Open Geospatial Consortium Inc.

Date: 2009-04-08

Reference number of this document: **08-122r2**

Version: **0.6**

Category: OpenGIS® Discussion Paper

Editors: Matthew Williams, Dan Cornford, Lucy Bastin & Edzer Pebesma

Uncertainty Markup Language (UnCertML)

Warning

Document type:	OpenGIS® Discussion Paper
Document subtype:	Encoding
Document stage:	Draft proposed version 0.6
Document language:	English

Copyright

Forward

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. The Open Geospatial Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

Recipients of this document are requested to submit, with their comments, notification of any relevant patent claims or other intellectual property rights of which they may be aware that might be infringed by any implementation of the standard set forth in this document, and to provide supporting documentation.

1 Table of Contents

2 Introduction

Most data contains uncertainty, arising from sources which include measurement error, observation operator error, processing/modelling errors, or corruption. Processing this uncertain data (typically through models, which can introduce their own errors), propagates the uncertainty, often unpredictably. The ability to optimally utilise data requires a description of its uncertainty which is as complete and detailed as possible, and in the geospatial context, this characterisation and quantification is particularly crucial when data is used for spatial decision making. Thus there is a well-recognised need for GIS frameworks which can handle and 'understand' incomplete knowledge in data inputs, in decision rules and in the geometries and attributes modelled. A substantial literature exists on mechanisms for representing and encoding geospatial uncertainty and its propagation. However, no framework yet exists to describe and communicate uncertainty (either in GI data or more generally) in an *interoperable* manner.

UncertML is an XML schema for describing uncertain information, which is capable of describing a range of uncertain quantities. Its descriptive capabilities range from summaries, such as simple statistics (e.g. the mean and variance of an observation), to more complex representations such as parametric distributions at each point of a regular grid, or even jointly over the entire grid.

The ISO/IEC guide to the expression of uncertainty in measurement (GUM) outlines the importance of quantifying uncertainty by stating that it is "obligatory that some quantitative indication of the quality of the result be given so that those who use it can assess its reliability" when discussing observations. The guide goes on to state that it is necessary to have a readily implemented and generally accepted procedure for characterizing the quality of a result of a measurement; however, it does not outline a mechanism for describing this information via an exchangeable medium. The GUM guide aspires to provide a worldwide consensus on the evaluation and expression of uncertainty in measurement, not dissimilar to the International System of Units. With the development of the UncertML standard and accompanying dictionary (http://dictionary.uncertml.org) a small step has been taken toward this vision.

Recent developments for sensor observation modeling within the Open Geospatial Consortium (e.g. the Observations & Measurements standard) have opened opportunities for interoperable, sensor-derived datasets to be exchanged over the Internet. As this Sensor-Web community grows, an increasing volume of data will become available and require processing, and much of this data will be used for decision support. However, rational decision-making using incomplete knowledge (i.e. sensor measurements) is only possible if we can quantify the uncertainty inherent in those measurements, and the uncertainty that is introduced or increased by subsequent processing. To be truly valuable in the context of 'discoverable' Web Services and datasets (for example, within automatic online risk management chains), this uncertainty must be represented in an interoperable manner. Currently, within the Sensor-Web framework no formal method of quantifying complex uncertainties (e.g. probabilistic representations) exists.

Figure 1 illustrates the importance of both quantifying and communicating uncertainty. The scenario is a 1D interpolation of sensor data from two distinct sensor types. One sensor model has Gaussian noise while the other has exponential noise. The graph on the left displays the result of an interpolation (performed, in this example, by a Web Processing Service) when the uncertainty is not explicitly quantified. In such a scenario the noise is assumed to be Gaussian for all observations. The prediction is particularly poor in the mid-section where the observations shown as plusses are actually subject to exponential noise, as opposed to the assumed Gaussian noise. When the uncertainty is fully described (in this case using UncertML) the interpolation algorithm is able to process this information and thus produce a more informed prediction, as depicted in the right-hand graph. In addition, fully quantified and characterized estimates of uncertainty can be returned as part of the interpolation result, in a format which can be automatically parsed and 'understood' by, for example, a Web Service application in a processing chain.

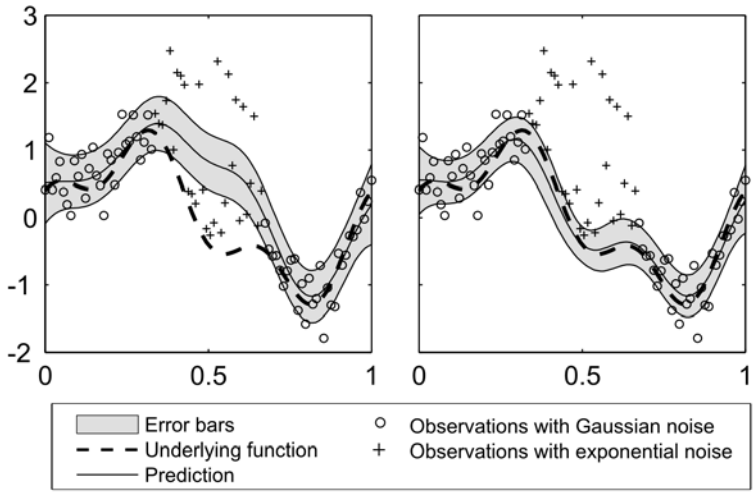

Figure 1: Without specific error information on individual measurements, an automated Bayesian interpolation algorithm is forced to assume Gaussian noise on all measurements, and thus achieves a bad estimate of the true environmental state. When observation-specific error characteristics are supplied via UncertML, the performance of the automated interpolator is much improved.

3 Scope

The Uncertainty Markup Language (UncertML) is an XML encoding for the transport and storage of information about uncertain quantities, with emphasis on quantitative representations based on probability theory.

This document describes the XML schema syntax and conventions that allow an interoperable description of uncertain data, which we define to be random quantities, in a variety of ways including:

- probability distributions including both uni- and multi-variate distributions and mixture models;
- statistics, including means, (co-)variances, standard deviations and quantiles;
- realisations or sampled data.

These three categories, illustrated in Figure 2, cover the full range of representations which one might commonly use for random quantities. The most precise description of a random quantity is in terms of its probability distribution, which is appropriate where the distributional form of that random quantity is known. Where a strong parametric form for the distribution is not appropriate, a more flexible semi-parametric mixture model may be used. A weaker, but often more realistic, option which is still useful and is widely employed, is to represent a random quantity in terms of its statistics. We allow for a range of statistics types, ranging from moments (e.g. mean and variance) to histogram and quantile based representations. Finally we also allow for fully non-parametric representations in terms of samples / realisations from the given distribution, such as might arise from a Bayesian Markov chain Monte Carlo analysis.

All types are designed to allow encoding of both uni- and multi-variate uncertainty, and can thus be used for specification of marginal and joint distributions. Both continuous and discrete random quantities are catered for.

There is a clear separation of concerns in the design of UncertML, in that it is **not** designed to address issues covered in other schemata. For example, there is no notion of units of measure in UncertML – the intention is that UncertML is used with other schema, and essentially replaces primitive value types in scalar and vector form. Illustrative examples later in this document will show how UncertML can be flexibly and easily combined with other XML schemata. In the context of this discussion, Geography Markup Language, SWE Common and Observations & Measurements are used to illustrate its application, but the design of UncertML means that it could as easily be used to describe uncertainty in datasets from other fields of research; for example, genetics, economics or linguistics.

UncertML is, at present, restricted to **probabilistic** representations of uncertainty in random quantities, and does not address concepts such as fuzzy sets, random processes or belief functions.

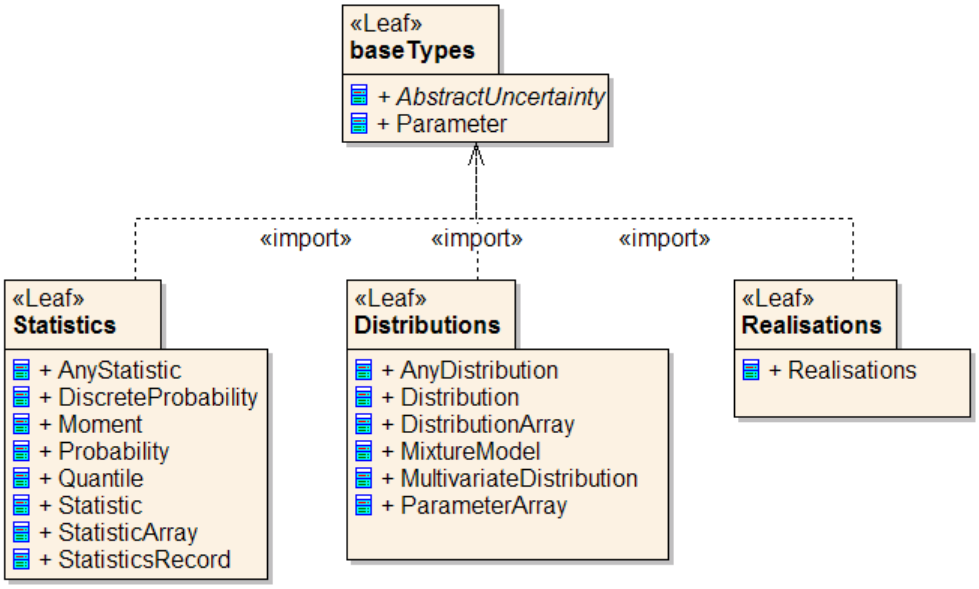

Figure 2: An overview of the UncertML package dependencies.

4 Normative references

ISO 19138: Geographic information – Data quality measures
ISO/IEC GUIDE 98-3: Uncertainty of measurement – Part 3: Guide to the expression of uncertainty in measurement (GUM:1995)
Geography Markup Language
Observations & Measurements
Sensor Model Language (SensorML)
Sensor Web Enablement Common (SWE Common)
W3C XLink, XML Linking Language (XLink) Version 1.0. W3C Recommendation (27 June 2001)
W3C XML, Extensible Markup Language (XML) 1.0 (Second Edition), W3C Recommendation (6 October 2000)
W3C XML Namespaces, Namespaces in XML. W3C Recommendation (14 January 1999)
W3C XML Schema Part 1, XML Schema Part 1: Structures. W3C Recommendation (2 May 2001)
W3C XML Schema Part 2, XML Schema Part 2: Datatypes. W3C Recommendation (2 May 2001)

5 Conventions

5.1 Symbols (and abbreviated terms)

GML	Geography Markup Language
SWE	Sensor Web Enablement
UML	Unified Modelling Language
UncertML	Uncertainty Markup Language
URI	Uniform Resource Identifier
XML	eXtensible Markup Language

5.2 UML Notation

The diagrams that appear in this document are presented using the Unified Modelling Language (UML) static structure diagram. The UML notations used in this document are described in the diagram below.

