as usable as possible across a wide range of contexts, and to encourage their uptake by novice users, we propose to separate out a self-contained 'core' of types which can be documented and used in isolation. This can be seen as a non-functional requirement of UncertML.

**NFR2: UncertML should be split into a ‘core’ and ‘extension’ model with the common statistics and distributions implemented in the core and the less common implemented as an extension.**

In Section 2.5.1 we identify a list of commonly used statistics within various scientific domains that UncertML should include. A brief description of each statistic is provided. Section 2.5.2 details a list of commonly used probability distributions that should be included within UncertML. Any statistic or distribution that has been explicitly mentioned in Section 2 is *not* listed below. These core statistics are summarised as **FTR11** in the Conclusions.

### 2.5.1 Common statistics

Below is a list of commonly used statistics that should be incorporated into UncertML.

**Mean** — the expected value of a random variable, which is also called the population mean.

**Mode** — the value that occurs the most frequently in a data set or a probability distribution.

**Median** — the numeric value separating the higher half of a sample, a population, or a probability distribution, from the lower half

**Quantile** — points taken at regular intervals from the cumulative distribution function (CDF) of a random variable.

**Skewness** — a measure of the asymmetry of the probability distribution of a real-valued random variable.

**Confidence interval** — an interval and level interval which is used to express the likelihood that a value falls within a specific range.

A number of other statistics exist that are deemed less common, but should still be included within UncertML. These statistics should be included within an ‘extension’ and should not dilute the core UncertML model. They are listed below, and are summarised as **FTR11** in the Conclusions:

**Credible interval** — a posterior probability interval which is used for interval estimation in contrast to point estimation.

**Correlation** — a measure of statistical relationship between two or more random variables.

**Decile** — any of the nine values that divide the sorted data into ten equal parts, so that each part represents  $\frac{1}{10}$  of the sample or population.

**Quartile** — one of four equal groups, representing a fourth of the distributed sampled population.

**Inter-quartile range** — a measure of statistical dispersion, being equal to the difference between the third and first quartiles.

**Kurtosis** — a measure of the ‘peakedness’ of the probability distribution of a real-valued random variable.

**Moment** — a quantitative measure of the shape of a set of points.

**Percentile** — the value of a variable below which a certain percent of observations fall.

**Range** — the length of the smallest interval which contains all the data.

Further to the inclusion of the above listed statistics, UncertML should provide a mechanism for grouping statistics into meaningful summaries. For example, a typical use case would be to provide the mean and variance of a dataset — UncertML should facilitate this.

**FTR12: UncertML should provide a means of grouping statistics into a logical structure.**

### 2.5.2 Common distributions

Below is a list of commonly used distributions. Each has been labelled with the type of data it can be used with (e.g., discrete or continuous). These distributions have been identified as ‘common’ due to their inclusion within several leading statistical applications (e.g., mathematica<sup>3</sup>). These are collected in the Conclusions as **FTR13**.

**Exponential distribution** — continuous.

**Gamma distribution** — continuous.

**Multivariate Gaussian distribution** — continuous.

**Uniform distribution** — categorical, discrete & continuous.

**StudentT distribution** — continuous.

**Log-normal distribution** — continuous.

**Poisson distribution** — discrete.

**Binomial distribution** — discrete.

A further list of less commonly used distributions is provided below. These should be included in the UncertML extension, and are summarised as **FTR14** in the Conclusions.

**Beta** — continuous.

**Laplace** — continuous.

**Cauchy** — continuous.

**Chi-square** — continuous.

**Weibull** — continuous.

**Logistic** — continuous.

---

<sup>3</sup><http://www.wolfram.com/products/mathematica/index.html>

**Geometric** — discrete.

It is highly likely that in the elicitation work of the UncertWeb project, variables will be identified whose distributions are themselves based on uncertain parameters. A typical example would be the case where multiple experts are polled as to their opinions about the average value of a phenomenon and its spread. In this setting the uncertainty on the mean value will depend on the uncertainty on the spread (variance) value). The compound distribution representing the total uncertainty will therefore be a *conditional distribution*.