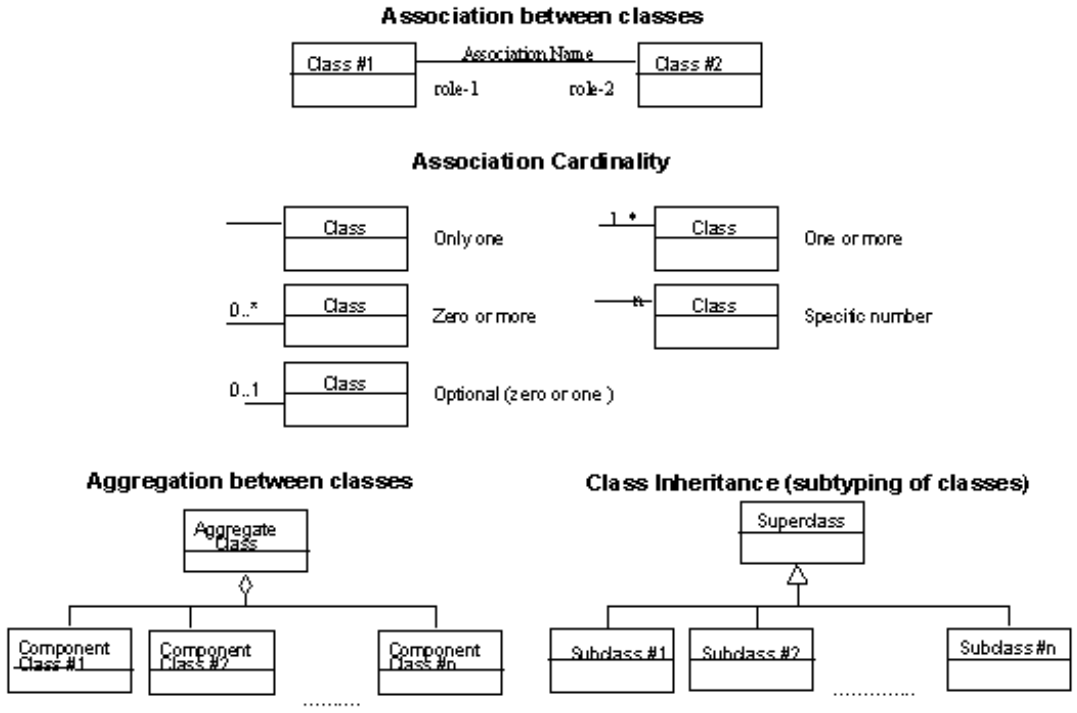

Figure 3: UML Notation.

In this document, the following three stereotypes of UML classes are used:

a) <<DataType>> is a set of properties that lack identity (independent existence and the possibility of side effects). A DataType is a class with no operations whose primary purpose is to hold the information.
b) <<Union>> is a set of properties. Semantic constraints ensure that only one of the properties may be present at any time.
c) <<Abstract>> is an abstract object type (the stereotype is used in addition to formatting the class name in italics).

In this document the following standard data types are used:

a) Double – a double precision floating point number
b) Integer – an integer number

5.3 Definitions of terms as used within this document

'Domain point': a uniquely-identifiable sampling location within a set, which will often, but not always, be distinguished by its location in space and/or time.

'Random quantity': a quantitative result that is not known with certainty. We do not debate the philosophical or technical questions that this might introduce here.

'Random variable': a random quantity that is attached to a specific variable or outcome, that is has units of measure and often a real physical interpretation. Note again this is not the precise mathematical definition.

'Realisation': one of many possible values derived by sampling or simulation from a probability density function.

6 Conceptual Models

This section provides a detailed conceptual model for all types in UncertML. Diagrams depicting all types and their properties are provided in UML notation (outlined in Section 5.2).

6.1 Base Types

Base types are types common to all UncertML types. Currently two such types exist in UncertML - the `AbstractUncertainty` type and the `Parameter` type discussed below.

6.1.1 AbstractUncertainty

The `AbstractUncertainty` type provides a root for the substitutability chain in UncertML – all common uncertainty types extend this base type and inherit any common properties.

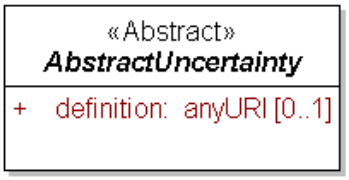

Figure 4: AbstractUncertainty type is the head of the UncertML substitutability chain.

Figure 4 displays the `AbstractUncertainty` type with a single property, `definition`. The `definition` property provides a link, through the use of URIs, from any uncertainty type (for example, 'distribution') to a definition describing the uncertainty type of interest (for example, 'Gaussian').

UncertML follows a 'soft-typed' design pattern which promotes the use of generically named elements with links to dictionary definitions to provide semantics. This approach is far more flexible than the more traditional 'strong-typed' design which has a unique element for each type (or feature) within the application domain. However, crucial to the use of a weak-typed design is the implementation of a dictionary describing the concepts referenced from within the schema. We are currently developing a dictionary, in parallel with the UncertML schemata, which describes the most common statistics, distributions & other related concepts, using mathematical formulae where appropriate.

The design and implementation of this dictionary is fundamental to the successful adoption and use of UncertML. Based loosely around the structure of the data quality measures outlined in ISO 19138, the definitions are encoded according to the GML dictionary schema. Included within the definition for each statistic, distribution and all other concepts are the following pieces of information:

- Name(s)
- Definition
- Description
- Measure type (integer, measure etc)
- Measure structure (list, matrix etc)

The root of the dictionary is located at http://dictionary.uncertml.org and follows a pseudo-RESTful design pattern. All statistics are located at '/statistics' with the name of the statistic following - e.g. the definition of the statistic 'mean' would be located at the following URL: http://dictionary.uncertml.org/statistics/mean. If the specified statistic has any parameters, these can be found at: http://dictionary.uncertml.org/statistics/<statistic name>/parameters. Individual parameter information can be located at the following: http://dictionary.uncertml.org/statistics/<statistic name>/parameters/<parameter name>. The same structure is applied to all parametric distributions, with the word 'distributions' substituted for 'statistics'.

An XSLT stylesheet exists to allow a 'human-readable' view of the entire dictionary and allows easy navigation through the various statistics and distributions.

6.1.2 Parameter

The second base type in UncertML is the `Parameter`, this type is common to all parametric distributions and certain statistics.

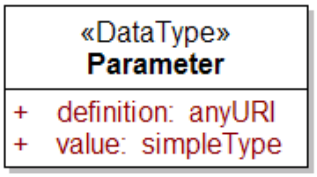

Figure 5: The Parameter type is common to distributions and certain statistics.

Displayed in Figure 5, the `Parameter` type contains two properties: `definition` and `value`. The `definition` of a `Parameter` is identical to that of the `AbstractUncertaintyType` and references a dictionary definition of a particular statistic (or distribution) parameter. The `value` of a `Parameter` contains any simple XML type, typically an integer or double value.

6.2 Realisations

In some situations, a user may not be able to simply represent the uncertainties of the data they are working with. In such a situation, a sample from the random quantity might be provided, allowing uncertainty to be described implicitly. However, when using this approach, a sufficiently large sample is required to deduce the uncertainties inherent in the data, which means that efficient encapsulation of large data volumes is an important issue for UncertML. The following sections discuss the `Realisations` type available within UncertML for describing a sample of data through a series of realisations.

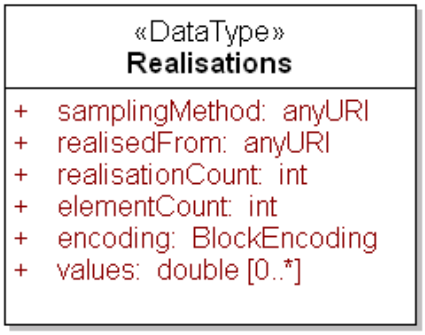

Figure 6: Structure of a Realisations type in UncertML .

Extending the `AbstractUncertainty` type discussed in Section 6.1.1 provides a `definition` property. In this instance the URI, if required, should resolve to a definition of the concept of a realisation. Two optional properties are included to provide greater information about any particular set of realisations. The `realisedFrom` property is a URI that links to a definition of the distribution from which the realisations are generated, where this can be provided. The second property, `samplingMethod`, is a URI resolving to the definition of the particular method that was used to sample the realisations. The `realisationCount` contains the number of realisations in each sample; this information is useful when describing multiple random quantities at multiple domain points. 'Domain point' here refers to a unique sampling location in a simulation series, which will often, but not always, be distinguished by its location in space and/or time.

As with all other array types in UncertML, the `Realisations` type is based around the SWE Common `DataArray` type (1). However, as the `Realisations` type can only describe a series of realisations, the `elementType` property of the `DataArray` is not used. The `elementCount` property is used to indicate the *total* number of values contained within the array. In cases where a dataset describes a single variable at a single domain point, this value will be the same as the `realisationsCount` property. When describing multiple variables and/or or multiple domain points, the size of the `elementCount` will be the product of the number of variables, number of domain points and the number of individual realisations. More information about how to decode the information within the array may be found in Section 8.

The last two properties are directly inherited from the SWE Common encoding schema, which provides an efficient and flexible solution to encoding data arrays. Loosely speaking, the format of the data (binary, ASCII, XML etc) is described in the `encoding` property and the `values` property contains the data which relates to the `elementType`; i.e. the actual values realised through sampling. More information about the SWE Common encoding schema can be found in (1) and example encodings are detailed in Section 8.

Aggregate types within UncertML, whether they be arrays or records, do not extend the `AbstractUncertainty` type, as they are merely perceived as 'containers' for uncertainty types with each individual constituent containing its own definition.

6.3 Statistics

This section discusses the extensive range of options available in UncertML for describing 'summary statistics'. Such statistics are used to provide a summary of a random quantity,

ranging from measures of centrality (mean, mode, median, etc) through measures of dispersion (range, standard deviation, variance etc.) to higher order moments, such as skewness and kurtosis. While certain statistics (e.g. mean, mode) do not provide any information about uncertainty in isolation, they are often used in conjunction with other statistics (e.g. variance, standard deviation) to provide a concise summary, and there is considerable value in the explicit information that a given value represents the mean, rather than some other statistic such as the mode, or even a single realisation.

6.3.1 Summary Statistics

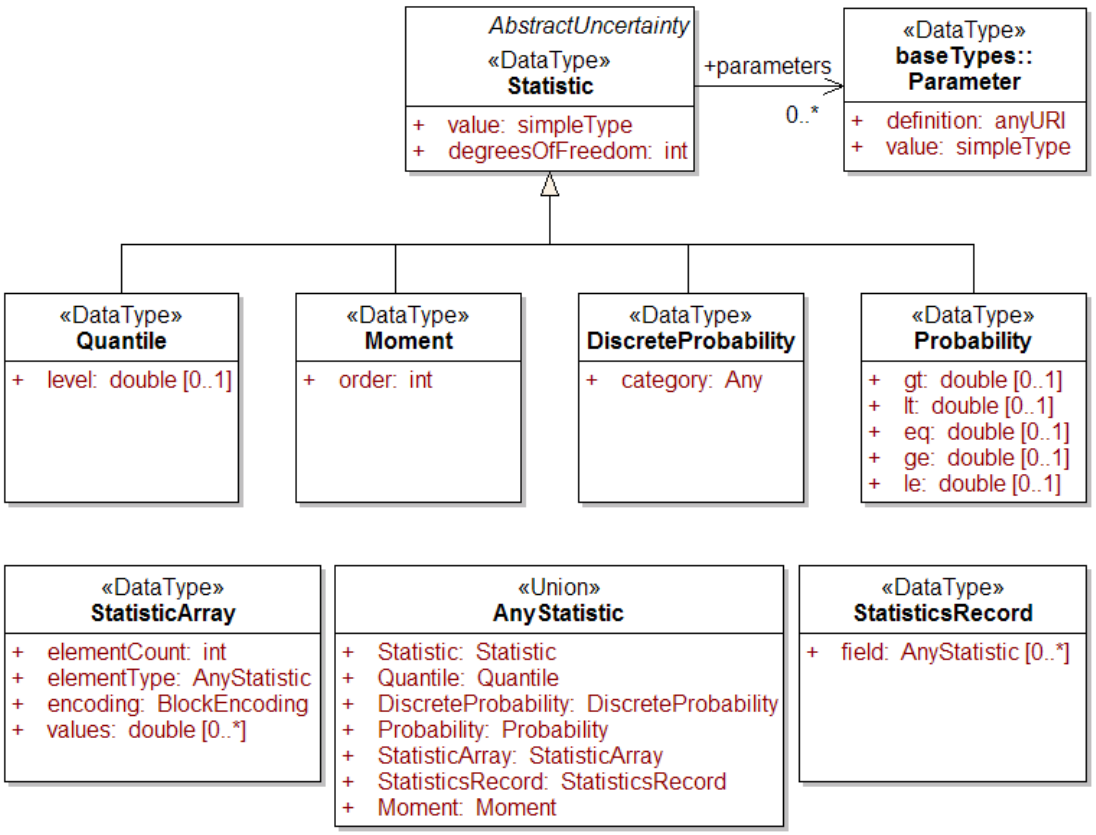

Figure 7: UncertML provides a suite of data types for describing various statistics including quantiles, probabilities and general statistics such as mean and variance.

Figure 7 displays the entire collection of types available within UncertML for describing statistics. The `Statistic` type is the simplest method of describing a statistic. Extending the `AbstractUncertainty` type provides a `definition` property which in this instance should resolve to a definition of the particular statistic, e.g. mean, variance, mode etc. Every `Statistic` type has a `value` property containing the actual value of the statistic, which is encoded as any simple XML type but would typically be an integer or double. When using statistics to describe a dataset it is often valuable to know the effective degrees of freedom, in order to assess the power and appropriate uses of the statistics. The base `Statistic` type in UncertML provides an optional attribute, `degreesOfFreedom`, which may be used for this purpose. This generic and concise concept of a statistic allows *most* statistics to be encoded, but for certain statistics more information is required. In such cases the `Statistic` type contains one or more `Parameter` types (Section 6.1.2), each with a `definition` and `value`.

While the Statistic type is capable of describing all common statistics, we feel that certain statistics are used so frequently as to warrant their own type. One such example is a quantile. When working with quantiles, a user needs to know which quantile is being referred to. UncertML therefore provides a specific `Quantile` type which extends the `Statistic` type and provides an additional property, `level`. Within UncertML we specify the particular quantile by its level. For example, the 0.3 quantile (the value of the random quantity below

which a proportion of 0.3 of the probability mass, which always sums to 1.0, lies) has a `level` property of value 0.3.

Another frequently-used statistic is the probability value, used to frame questions such as: "What is the probability that the water level at a specific location will exceed 25m?", "What is the probability that the temperature will fall below 24° Celsius?", "What is the probability that this tree is an Oak tree?". For such instances UncertML provides two further types, `DiscreteProbability` and `Probability`, which both extend the `Statistic` type. `DiscreteProbability` is used to describe the probability that a variable falls within a certain enumerated class, as in the third example above. The addition of a `category` property provides the ability to specify any data type as a class but will typically be realised as a string; in the example above the `category` would be "Oak".

`Probability` may be used to describe the probability that a variable exceeds (or does not exceed) a certain threshold. Such thresholds are defined through the use of the `gt` (greater than), `lt` (less than), `eq` (equal to), `ge` (greater than or equal to) and `le` (less than or equal to) properties, which may be used either individually or in combination. The `value` property of both the `DiscreteProbability` and `Probability` types always contains a value restricted to fall between 0.0 and 1.0, in contrast to the other statistic types, which contain a value in the relevant scale of the random quantity.

The final statistic available within UncertML is the `Moment` type. A random quantity is often described by a collection of its (centered) moments. A `Moment` extends the `Statistic` type by adding an `order` property which contains the order of the moment as an integer; for example '3' for the 3rd order centered moment often referred to as the skewness.

From this point, when referring to 'statistics' we mean the generic `Statistic` type as well as all children (`Quantile`, `DiscreteProbability`, `Probability` and `Moment`).

6.3.2 StatisticsRecord

Grouping of statistics provides a mechanism by which a random quantity can be summarised in terms of its centrality and dispersion or other attributes, UncertML provides the `StatisticsRecord` type for such use cases. As with all 'record' types within UncertML, the `StatisticsRecord` is closely modelled on the SWE Common `DataRecord` type (1).

A `DataRecord` type in SWE Common, and therefore a `StatisticsRecord` type in UncertML, consists of a number of `field` properties. Each `field` of a `StatisticsRecord` may be any type that belongs to the `AnyStatistic` union: `Statistic`, `Quantile`, `Moment`, `DiscreteProbability`, `Probability`, `StatisticsArray` or `StatisticsRecord`. A combination of general statistics in a `StatisticsRecord` provides a clearly-structured set of summary statistics for a given variable. The ability to construct complex structures such as records of arrays and records of records allows users to construct complex representations of uncertainty with relative ease.

6.3.3 StatisticsArray

Arrays of statistics are useful when describing a variable at several domain points, or several variables at a given domain point. The `StatisticsArray` type in UncertML, closely modelled on the `DataArray` of SWE Common, provides such a mechanism. The `elementType` property of a `StatisticsArray` may be any type from within the `AnyStatistic` union, allowing multiple summaries to be encoded flexibly as arrays of single statistics, or an array of `StatisticsRecord`s. More complex structures, such as two dimensional arrays, are also possible. Exploiting the efficiency of the SWE Common encoding schema means that complex structures can be encoded in the best possible manner.

6.4 Distributions

When the uncertainties of a data set are better understood, it may be desirable to describe them through the use of probability distributions. The types contained within this section of UncertML are specifically designed to allow a concise parametric or semi-parametric encapsulation of *all* distributions without sacrificing the simplicity of UncertML.

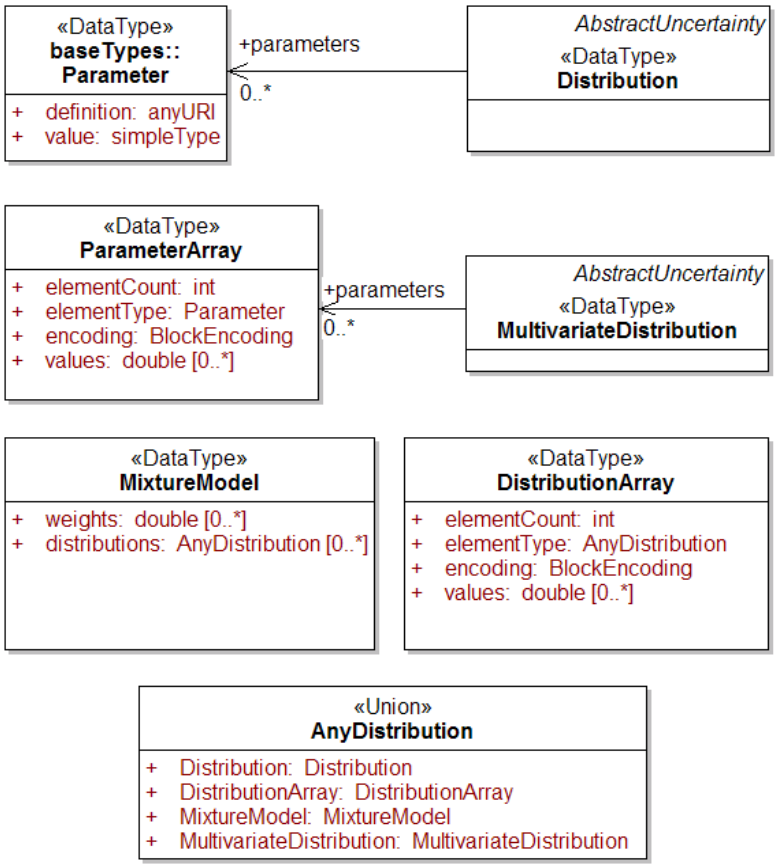

Figure 8: Parametric distributions in UncertML are encoded using one of the types above.

6.4.1 Distribution

For the simplest case (describing the probability distribution of a single variable), UncertML provides the `Distribution` type. Like all uncertainty types in UncertML, the `Distribution` type extends `AbstractUncertainty`, inheriting the `definition` property. In the case of distributions, this definition may contain both a textual description, and a complex mathematical description of the distribution (for example cumulative distribution function and

probability density function). It is important to note that the `Distribution` type is not a mechanism for completely describing a probability distribution in terms of its functions, parameters and how they relate to each other; it should be thought of as a mechanism for describing an *instance* of a distribution which is defined elsewhere. As previously stated, a dictionary containing definitions of common distributions is currently under development.

Complementing the `definition` property is a `parameters` property which contains a number of `Parameter` types. Each `Parameter` of a distribution is not considered to be an uncertainty type in itself, but it contains a `definition` property which can be used to specify this particular parameter. Each `Parameter` also has a `value` property holding the actual value of that parameter. We may extend this in the future to allow more complex conditional distributions to be encoded.

> The fact that complex mathematical functions are not embedded within the `Distribution` type has allowed a simple solution to encoding distributions. It can be argued that allowing functions to be described in-line, for example using content MathML, could act as an interoperable mechanism for automatically exchanging any distributions. However, we feel that in any context where users are working with previously unknown distributions, it will be necessary to spend time developing the required mathematics for processing these distributions. This obviates the need for a completely interoperable solution *in this instance*. Once an application is able to process a specific distribution all that is required to proceed with processing is a definition of that distribution. Examples of how various distributions may be encoded are detailed in Section8.
>
> The parsing and manipulation of distributions based on their definition can be a complex procedure, yet the majority of data users and analysts work with a small number of distributions (for example Gaussian, Poisson, exponential & log-normal).

6.4.2 DistributionArray

When describing the marginal distributions of several variables at a given domain point, or a univariate, marginal distribution over several domain points, the need for an array emerges. The `DistributionArray` type provided by UncertML is similar to the `StatisticsArray`. However, in this instance the `elementType` property is realised as any type from the `AnyDistribution` union. Typically, this will be a simple `Distribution` type; however, the mechanism for describing arrays of arrays, and arrays of `MultivariateDistributions`, is also available. The rest of the properties remain the same as in the `StatisticsArray`, with one subtle difference. Distributions often have numerous parameters that help describe them (e.g. a Gaussian distribution has both a location (centrality) and a scale (variance) parameter). In this instance the `Distribution` contained within the `elementType` property acts as a form of 'record'. Therefore, when interpreting distributions which have been encoded within the `values` property, care should be taken to clearly understand which set of values refers to

which parameter. The examples of `DistributionArrays` in Section 8 demonstrate how one may encode such a scenario, using a logical ordering of values to clarify their meaning.

6.4.3 MixtureModel

A `MixtureModel` is a specialised form of record. Typically, when describing a variable using a mixture of distributions, a specific weight is assigned to each distribution specifying the relative influence of that distribution in the mixture. This constraint meant that a simple 'DistributionRecord' type would not have been sufficient, so a dedicated `MixtureModel` was designed.

The `distributions` property is equivalent to the `fields` property of a standard record type which may contain a type from the `AnyDistribution` union. The addition of a `weights` property allows a weight (double) to be assigned to each distribution within the `distributions` property. It should be noted that when constructing `MixtureModels` of `DistributionArrays` or `MultivariateDistributions` care should be taken to ensure that there is a value for *each* individual distribution within the `distributions` property. In addition, though the sum of all values within the `weights` property should equal 1.0, this is not enforced within UncertML and it is the responsibility of the user to ensure such constraints are met.

Examples of `MixtureModels` may be found in Section 7.