**FTR15: UncertML should provide a mechanism for quantifying conditional distributions, i.e. distributions where the parameters are uncertain.**

## 2.6 Overarching non-functional requirements

In the process of gathering the model requirements for Sections 2.1–2.4, it became clear that there are clear common requirements for all models which are to be exposed via Web Services and orchestrated into chains. In particular, we identified the need for tight profiling of schemas and services, and for the uncertainty on model inputs and outputs to be clearly identifiable with reference to a single common vocabulary, in order to allow automated chaining and consumption of the models. This issue will be addressed in more detail in the deliverables for work package 2, but it has an important impact on the design of UncertML, since a weak-typed design which allows users to define and reference their own uncertainty types means that extra semantic information is required for the consumption and proper use of those types. By contrast, a strong-typed design, where uncertainty types are characterised by a limited set of specifically defined elements, allows legal input/output types to be more clearly defined and restricted for each model, and facilitates interoperability.

**NFR3: UncertML should use a strong-typed approach, where selected entities are explicitly modelled by the definition of explicit types as appropriate.**

Separation of concerns has always been an important priority for UncertML: version 1.0 was designed to be agnostic to context and to represent only uncertainty information. Typically, the environmental observations involved in use cases will require the recording of units of measurement, descriptions of the phenomena measured or information on the spatio-temporal domains of the measurements. The wide variety of model inputs and outputs documented in Sections 2.1–2.4 demonstrates the variability and complexity of this ancillary information. However, the uncertainties themselves may be numerically identical, and should be handled as such in order for UncertML to focus on effective encoding and parsing of information. For example, consider the probability of landcover misclassification in WP4 (1) and the probability of wrongly-predicted activity as an output of the Albatross model in WP7.

The numerical probability values represent uncertainties with very different contexts and consequences in the real world. While this information can be encoded elsewhere in the document and used to help interpret and use the values, the fact that they are both discrete probabilities gives them some important commonalities. They are both bounded between 0 and 1: they can both be used in simulation or modelling in exactly the same way: and in the case of these two particular values, they are both likely to be components of confusion matrices which impose further restrictions on the groups of values within them. A common

approach which encodes the two uncertainty values in exactly the same way (as DiscreteProbabilities, or, if matrix restrictions apply, as components in a Dirichlet distribution), allows for re-use of tools such as parsers, and recognises important functional similarities in terms of modelling. At the same time, rich information on the context of the numerical information can be communicated using specific schemata such as GML and O&M where appropriate. The profiling necessary to do this effectively is discussed in Section 5 of Deliverable 8.1.

**NFR4: UncertML should remain domain-agnostic and *not* include specific meta-data for concepts such as space, time or phenomena.**

## 2.7 Other non-functional requirements

Sections 2.1–2.5 outlined the specific *types* that UncertML will have to support to satisfy all requirements of the UncertWeb project. However, further non-functional requirements can be inferred from the tables.

The process of gathering requirements for the processing chains within the UncertWeb project has identified the need to encode uncertainty in other formats. The model inputs and outputs listed in Sections 2.1–2.4 cover a wide variety of data formats, across a varying range of spatial domains and extents. The original version of UncertML<sup>4</sup> was envisaged as a single XML schema, specifically designed to efficiently handle *vector* data. However, because of the restrictions of the XML language, it can not handle raster data very effectively. Given the need within UncertWeb to handle large raster grids, UncertML should be provided in a variety of encodings to suit a range of needs. For example, a binary encoding of UncertML should be provided to allow efficient encoding of large, multidimensional raster datasets. In other contexts, UncertML should facilitate fast exchange of lightweight data without compromising its information value. The issue of data formats and domains is discussed in detail in Section 3 of Deliverable 8.1. To facilitate this requirement, UncertML should be split into three distinct parts: a conceptual data model, a dictionary of terms and a number of encodings adhering to the conceptual model and referencing the dictionary.