6.4.4 MultivariateDistribution

The final, and perhaps most complex, type provided in UncertML is the `MultivariateDistribution` type. Typical use cases for a multivariate (joint) distribution are when two variables are correlated, or a single variable is spatially correlated. As each of these scenarios requires the inclusion of a covariance matrix, the `DistributionArray` is not sufficient to describe the structure of the uncertainty.

A `MultivariateDistribution` is similar to the `Distribution` type, containing both a `definition` and `parameters` property. However, a significant difference is that the `parameters` property of a `MultivariateDistribution` now contains a number of `ParameterArrays` rather than `Parameter` types, due to the fact that multivariate distributions, by definition, always deal with arrays of parameters.

The `ParameterArray` type is similar to all other array types within UncertML, consisting of an `elementType`, `elementCount`, `encoding` and `values` properties. The `elementType` property contains a `Parameter` type which provides a `definition` property. The `values` property then contains all values for that given parameter. A collection of such arrays allows the description of complex joint distributions in an efficient manner. Examples of how to use the `MultivariateDistribution` type are detailed in Section 8.

7 Relation to existing ISO standards

UncertML is not intended to be a solely geospatial standard, since the fundamental principle of UncertML (interoperable representation of probabilistic uncertainty) appeals to a wider set of application domains. A conscious effort has been made to separate concerns and to ensure that

UncertML is complete with respect to its remit: the probabilistic representation of uncertainty. Probabilistic representation is pertinent to many applications, and provides significant benefits in terms of a well defined theoretical basis for propagating and working with uncertainty. However, while we believe that UncertML has wide applicability, we feel that the largest potential application of, and our motivation for developing, UncertML lies within the Sensor Web; hence the submission of this document for discussion within the OGC.

This section briefly summarises our perceptions of the relationship of UncertML to existing standards, while later examples address the specific relevance of UncertML to these standards in practice. For example, section 8.4 shows the encoding of specific data quality measures from ISO 19138.

7.1 ISO 19115: Metadata

It appears that there are two ways of thinking when it comes to uncertainty and metadata - those who say that uncertainty is in fact data about data and those who say that the data itself is uncertain and thus the uncertainty is the actual data and not metadata. We would argue that both cases are applicable in different situations. For instance, when you have an observation, the result (or observed value), of this observation is considered the data. Any uncertainty in this observed value (e.g., measurement bias) can be considered metadata. However, when you consider a process, such as interpolation, the output of that process is itself uncertain - in this instance, the 'data' you are actually interested in is inherently uncertain.

With that in mind, the concepts that are discussed in ISO 19115 would suggest that UncertML is actually a realisation of the DQ_QuantitativeResult element, although in UncertML we have chosen not to include units of measure. Comparing UncertML to a DQ_QuantitativeResult in this manner may suggest that we consider uncertainty to be metadata, and this is certainly a domain where UncertML might be used. We have deliberately extended the definition in the DQ_QuantitativeResult section to allow very explicit but flexible representation of the uncertainty in the various types supported by UncertML. As noted in Section 3, and illustrated in Section 9, units of measure are not included in UncertML, but are intended to be represented by existing schemas which will fully describe the phenomena to which the uncertainty pertains, while using UncertML to efficiently describe that uncertainty.

A DQ_QuantitativeResult contains the following properties:

Property	**Nature**
`valueType[0..1]:`	`RecordType`
`valueUnit :`	`UnitOfMeasure`
`errorStatistic[0..1]:`	`CharacterString`
`Value [1..*]:`	`Record`

This information can be encoded using UncertML by delegating the ‘valueUnit’ to a ‘wrapper’ which could also encode measurement-specific information such as location and sensor characteristics. The uncertainty would be represented by an UncertML type in which the nature of the statistic is given by a URI, rather than the errorStatistic String used here. In Example 25, we show how a RandomVariable can be used in this way to wrap an UncertML type.

An alternative option would be to supply an UncertML type as the actual value of the DQ_QuantitativeResult, allowing the representation of qualified estimates for metrics which are

more usually supplied as a single value for a dataset. For example, traditionally a DQ_QuantitativeResult would be used to represent the fact that the value of 'TopologicalConsistency' (ISO 19138) for a polygon dataset is '75%'. UncertML allows us to record a more informative representation of this estimate, which is likely to be uncertain and to have been derived from a set of samples. This set of sample consistencies is also highly unlikely to have a symmetrical Gaussian distribution, given the 0-100 bounded nature of the measurement scale. UncertML can represent the variability of the samples without over-simplifying their distribution; for example, (a) by encoding a histogram/set of quantiles recording the full range of consistency values recorded; or (b) by recording an exceedance probability which tells us that, based on our evidence, there is a 99% chance that the 'TopologicalConsistency' exceeds 72%; or (c) by recording the raw sample values, representing the variation across the dataset. In those cases where the value of a data quality metric genuinely is certain, UncertML also offers benefits: encoding a value as a Dirac (delta) distribution conveys the fact that there is a strong and certain belief in the value.

7.2 ISO 19114: Quality evaluation procedures

UncertML fits with this standard in the same way as with ISO 19115 – via the DQ_QualityResult (realised as the subclass DQ_QuantitativeResult). ISO 19114 draws a strong distinction between the different types of data quality measures (e.g., thematic, positional or temporal). This is a topic that we do not address in UncertML; our thinking is that all these uncertainties can be described using the same basic UncertML types, within elements which deal with the context of the uncertainty in another schema. Thus the semantic description of, for example, how a 'vertical measurement error' differs from a 'percentage of incorrectly-classified pixels' would be the responsibility of elements specifically designed to describe these phenomena, while the statistical representation of uncertainty would be performed by UncertML. We feel that there is a fundamental difference between the conceptual approach of UncertML and this ISO standard, since while this standard looks at the different ways to describe the quality of a data set, UncertML looks at how the results of these measures can be encoded concisely.

7.3 ISO 19138: Data quality measures

ISO 19138 defines an extensive list of data quality measures that may be applied to datasets, but looks at describing these measures, rather than offering standard-specific means of actually representing those measures. It could be argued that UncertML should be capable of encoding all of the listed data quality measures, however, we are not convinced that this is within the scope of UncertML. Such a function might belong in a broader "DQML" - which would make extensive use of the types in UncertML.

There are a few terms that they use in this standard that we also use. 'Random variables' is one such concept which is shared, (though within this discussion document we refer to the values themselves as 'random quantities' while 'random variable' refers to a more detailed measurement record complete with units of measure). ISO 19138 relies on three assumptions when using random variables which seem rather restrictive in our view. These are:

1) Uncertainties are homogeneous for all observed values

2) The observed values are not correlated

3) The observed values are normally distributed

UncertML addresses these issues as follows:

1) UncertML provides the flexibility to encode individual and global measures of uncertainty for a dataset, rather than using a single statistic. This is quite essential for the applications we are considering; for example in a heterogeneous sensor network it is quite likely that different instruments, and deployments of those instruments, may generate very different observational errors / uncertainties. It is also true that the result of a processing method (for example an interpolation operation) would produce different uncertainties at each location within the interpolation domain.

2) This assumption seems contradictory, since the standard later discusses the possibility of describing 2 or 3 random variables using a covariance matrix. UncertML has been developed as part of the INTAMAP project which deals with interpolation. As a consequence, handling multiple correlated variables is a necessity for UncertML and we cannot make the same assumption as made in the document. Indeed the creation of the flexibility to model / represent correlated random variables was not taken lightly, and had the potential to greatly complicate the schema. It seems natural to us that one should be able to represent correlated random quantities.

3) Many of the uncertainties in this document's examples are assumed to be normally distributed - however, the flexibility of UncertML means that a user is not limited to using a normal distribution and may in fact describe a variable using any probability distribution. We anticipate that this will become increasingly important in the future as users better understand and model the uncertainty in their data. For example the results of many processing applications are likely to result in non-Gaussian distributions over their outputs, even when the inputs have an approximately Gaussian distribution. Also for many sensor types, for example most used in remote sensing, Gaussian errors are the exception, not the norm.

7.4 ISO 19118: Encoding

ISO 19118 specifies how encoding rules should be defined for interchange of geographical data within the ISO 19100 group of standards. UncertML has been designed with a careful eye to compliance with these recommendations; in particular, we have been careful to ensure that there is no obstacle to the use of UncertML within the ISO 19100 context; for example as a component in a GML application schema.

7.5 ISO 19119: Services

ISO 19119 defines and classifies architectures for geographic services, with particular focus on how platform-neutral services may be achieved. Since UncertML was originally developed within the context of Web Processing Services and service chaining (the INTAMAP project), a high priority in its design has been the provision of features to allow automated processing and 'understanding' of complex uncertainty data. The communication of uncertainty in the inputs and outputs of geographic services can only become more important as these services are used to process increasing quantities of online data and to inform decision-making. Our priority has been to ensure that UncertML would be usable within *any* geographic service which is correctly specified according to ISO 19119, and not just within the limited experimental context of our research. ISO 19119 states that 'a service specification shall include relevant information

models from the appropriate geographic information standards in the !SO 19100 series'. While UncertML is not an accepted standard in this group, we have ensured that its taxonomy is suitable for inclusion in such a service specification.

8 XML Encoding and Examples

The following section provides a series of XML instance examples to illustrate how UncertML may be utilised in practice.

All models discussed in the previous section are encoded as a set of XML schemata. When using UncertML, instance documents conforming to the rules set out in these schemata shall be created. All types within UncertML are concrete types that may be used without the development of an additional application schema.

The normative reference of UncertML is provided in the schemata in Section 10. All elements within the UncertML schemata belong to the *http://www.uncertml.org* namespace. However, the use of namespaces within this section has been omitted for brevity. Any element using the prefix *un* belongs to the UncertML namespace.

The UncertML schemata also utilise two external schemata:

- Geography Markup Language (GML), version 3.1.1, *http://www.opengis.net/gml*
- Sensor Web Enablement Common (SWE), version 1.0, *http://www.opengis.net/swe/1.0*

Elements prefixed with either *swe* or *gml* belong to the SWE Common or GML schemata respectively.

The rules used to encode the UncertML models into XML schemata are similar to those outlined in GML (2). UncertML uses the concept that there are 'Objects' that have 'properties' which may themselves be realised as 'Objects'. According to these rules 'Objects' are realised as XML elements and are named using UpperCamelCase and 'properties' are named using lowerCamelCase. XML types are named in UpperCamelCase and end with the word 'Type'.

Most properties and elements in the models above are encoded as elements in XML. However, several properties are realised as XML attributes where deemed necessary.

8.1 Realisations

Realisations exist for situations where no a priori knowledge of the distribution is available, or where complex processing of data, such as Monte Carlo analysis, is required. Typically, a data sample made up of realisations will contain hundreds or thousands of values and an efficient encoding is required.

```
<un:Realisations definition="http://dictionary.uncertml.org/realisation"
samplingMethod="http://dictionary.uncertml.org/realisations/sampling_met
hods/direct"
realisedFrom="http://dictionary.uncertml.org/distributions/gaussian">
   <un:realisationsCount>100</un:realisationsCount>
   <un:elementCount>100</un:elementCount>
   <swe:encoding>
      <swe:TextBlock decimalSeparator="." blockSeparator=" "
tokenSeparator=","/>
   </swe:encoding>
   <swe:values>
      <!-- [100 space separated values] -->
   </swe:values>
</un:Realisations>
```

Example 1: A set of realisations may be encoded using the Realisations type.

Example 1 shows how a set of 100 sample realisation values can be encoded. The `realisationsCount` property tells us that 100 realisations were generated, and in combination with the `elementCount` (also 100) it is possible to deduce that these realisations must refer to a single variable, at one domain point. However, information of this type is not intended to be extracted from the UncertML, as it would typically be encoded in a higher level wrapper representing the domain points and phenomena sampled (as shown in the later Example 22).

The three URIs present in this example provide information about this particular realisation. The `definition` property resolves to a description about the notion of a realisation, `realisedFrom` resolves to a definition of a Gaussian distribution (from which this sample is realised) and the `samplingMethod` resolves to a description of a direct sampling technique. Resolving these URIs gives the user a complete picture of the true meaning of these realisations when combined with the `values` property.

For this example, and all subsequent examples, we have chosen to use the `TextBlock` encoding as this best illustrates the examples. For more information about the other forms of encoding available via the SWE Common `EncodedValuesGroup` we refer to the SWE Common specification (1).

The `TextBlock` contains three properties. The `decimalSeparator` defines a single character that represents a decimal place, `blockSeparator` represents the character that separates each new element in the `values` block and `tokenSeparator` separates individual items within an element. In the above example there is only a single value within each element – the actual realisation itself – therefore the need for a token separator is removed. In later sections we demonstrate how the `tokenSeparator` may be used to encode complex elements, e.g. distributions.

8.2 Statistics

Statistics may be used to provide a concise summary of a particular variable. Section 8.2.1 gives several examples of simple statistics, quantiles and probabilities. Section 8.2.3 provides an example of how to group several statistics to form a customised summary of a variable, and goes on to demonstrate how one may encode complex structures such as histograms. Finally, Section

8.2.2 demonstrates how statistics and records can be encoded in an array to represent a set of statistics at several domain points (in this case, spatial locations).

8.2.1 Statistics, Quantiles, Probabilities & Moments

Due to the soft-typed approach of UncertML all simple statistics will appear identical in structure. What separates a 'mean' from a 'median' is the URI (and definition upon resolving) of the `definition` property. This is demonstrated in Example 2 and Example 3, where almost identical `Statistic` elements point to different definition URIs, maintaining a concise, yet flexible solution. Assuming the existence of a dictionary containing definitions of the most common statistics, only the URI is needed in order for an application to 'understand' how to process the data.

```
<un:Statistic
definition="http://dictionary.uncertml.org/statistics/mode">
   <un:value>34.67</un:value>
</un:Statistic>
```

Example 2: The Statistic type is used for encoding basic statistics such as mean and variance. A URI is used to provide the necessary semantics.

```
<un:Statistic
definition="http://dictionary.uncertml.org/statistics/standard_deviation
">
   <un:value>12.08</un:value>
</un:Statistic>
```

Example 3: A standard deviation encoded in UncertML.

A whole host of statistics may be encoded in a similar fashion to the examples above, including mean, variance & median. However, certain statistics require slightly more detail. One such example is a quantile (Example 4).

```
<un:Statistic
definition="http://dictionary.uncertml.org/statistics/quantile"
level="0.95">
   <un:value>34.34</un:value>
</un:Statistic>
```

Example 4: A Quantile extends the base Statistic type adding an additional level property which specifies the particular quantile of interest.

When dealing with quantiles, it is crucial to understand the particular quantile of interest, and this information is encoded in the `level` property. In the example above (Example 4) the 0.95 level quantile has a value of 34.34.

```
<un:Moment
definition="http://dictionary.uncertml.org/statistics/centred_moment">
   <un:value>34.5</un:value>
   <un:order>2</un:order>
</un:Moment>
```

Example 5: A Moment extends the base Statistic type but contains an additional order property.

Similar to the quantile example is the `Moment` (Example 5). When working with moments a user must know the moment *order* for meaningful processing. In Example 5 the 2nd order centred moment, or variance, is being described.

Another set of statistics requiring more information is probabilities, both discrete and continuous. A discrete probability requires an association to a particular class. An example of how one may encode a discrete probability in UncertML can be seen in Example 6.

```
<un:DiscreteProbability
definition="http://dictionary.uncertml.org/discrete_probability">
   <un:value>0.25</un:value>
   <un:category
definition="http://www.mydomain.com/trees/list">Oak</un:category>
</un:DiscreteProbability>
```

Example 6:The DiscreteProbability type should be used for quantifying variables which can be catagorised into discrete values..

The value of the `category` property may be of *any* type, but the typical use case would be to use a string (as demonstrated above). It should be noted that the `value` property of all probabilities represents a 'probability' in the range of 0.0 – 1.0, in contrast to other statistic types.

The final example of a statistic is a continuous probability (Example 7).

```
<un:Probability definition="http://dictionary.uncertml.org/probability">
   <un:value>0.25</un:value>
   <un:gt>35.6</un:gt>
</un:Probability>
```

Example 7: A continuous probability adds one or more properties to specify the thresholds that a variable may or may not exceed.

Continuous probabilities see the addition of five properties: `gt` (greater than), `lt` (less than), `eq` (equal to), `ge` (greater than or equal to) and `le` (less than or equal to). Using these properties individually, or in combination, allows a user to specify thresholds of importance for a particular variable. Examples include: "What is the probability that a variable exceeds a given value?" Or, "what is the probability that a variable falls between an upper and lower bound?" Utilising the upper and lower bounds of several `Probability` types in combination with the grouping of the `StatisticsRecord` type allows the user to construct a histogram (Example 11).

8.2.2 StatisticsArray

Large numbers of statistics can be efficiently handled using the `StatisticsArray` type. This aggregate type utilises the SWE Common `EncodedValuesGroup` to provide a suite of encoding options to suit a variety of needs.

In its simplest form, a `StatisticsArray` can provide a collection of instances of a single statistic, as in Example 8 below.

```
<un:StatisticsArray>
   <un:elementType>
      <un:Statistic
definition="http://dictionary.uncertml.org/statistics/mean"/>
   </un:elementType>
   <un:elementCount>5</un:elementCount>
   <swe:encoding>
      <swe:TextBlock decimalSeparator="." blockSeparator=" "
tokenSeparator=","/>
   </swe:encoding>
   <swe:values>46.76 25.75 57.432 12.42 53.64</swe:values>
</un:StatisticsArray>
```

Example 8: A StatisticsArray is used to group numerous identical statistics into an efficient structure.

When one wishes to encode a series of summary statistics (i.e. mean & standard deviation) a combination of the `StatisticsArray` and `StatisticsRecord` must be used. An example of such a scenario can be seen in Example 9 below.

```
<un:StatisticsArray>
   <un:elementType>
      <un:StatisticsRecord>
         <un:field>
            <un:Statistic
definition="http://dictionary.uncertml.org/statistics/mean" />
         </un:field>
         <un:field>
            <un:Statistic
definition="http://dictionary.uncertml.org/statistics/standard_deviation
" />
         </un:field>
      </un:StatisticsRecord>
   </un:elementType>
   <un:elementCount>5</un:elementCount>
   <swe:encoding>
      <swe:TextBlock decimalSeparator="." blockSeparator=" "
tokenSeparator=","/>
   </swe:encoding>
   <swe:values>
      34.64,67.86 45.65,78.41 56.7,45.75 29.86,56.74 45.65,76.43
   </swe:values>
</un:StatisticsArray>
```

Example 9: A StatisticsArray can be used in conjunction with the StatisticsRecord type to provide 'summary statistic' groups.

When using a record inside an array, the `tokenSeparator` separates each individual item within the record. Dissecting the `values` block in Example 9, a user can deduce that the values are in the following order: `Mean`, `Standard Deviation`.

8.2.3 StatisticsRecord

The previous section included an example of a `StatisticsRecord` inside a `StatisticsArray`, but sometimes only a single group of statistics may be required. An example of a single `StatisticsRecord` may be seen in Example 10, demonstrating how the mean, variance and the probability that a variable exceeds a certain threshold can be meaningfully grouped into a single structure.

```
<un:StatisticsRecord>
   <un:field>
      <un:Statistic
definition="http://dictionary.uncertml.org/statistics/mean">
         <un:value>34.5</un:value>
      </un:Statistic>
   </un:field>
   <un:field>
      <un:Statistic
definition="http://dictionary.uncertml.org/statistics/variance">
         <un:value>2.34</un:value>
      </un:Statistic>
   </un:field>
   <un:field>
      <un:Probability>
         <un:value>0.12</un:value>
         <un:gt>45.6</un:gt>
      </un:Probability>
   </un:field>
</un:StatisticsRecord>
```

Example 10: StatisticsRecords allow individual statistics to be grouped into a meaningful structure, summarising a variable at a single domain point.

Grouping statistics in this fashion allows complex structures to be formed. One such example is given in Example 11, which illustrates how a histogram can be constructed by grouping several `Probability` types into a `StatisticsRecord`.

```
<un:StatisticsRecord>
   <un:field>
      <un:Probability
definition="http://dictionary.uncertml.org/statistics/probability">
         <un:value>0.24</un:value>
         <un:lt>30</un:lt>
         <un:ge>10</un:ge>
      </un:Probability>
   </un:field>
   <un:field>
      <un:Probability
definition="http://dictionary.uncertml.org/statistics/probability">
         <un:value>0.57</un:value>
         <un:lt>50</un:lt>
         <un:ge>30</un:ge>
      </un:Probability>
   </un:field>
   <un:field>
      <un:Probability
definition="http://dictionary.uncertml.org/statistics/probability">
         <un:value>0.19</un:value>
         <un:lt>80</un:lt>
         <un:ge>50</un:ge>
      </un:Probability>
   </un:field>
</un:StatisticsRecord>
```

Example 11: StatisticsRecord can be used to form complex structures such as histograms. In this example, only a few bins are encoded, for brevity.

8.3 Distributions

The final subset of UncertML is concerned with the encoding of probability distributions. A range of options are available, including single distributions, arrays of distributions, joint distributions and mixture models. In Section 8.3.1 we demonstrate how some of the more common distributions may be encoded. Section 8.3.2 examines how to encode an array of distributions in the `DistributionArray` type. Sections 8.3.3 – 8.3.4 look at the more complex encodings of mixture models and multivariate distributions respectively.

8.3.1 Distribution

A `Distribution` type in UncertML may be thought of as a 'record' type. However, rather than having an unbounded number of 'fields' it has 'parameters'. The decision to extract all mathematical functions from the encoding of a distribution has enabled complex notions, such as 'Gaussian distribution' to be easily encoded in a simple structure, as shown in Example 12.

```
<un:Distribution
definition="http://dictionary.uncertml.org/distributions/gaussian">
   <un:parameters>
      <un:Parameter
definition="http://dictionary.uncertml.org/distributions/gaussian/mean">
         <un:value>34.564</un:value>
      </un:Parameter>
      <un:Parameter
definition="http://dictionary.uncertml.org/distributions/gaussian/varian
ce">
         <un:value>7.45</un:value>
      </un:Parameter>
   </un:parameters>
</un:Distribution>
```

Example 12: A Gaussian distribution is simple to encode in UncertML.

Generating a weak-typed framework such as this allows *any* distribution to be encoded in one generic 'distribution' type. Provided that any processing applications understand which distribution is being described (by resolving the URIs) then there is no need to include functions inline.