**NFR5: UncertML should be structured as a vocabulary and conceptual model which will support multiple implementations in different languages.**

**NFR6: UncertML should provide multiple encodings (e.g., XML and binary) to allow for efficient transport of large raster datasets.**

The types of model input and output within work packages 4–7 include categorical, discrete and continuous data, all of which may have attached uncertainty. It is important to specifically consider these measurement scales and the specific effects on the applicability and existence of statistics for each.

**NFR7: UncertML should provide a means to encode all statistics and distributions for categorical, discrete and continuous data, where applicable.**

---

<sup>4</sup><http://www.uncertml.org>

## 3 Summary

The contents of this document identify the requirements for the second iteration of UncertML. Section 2 gave a detailed description of the model requirements listed in deliverables 4.1, 5.1, 6.1 & 7.1. The information gathered from these deliverables resulted in several statistics and distributions that *must* be included within UncertML. Section 2.5 then gave a list of further statistics and distributions that are commonly used and should also be included within UncertML. Meeting these requirements will ensure that UncertML can satisfy all use cases within the UncertWeb project and many outside the project.

### 3.1 Summary of UncertML requirements

A definitive list of requirements for UncertML is summarised below.

#### Functional Type requirements.

- **FTR1:** UncertML should provide a mechanism for quantifying statistics using proportions where applicable.
- **FTR2:** UncertML should include explicit types for recording standard deviation and variance.
- **FTR3:** UncertML should retain the Realisations element type included in version 1.0.
- **FTR4:** UncertML should include an explicit GaussianDistribution type.
- **FTR5:** UncertML should retain the explicit DiscreteProbability statistic type.
- **FTR6:** UncertML should allow the communication of standard errors of mean and variance.
- **FTR7:** UncertML should provide the necessary type to encode a Dirichlet distribution.
- **FTR8:** UncertML should provide a type to represent Ensembles.
- **FTR9:** UncertML should encode common statistics and distributions sufficient to allow the handling of all reasonable sources of uncertainty which will arise during the course of the project - these are listed in FTR11, FTR12, FTR14 and FTR15.
- **FTR10:** The core Statistics in UncertML should be Mean, Mode, Median, Standard Deviation and Variance(FTR2), Quantile (already present in v1.0), Skewness, Probability and Discrete Probability (FTR5)
- **FTR11:** The Statistics included in the extension of UncertML should be Credible Interval, Correlation, Decile, Quartile, Percentile, Inter-quartile Range, Kurtosis, Moment, and Range
- **FTR12:** UncertML should provide a means of grouping statistics into a logical structure.

- **FTR13:** The core Distributions in UncertML should be Gaussian (FTR4), Dirichlet (FTR7), Log-normal, Exponential, Gamma, Multivariate Gaussian, Uniform, Student T, Poisson and Binomial.
- **FTR14:** The Distributions included in the extension of UncertML should be Beta, Laplace, Cauchy, Chi-square, Weibull, Logistic and Geometric.
- **FTR15:** UncertML should provide a mechanism for quantifying conditional distributions, i.e. distributions where the parameters are uncertain.

#### Non-functional Requirements.

- **NFR1:** UncertML will permit the communication of uncertainty information through a standardised and universally-accessible conceptual model.
- **NFR2:** UncertML should be split into a ‘core’ and ‘extension’ model with the common statistics and distributions implemented in the core and the less common implemented as an extension.
- **NFR3:** UncertML should use a strong-typed approach, where selected entities are explicitly modelled by the definition of explicit types as appropriate.
- **NFR4:** UncertML should remain domain-agnostic and *not* include specific metadata for concepts such as space, time or phenomena.
- **NFR5:** UncertML should be structured as a vocabulary and conceptual model which will support multiple implementations in different languages.
- **NFR6:** UncertML should provide multiple encodings (e.g., XML and binary) to allow for efficient transport of large raster datasets.
- **NFR7:** UncertML should provide a means to encode all statistics and distributions for categorical, discrete and continuous data, where applicable.