This generic feature is demonstrated in the example below (Example 13), which encodes an exponential distribution.

```
<un:Distribution
definition="http://dictionary.uncertml.org/distributions/exponential">
   <un:parameters>
      <un:Parameter
definition="http://dictionary.uncertml.org/distributions/exponential/rat
e">
         <un:value>34.564</un:value>
      </un:Parameter>
   </un:parameters>
</un:Distribution>
```

Example 13: An Exponential distribution encoded in UncertML

8.3.2 DistributionArray

As with statistics, a situation may arise where a user wishes to encode multiple marginal distributions in an efficient structure. The `DistributionArray` type included in UncertML is similar in structure to a `StatisticsArray` that contains a `StatisticsRecord`.

```
<un:DistributionArray>
   <un:elementType>
      <un:Distribution
definition="http://dictionary.uncertml.org/distributions/gaussian">
         <un:parameters>
            <un:Parameter
definition="http://dictionary.uncertml.org/distributions/gaussian/mean"/
>
            <un:Parameter
definition="http://dictionary.uncertml.org/distributions/gaussian/varian
ce"/>
         </un:parameters>
      </un:Distribution>
   </un:elementType>
   <un:elementCount>5</un:elementCount>
   <swe:encoding>
      <swe:TextBlock decimalSeparator="." blockSeparator=" "
tokenSeparator=","/>
   </swe:encoding>
   <swe:values>
      35.2,56.75
      31.2,65.31
      28.2,54.23
      35.6,45.21
      41.5,85.24
   </swe:values>
</un:DistributionArray>
```

Example 14: A DistributionArray containing several Gaussian distributions. The values block contains 5 'mean,variance ' tuples, one for each Gaussian distribution.

In the example above, you can see that each parameter of the distribution (mean & variance) is separated by the `tokenSeparator`. Each individual distribution is then separated by the `blockSeparator` giving the following pattern: `Mean`,`Variance` `Mean`,`Variance` etc…

8.3.3 MixtureModel

Mixture models are distributions that are convex combinations of other, weighted, distributions. Semantically, a `MixtureModel` is not very dissimilar to any other record type in UncertML. There are a number of constituent fields (distributions) that make up a larger structure. However, the addition of a `weights` property allows you to assign a relative weight, or influence, to each constituent part. Certain guidelines should be followed when using `MixtureModel`s to ensure interoperability. These are as follows:

- There should be a corresponding weight for each constituent distribution within the `distributions` property
- Each weight should be a value between 0.0 – 1.0 inclusive (but this is not enforced).
- The sum of all weights should equal 1.0 (though again, this is not enforced).

Below is an example of a `MixtureModel` (Example 15) consisting of multiple Gaussian distributions and adhering to the above rules.

```
<un:MixtureModel>
   <un:weights>0.24 0.76</un:weights>
   <un:distributions>
      <un:Distribution
definition="http://dictionary.uncertml.org/distributions/gaussian">
         <un:parameters>
            <un:Parameter
definition="http://dictionary.uncertml.org/distributions/gaussian/mean">
               <un:value>34.564</un:value>
            </un:Parameter>
            <un:Parameter
definition="http://dictionary.uncertml.org/distributions/gaussian/varian
ce">
               <un:value>67.45</un:value>
            </un:Parameter>
         </un:parameters>
      </un:Distribution>
      <un:Distribution
definition="http://dictionary.uncertml.org/distributions/gaussian">
         <un:parameters>
            <un:Parameter
definition="http://dictionary.uncertml.org/distributions/gaussian/mean">
               <un:value>21.564</un:value>
            </un:Parameter>
            <un:Parameter
definition="http://dictionary.uncertml.org/distributions/gaussian/varian
ce">
               <un:value>34.45</un:value>
            </un:Parameter>
         </un:parameters>
      </un:Distribution>
   </un:distributions>
</un:MixtureModel>
```

Example 15: A MixtureModel includes a number of 'weights' that corresponds to the number of constituent distributions.

Combining distributions and weights in this fashion allows the construction of complex mixture models. The ability to include `DistributionArrays` into the `distributions` property allows a user to create large mixture models efficiently. Care should be taken when working with large mixture models to ensure adherence to the above rules.

8.3.4 MultivariateDistribution

The final type in the distributions subset of UncertML is the `MultivariateDistribution`. This type is required when there is correlation between variables, or a variable varies over a spatial-temporal domain. Much like a `Distribution` type, a `MultivariateDistribution` consists of a `definition` and `parameters` properties. However, instead of each parameter having a single value, it now consists of multiple values, depending upon either the number of variables or the extent of the domain. The notion of a multivariate distribution is complex, but the example in Example 16 demonstrates the relative ease with which a multivariate Gaussian distribution may be encoded.

```
<un:MultivariateDistribution
definition="http://dictionary.uncertml.org/distributions/multivariate_ga
ussian">
   <un:parameters>
      <un:ParameterArray>
         <un:elementType>
            <un:Parameter
definition="http://dictionary.uncertml.org/distributions/multivariate_ga
ussian/mean"/>
         </un:elementType>
         <un:elementCount>5</un:elementCount>
         <swe:encoding>
            <swe:TextBlock decimalSeparator="." blockSeparator=" "
tokenSeparator=","/>
         </swe:encoding>
         <swe:values>
            45.42 53.12 12.53 64.21 55.22
         </swe:values>
      </un:ParameterArray>
      <un:ParameterArray>
         <un:elementType>
            <un:Parameter
definition="http://dictionary.uncertml.org/distributions/multivariate_ga
ussian/covariance"/>
         </un:elementType>
         <un:elementCount>25</un:elementCount>
         <swe:encoding>
            <swe:TextBlock decimalSeparator="." blockSeparator=" "
tokenSeparator=","/>
         </swe:encoding>
         <swe:values>
            2.71828 0 0 0 0
            0 2.71828 0 0 0
            0 0 2.71828 0 0
            0 0 0 2.71828 0
            0 0 0 0 2.71828
         </swe:values>
      </un:ParameterArray>
   </un:parameters>
</un:MultivariateDistribution>
```

Example 16: MultivariateDistributions may be used to describe a variable that varies over space or time as well as the correlation between multiple variables.

The first `ParameterArray` contains a vector of mean values whose dimensions depend upon the number of variables, and the size of the spatial or temporal domain. The second `ParameterArray` contains the covariance matrix, which will contain the appropriate number of covariance values. Utilising the SWE Common `EncodedValuesGroup` will ensure that the most efficient encoding of covariances is achieved in applications with large numbers of domain points, since these matrices can become exceedingly large under normal use.

8.4 ISO 19138 data quality measures

The ISO 19138 specification outlines a list of commonly used data quality reporting measures for the data quality subelements identified in ISO 19113. This section provides several examples of how UncertML can be used to encode these data quality measures in XML.

```
<un:Statistic
definition="http://dictionary.uncertml.org/statistics/Mean_value_of_posi
tional_uncertainties" degreesOfFreedom="228">
   <un:value xsi:type="xs:double">24.21</un:value>
</un:Statistic>
```

Example 17: Mean value of positional uncertainties (ISO 19138 Table D.29)

Example 17 demonstrates how the statistic 'mean value of positional uncertainties' may be encoded using UncertML. The example is identical to previous statistic examples with only the `definition` URL differentiating it. With the addition of a Parameter, Example 18 demonstrates how the 'mean value of positional uncertainties' can be extended to exclude outliers. In the example, the e_{max} parameter is used to specify the definition of what an outlier is in this particular context, and its threshold nature and value type is therefore also specified within the dictionary.

```
<un:Statistic
definition="http://dictionary.uncertml.org/statistics/Mean_value_of_posi
tional_uncertainties_excluding_outliers">
   <un:parameters>
      <un:Parameter
definition="http://dictionary.uncertml.org/statistics/Mean_value_of_posi
tional_uncertainties_excluding_outliers/parameters/e_max">
         <un:value xsi:type="xs:double">25.12</un:value>
      </un:Parameter>
   </un:parameters>
   <un:value xsi:type="xs:double">23.45</un:value>
</un:Statistic>
```

Example 18: Mean value of positional uncertainties excluding outliers (ISO 19138 Table D.30)

The inclusion of data quality elements from the ISO 19138 specification, such as 'data quality value type' and 'data quality value structure', within the dictionary allows more complex statistics, such as a covariance matrix (Example 19), to be encoded.

```
<un:Statistic
definition="http://dictionary.uncertml.org/statistics/Covariance_matrix"
>
   <un:value xsi:type="xs:base64Binary">
      Mi43MTgyOCAwIDAgMCAwIDAgMi43MTgyOCAwIDAgMCAwIDAgMi43MTgyOCAwIDAgMC
      AwIDAgMi43MTgyOCAwIDAgMCAwIDAgMi43MTgyOA==
   </un:value>
</un:Statistic>
```

Example 19: Covariance matrix (ISO 19138 Table D.33)

Example 19 also demonstrates the ability of a statistic `value` to contain any simple XML type; in this instance the covariance matrix is encoded in base 64.

Example 20 is an extract from the statistics dictionary that describes the concept and structure of a covariance matrix. The `dataQualityValueType` property states that it is of type 'Measures', and the `dataQualityValueStructure` defines it as 'matrix'. With this knowledge it is possible to deduce what the statistic `value` property contains.

```
<un:StatisticDefinition gml:id="Covariance_matrix">
   <gml:description>
   The covariance matrix generalises the concept of variance from one to
n dimensions, i.e. from scalar-valued random quantities to vector-valued
random quantities (tuples or scalar random quantities).
   </gml:description>
   <gml:name>Covariance matrix</gml:name>
   <gml:name>variance-covariance matrix</gml:name>
   <un:definition>
   A symmetrical square matrix with variances or point coordinates on
the main diagonal and covariances between these coordinates as off-
diagonal elements
   </un:definition>
   <un:dataQualityValueType>Measures</un:dataQualityValueType>
   <un:dataQualityValueStructure>matrix</un:dataQualityValueStructure>
</un:StatisticDefinition>
```

Example 20: Dictionary extract for the description of a covariance matrix

Another example of how the `dataQualityValueType` and `dataQualityValueStructure` properties within the definitions are used can be seen in Example 21. The uncertainty ellipse, as defined in ISO 19138, does not contain a single value, rather a list (a, b, φ) of values. Including this information within the dictionary definition allows the base `Statistic` type to encode a multitude of different statistics, including the uncertainty ellipse, within the same structure.

```
<un:Statistic
definition="http://dictionary.uncertml.org/statistics/Uncertainty_ellips
e">
   <un:value xsi:type="xs:string">21.14, 12.42, 125.12</un:value>
</un:Statistic>
```

Example 21: Uncertainty ellipse (ISO 19138 Table D.52

9 UncertML Best Practice

In this section we present a series of use cases that demonstrate how we feel UncertML should be used within a variety of domains. In particular we propose a `RandomVariable` type to encode information on units of measure and other properties, as a helper to UncertML. This could be included in the SWE standard in the future to augment the `swe:Quantity` type. The examples are intended to suggest how we think UncertML should be used within a range of scenarios. We hope that the examples show that we have achieved our primary goal of making UncertML usable within a very wide range of situations, with minimal dependencies.

9.1 3 tier architecture

This document concentrates on how to encode and implement uncertain information into existing standards. It does *not* instruct on how to encode supporting information such as units of measure and spatial domains. In geospatial contexts, it is anticipated that UncertML may be used in a 3-tier architecture as demonstrated in Figure 9, with each supporting layer in this architecture adding an extra level of detail. Delegating appropriate responsibilities to supporting schemata decouples UncertML from any one existing standard and ensures that it may be integrated into a wide range of domains.

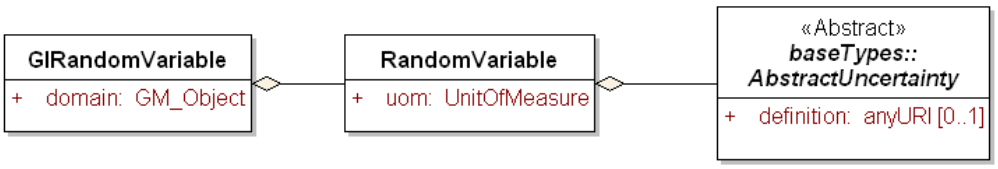

Figure 9: UncertML does not provide a mechanism for describing units of measure, geospatial domains or any other such properties. The removal of such constraints allows UncertML to be utilised in a variety of different domains.

Given the wide applicability of UncertML, this tiered arrangement may consist of as many levels as are necessary to clearly characterise a measurement, using the most appropriate schemata for that problem domain.

9.2 Sensor Noise Model

Example 22 demonstrates how to encode a sensor noise model in an Observations & Measurements (O&M) document. The 'sa' and 'gml' namespaces are used, in keeping with common practice, to encode the results of a measurement sampled by the sensor in question, and the point location of the sensor.

The `result` property contains a temperature measurement of 19.4° Celsius. However, the `resultQuality` property indicates (through the use of an encapsulated UncertML `Distribution`) that this temperature has an associated variance of 3.6 around the measured value with no bias, as indicated by the mean value of 0.0. An assumption is made that the units of measure within UncertML are the same as those specified within the `result` property (degrees Celsius). Units of measure are discussed later when we propose a random variable type.

```
<om:Observation>
   <om:samplingTime>
      <gml:TimeInstant>
         <gml:timePosition>2008-07-07T13:59</gml:timePosition>
      </gml:TimeInstant>
   </om:samplingTime>
   <om:procedure
xlink:href="http://www.mydomain.com/sensor_models/temperature"/>
   <om:resultQuality>
      <un:Distribution
definition="http://dictionary.uncertml.org/distributions/gaussian">
         <un:parameters>
            <un:Parameter
definition="http://dictionary.uncertml.org/distributions/gaussian/parame
ters/mean">
               <un:value>0.0</un:value>
            </un:Parameter>
            <un:Parameter
definition="http://dictionary.uncertml.org/distributions/gaussian/parame
ters/variance">
               <un:value>3.6</un:value>
            </un:Parameter>
         </un:parameters>
      </un:Distribution>
   </om:resultQuality>
   <om:observedProperty xlink:href="urn:x-
ogc:def:phenomenon:OGC:AirTemperature"/>
   <om:featureOfInterest>
      <sa:SamplingPoint>
         <sa:sampledFeature
xlink:href="http://www.mydomain.com/sampling_stations/ws-04231"/>
         <sa:position>
            <gml:Point>
               <gml:pos srsName="urn:x-ogc:def:crs:EPSG:4326">
                  52.4773635864 -1.89538836479
               </gml:pos>
            </gml:Point>
         </sa:position>
      </sa:SamplingPoint>
   </om:featureOfInterest>
   <om:result xsi:type="gml:MeasureType"
uom="urn:ogc:def:uom:OGC:degC">19.4</om:result>
</om:Observation>
```

Example 22: UncertML can be used to encode a noise model for a particular sensor in conjunction with the O&M standard.

9.3 Interpolation Results

The following example (Example 23) illustrates the encoding of results from a commonly used interpolation process. Since interpolation is essentially a method of prediction, there are uncertainties inherent within the results generated. Example 23 displays the encoding of typical output from a spatial interpolation process such as kriging where values have been predicted, along with associated uncertainties, to a regular grid of points.

This example uses the GML `RectifiedGrid` type to encode a 2x3 grid – the spatial domain of the interpolation. Within the `range` is the result of the interpolation; a `StatisticsArray`. This example demonstrates a series of six marginal (uncorrelated) mean and variance values.

The `RandomVariable` type is an example of one of the many supporting schemata which may be used to wrap and present UncertML, ensuring that numerical uncertainty information is cleanly separated from metadata such as units of measure and spatio-temporal domain descriptions. This clear division of scope allows UncertML to be used in a wide variety of data representation contexts, and embedded within many existing application-specific XML schemata. The careful reader will spot that there is considerable duplication in the example: the `rv:domainPointCount` is designed to identify the number of points at which the random variable is represented; clearly this can be derived from the domain set, but the `RandomVariable` is designed such that it could be used independently of a spatial context, where the number of points might be an important item to provide. This information is again duplicated in the `un:elementCount` - again, with the rationale that UncertML might be used outside the `RandomVariable` type. This illustrative example includes all three values, however, since they are envisaged to be optional outside UncertML this sort of duplication would be unlikely in practice.

9.4 Discrete Probabilities

The use case in Example 24 demonstrates the results of a Monte Carlo simulation of rabies in Finland, whose results have been categorised into two classes. Such an encoding could easily be wrapped in an observation supplying other information, e.g. the feature of interest & observation time.

The results of the simulation show that there is an 87% chance that the disease dies out within 5 years and only a 13% chance it will survive for longer. Using discrete probabilities in this way provides a quick summary overview of a situation, rather than requiring a user to process the individual realisations.

9.5 Probabilistic Weather Forecast

This use case consists of a histogram of predicted temperature values generated from an ensemble weather forecasting system. The example is encoded as an observation in the O&M schema. However, in contrast to the first use case, this example uses UncertML as the `result` property and the `resultQuality` property is left empty. This is because in this case the result, being predicted from a model, is inherently uncertain, whereas in the observation case, an actual measurement was being represented, in combination with associated quality information about the likely sampling errors.

From the example in Example 25 you can see that there are 5 bins: 18.0-20.0, 20.0-22.0, 22.0-24.0 & 24.0-26.0. Each bin has a different associated probability: 0.07, 0.22, 0.41, 0.21 & 0.09 respectively. An application could use this data in several ways; a typical use would be to produce a graphical display of the histogram. The `result` of this observation is wrapped in a `RandomQuantity` type, demonstrating how one may use UncertML with a supporting schema to encode a random variable. The structure of the `RandomVariable` type mimics the SWE `Quantity` type, adding a definition of the phenomenon (variable), its units of measure and the actual value. The difference between a `RandomVariable` and a normal `Quantity` is that the `RandomVariable` may have any UncertML type as its value – providing the necessary quantification of uncertainty.

9.6 Multivariate Statistics

This section gives three examples of how UncertML may be used to describe multivariate statistics either through realisations, statistics or distributions.

Consider the scenario of a complex micro-climate simulation model (for example within a building) which simulates temperature and pressure on a staggered grid. Stochastic forcing from the simulated external weather means the results of the model are also stochastic. In Example 26, 100 realisations have been generated from this simulation and encoded using UncertML.

As already noted, UncertML is an XML language designed for describing uncertainty, and certain attributes commonly associated with uncertainty (e.g. units of measure) are *not* encoded within the UncertML specification. Such attributes are expected to be encoded by other supporting schemata. In this example we have included a `JointRandomVariable` type to provide information such as the phenomena being described, their units of measure and the number of domain points at which they are measured, which in this case are required for decoding. Providing this information allows multivariate realisations to be encoded in a single `Realisations` type. The values within the array are ordered according to the following rules.

- Individual realisations are encoded for every domain point
- Individual realisations are grouped by phenomenon
- The order of domain points refer to some 'domain' that is described in another schema

We note that this goes a little against the desire to explicitly define everything inside XML tags, however for the usage scenarios we have in mind, which might involve very large data sets, we believe that the use of encoding is well justified. We also endeavoured to separate responsibility in UncertML to keep it as clean and clear as possible.

Example 27 uses various types within the `Statistics` schema to encode the mean values of, and covariance within and between, annual mean temperature and rainfall, observed over 50 years instances. Note the covariance encodes both the internal relationship between the temperature and rainfall, and their joint structure.

The `JointRandomVariable` type is syntactically identical to that used in the previous example. However, the random variables are now described by their mean values and a corresponding covariance matrix, describing the correlation between them.

The use of a covariance matrix is often seen when two (or more) variables are jointly distributed. Decoding the covariance matrix is done in row major format at present, although a dedicated type might be employed to encode the covariance more compactly in the future. Example 28 demonstrates an interpolation where the results are correlated spatially, and where users can provide a joint distribution to fully represent the characteristics of the interpolated result and its uncertainty.

The multivariate example uses a smaller domain (2x2 grid) which provides a set of four mean values and a covariance matrix of size sixteen (four by four).

It should be noted that the supporting elements in these examples are for illustration only and are not part of the UncertML specification.

```
<InterpolationResult>
   <domain>
      <gml:RectifiedGrid dimension="2">
         <gml:limits>
            <gml:GridEnvelope>
               <gml:low>0 0</gml:low>
               <gml:high>2 3</gml:high>
            </gml:GridEnvelope>
         </gml:limits>
         <gml:axisName>x</gml:axisName>
         <gml:axisName>y</gml:axisName>
         <gml:origin>
            <gml:Point>
               <gml:pos srsName="urn:x-
ogc:def:crs:EPSG:4326">52.4773635864 -1.89538836479</gml:pos>
            </gml:Point>
         </gml:origin>
         <gml:offsetVector srsName="urn:x-
ogc:def:crs:EPSG:4326">0.1453232 0</gml:offsetVector>
         <gml:offsetVector srsName="urn:x-ogc:def:crs:EPSG:4326">0
0.1453232</gml:offsetVector>
      </gml:RectifiedGrid>
   </domain>
   <range>
      <rv:RandomVariable>
         <rv:phenomena>
            <rv:Phenomenon definition="urn:x-
ogc:def:phenomenon:OGC:AirTemperature">
               <rv:uom href="urn:ogc:def:uom:OGC:degC"/>
               <rv:domainPointCount>6</rv:domainPointCount>
            </rv:Phenomenon>
         </rv:phenomena>
         <rv:representation>
            <un:StatisticsArray>
               <un:elementType>
                  <un:StatisticsRecord>
                     <un:field>
                        <un:Statistic
definition="http://dictionary.uncertml.org/statistics/mean"/>
                     </un:field>
                     <un:field>
                        <un:Statistic
definition="http://dictionary.uncertml.org/statistics/variance"/>
                     </un:field>
                  </un:StatisticsRecord>
               </un:elementType>
               <un:elementCount>6</un:elementCount>
               <swe:encoding>
                  <swe:TextBlock decimalSeparator="." blockSeparator=" "
tokenSeparator=","/>
               </swe:encoding>
               <swe:values>
                  21.3,4.4
                  19.6,4.8
                  18.3,5.9
                  22.1,4.9
                  20.5,6.1
                  19.8,8.3
               </swe:values>
            </un:StatisticsArray>
         </rv:representation>
      </rv:RandomVariable>
   </range>
</InterpolationResult>
```

Example 23: UncertML is used to describe the result of an interpolation.

```
<un:StatisticsRecord>
   <un:field>
      <un:DiscreteProbability
definition="http://dictionary.uncertml.org/discrete_probability">
         <un:value>0.87</un:value>
         <un:category
definition="http://www.mydomain.com/finland/rabies_classification">Disea
se dies out within 5 years</un:category>
      </un:DiscreteProbability>
   </un:field>
   <un:field>
      <un:DiscreteProbability
definition="http://dictionary.uncertml.org/discrete_probability">
         <un:value>0.13</un:value>
         <un:category
definition="http://www.mydomain.com/finland/rabies_classification">Disea
se survives more than 5 years</un:category>
      </un:DiscreteProbability>
   </un:field>
</un:StatisticsRecord>
```

Example 24: `DiscreteProbabilitys` can be used to concisely describe the outcome of a simulation.

```
<om:Observation>
   <om:samplingTime>
      <gml:TimeInstant>
         <gml:timePosition>2008-07-29T12:00</gml:timePosition>
      </gml:TimeInstant>
   </om:samplingTime>
   <om:procedure
xlink:href="http://www.mydomain.com/model/myEnsembleModel.xml"/>
   <om:observedProperty xlink:href="urn:x-
ogc:def:phenomenon:OGC:AirTemperature"/>
   <om:featureOfInterest>
      <sa:SamplingPoint>
         <sa:sampledFeature
xlink:href="http://www.mydomain.com/sampling_stations/ws-04236"/>
         <sa:position>
            <gml:Point>
               <gml:pos srsName="urn:x-ogc:def:crs:EPSG:4326">
                  52.1 2.5
               </gml:pos>
            </gml:Point>
         </sa:position>
      </sa:SamplingPoint>
   </om:featureOfInterest>
   <om:result>
      <rv:RandomVariable>
         <rv:phenomena>
            <rv:Phenomenon
definition="ogc:def:phenomenon:OGC:AirTemperature">
               <rv:uom xlink:href="urn:ogc:def:uom:OGC:degC"/>
               <rv:domainPointCount>1</rv:domainPointCount>
               <rv:representation>
                  <un:StatisticsRecord>
                     <un:field>
                        <un:Probability
definition="http://dictionary.uncertml.org/statistics/probability">
                           <un:value>0.07</un:value>
                           <un:lt>18.0</un:lt>
```

```
                              <un:ge>16.0</un:ge>
                           </un:Probability>
                        </un:field>
                        <un:field>
                           <un:Probability
definition="http://dictionary.uncertml.org/statistics/probability">
                              <un:value>0.22</un:value>
                              <un:lt>20.0</un:lt>
                              <un:ge>18.0</un:ge>
                           </un:Probability>
                        </un:field>
                        <un:field>
                           <un:Probability
definition="http://dictionary.uncertml.org/statistics/probability">
                              <un:value>0.41</un:value>
                              <un:lt>22.0</un:lt>
                              <un:ge>20.0</un:ge>
                           </un:Probability>
                        </un:field>
                        <un:field>
                           <un:Probability
definition="http://dictionary.uncertml.org/statistics/probability">
                              <un:value>0.21</un:value>
                              <un:lt>24.0</un:lt>
                              <un:ge>22.0</un:ge>
                           </un:Probability>
                        </un:field>
                        <un:field>
                           <un:Probability
definition="http://dictionary.uncertml.org/statistics/probability">
                              <un:value>0.09</un:value>
                              <un:lt>26.0</un:lt>
                              <un:ge>24.0</un:ge>
                           </un:Probability>
                        </un:field>
                     </un:StatisticsRecord>
                  </rv:representation>
               </rv:Phenomenon>
            </rv:phenomena>
         </rv:RandomVariable>
      </om:result>
</om:Observation>
```

Example 25: UncertML can be used within the result property of an Observation as well as the resultQuality property.

```
<rv:JointRandomVariable>
   <rv:phenomena>
      <rv:Phenomenon definition="urn:x-
ogc:def:phenomenon:OGC:AirTemperature">
         <rv:uom xlink:href="urn:ogc:def:uom:OGC:degC"/>
         <rv:domainPointCount>4</rv:domainPointCount>
      </rv:Phenomenon>
      <rv:Phenomenon definition="urn:x-ogc:def:phenomenon:OGC:Pressure">
         <rv:uom xlink:href="urn:ogc:def:uom:OGC:kPa"/>
         <rv:domainPointCount>7</rv:domainPointCount>
      </rv:Phenomenon>
   </rv:phenomena>
   <rv:representation>
      <un:Realisations
definition="http://dictionary.uncertml.org/realisation"
samplingMethod="http://dictionary.uncertml.org/sampling_methods/MCMC"
realisedFrom="http://dictionary.uncertml.org/distribtuions/gaussian">
         <un:numberRealisations>100</un:numberRealisations>
         <un:elementCount>1100</un:elementCount>
         <swe:encoding>
            <swe:TextBlock decimalSeparator="." blockSeparator=" "
tokenSeparator=","/>
         </swe:encoding>
         <swe:values>
            <!-- Realisation 1 -->
            [4 Temperature Realisations + 7 Pressure Realisations]
            <!-- Realisation 2 -->
            [4 Temperature Realisations + 7 Pressure Realisations]
            <!-- Realisation 3 -->
            [4 Temperature Realisations + 7 Pressure Realisations]

            <!-- ................... -->

            <!-- Realisation 98 -->
            [4 Temperature Realisations + 7 Pressure Realisations]
            <!-- Realisation 99 -->
            [4 Temperature Realisations + 7 Pressure Realisations]
            <!-- Realisation 100 -->
            [4 Temperature Realisations + 7 Pressure Realisations]
         </swe:values>
      </un:Realisations>
   </rv:representation>
</rv:JointRandomVariable>
```

Example 26: Realisations used to describe the correlation between two phenomena.

```
<rv:JointRandomVariable>
   <rv:phenomena>
      <rv:Phenomenon definition="urn:x-ogc:def:phenomenon:OGC:Rainfall">
         <rv:uom xlink:href="urn:ogc:def:uom:OGC:mm"/>
         <rv:domainPointCount>50</rv:domainPointCount>
      </rv:Phenomenon>
      <rv:Phenomenon definition="urn:x-
ogc:def:phenomenon:OGC:AirTemperature">
         <rv:uom xlink:href="urn:ogc:def:uom:OGC:degF"/>
         <rv:domainPointCount>50</rv:domainPointCount>
      </rv:Phenomenon>
   </rv:phenomena>
   <rv:representation>
      <un:StatisticsArray>
         <un:elementCount>100</un:elementCount>
         <un:elementType>
            <Statistic
definition="http://dictionary.uncertml.org/statistics/mean"/>
         </un:elementType>
         <swe:encoding decimalSeperator="." tupleSeperator=" "
blockSeperator=","/>
         <swe:values>
            <!-- [50 Mean Rainfall] -->
            <!-- [50 Mean Temperature] -->
         </swe:values>
      </un:StatisticsArray>
      <un:StatisticsArray>
         <un:elementCount>10000</un:elementCount>
         <un:elementType>
            <un:Statistic
definition="http://dictionary.uncertml.org/statistics/covariance"/>
         </un:elementType>
         <swe:encoding decimalSeperator="." tupleSeperator=" "
blockSeperator=","/>
         <swe:values>
            <!-- [10000 'Covariance' Values] -->
         </swe:values>
      </un:StatisticsArray>
   </rv:representation>
</rv:JointRandomVariable>
```

Example 27: A StatisticsArray is used to describe the mean values of two phenomena and a covariance matrix summarising the relationships between them.

```
<InterpolationResult>
   <domain>
      <gml:RectifiedGrid dimension="2">
         <gml:limits>
            <gml:GridEnvelope>
               <gml:low>0 0</gml:low>
               <gml:high>2 2</gml:high>
            </gml:GridEnvelope>
         </gml:limits>
         <gml:axisName>x</gml:axisName>
         <gml:axisName>y</gml:axisName>
         <gml:origin>
            <gml:Point>
               <gml:pos srsName="urn:x-ogc:def:crs:EPSG:4326">52.4773635864
-1.89538836479</gml:pos>
```

```
            </gml:Point>
         </gml:origin>
         <gml:offsetVector srsName="urn:x-ogc:def:crs:EPSG:4326">0.1453232
0</gml:offsetVector>
         <gml:offsetVector srsName="urn:x-ogc:def:crs:EPSG:4326">0
0.1453232</gml:offsetVector>
      </gml:RectifiedGrid>
   </domain>
   <range>
      <rv:JointRandomVariable>
         <rv:phenomena>
            <rv:Phenomenon definition="urn:x-
ogc:def:phenomenon:OGC:AirTemperature">
               <rv:uom href="urn:ogc:def:uom:OGC:degC"/>
               <rv:domainPointCount>4</rv:domainPointCount>
            </rv:Phenomenon>
         </rv:phenomena>
         <rv:representation>
            <un:MultivariateDistribution
definition="http://dictionary.uncertml.org/distributions/multivariate_gauss
ian">
               <un:parameters>
                  <un:ParameterArray>
                     <un:elementType>
                        <un:Parameter
definition="http://dictionary.uncertml.org/distributions/multivariate_gauss
ian/parameters/mean"/>
                     </un:elementType>
                     <un:elementCount>4</un:elementCount>
                     <swe:encoding>
                        <swe:TextBlock decimalSeparator="."
blockSeparator=" " tokenSeparator=","/>
                     </swe:encoding>
                     <swe:values>
                        21.3 19.6 22.1 20.5
                     </swe:values>
                  </un:ParameterArray>
                  <un:ParameterArray>
                     <un:elementType>
                        <un:Parameter
definition="http://dictionary.uncertml.org/distributions/multivariate_gauss
ian/parameters/covariance"/>
                     </un:elementType>
                     <un:elementCount>16</un:elementCount>
                     <swe:encoding>
                        <swe:TextBlock decimalSeparator="."
blockSeparator=" " tokenSeparator=","/>
                     </swe:encoding>
                     <swe:values>
                        4.4 3.3 3.1 1.7 3.3
                        4.8 3.7 2.9 3.1 3.7
                        4.9 3.9 1.7 2.9 3.9
                        6.1
                     </swe:values>
                  </un:ParameterArray>
               </un:parameters>
            </un:MultivariateDistribution>
         </rv:representation>
      </rv:JointRandomVariable>
   </range>
```

```
</InterpolationResult>
```

Example 28: UncertML is used to describe the result of an interpolation as a joint distribution.

10 UncertML XML Schemata

10.1 UncertML.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:un="http://www.uncertml.org"
xmlns:swe="http://www.opengis.net/swe/1.0.1"
targetNamespace="http://www.uncertml.org" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <annotation>
      <documentation>[WARN-A001] - No package description in UML
model</documentation>
   </annotation>
   <import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <import namespace="http://www.opengis.net/swe/1.0.1"
schemaLocation="http://schemas.opengis.net/sweCommon/1.0.1/swe.xsd"/>
   <include schemaLocation="realisations.xsd"/>
   <include schemaLocation="baseTypes.xsd"/>
   <include schemaLocation="statistics.xsd"/>
   <include schemaLocation="distributions.xsd"/>
</schema>
```

10.2 baseTypes.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:un="http://www.uncertml.org"
targetNamespace="http://www.uncertml.org" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <!-- ========================================================== -->
   <!-- ====== Abstract Uncertainty ======-->
   <!-- ========================================================== -->
   <complexType name="AbstractUncertaintyType" abstract="true">
      <annotation>
         <documentation>
         Base type for UncertML - contains a single attribute definition
that is used to provide semantics to every concrete uncertainty instance.
The definition attribute may be any URI and it is anticipated that this URI
will refer to an entry in a dictionary
         </documentation>
      </annotation>
      <attribute name="definition" type="anyURI" use="optional">
         <annotation>
            <documentation>
            The definition is a URI that resloves to an entry in a
dictionary to provide semantics to an uncertainty type
            </documentation>
         </annotation>
      </attribute>
   </complexType>
   <!-- ..................... -->
   <element name="AbstractUncertainty" type="un:AbstractUncertaintyType"
abstract="true">
      <annotation>
         <documentation>
```

```
         Base type for UncertML. All types that may be used to represent
uncertainty substitute for this type
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="AbstractUncertaintyPropertyType">
      <sequence>
         <element ref="un:AbstractUncertainty"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <!-- ====== Parameter ======-->
   <!-- ========================================================= -->
   <element name="Parameter" type="un:ParameterType">
      <annotation>
         <documentation>A parameter can belong to a distribution or
statistic</documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="ParameterType">
      <sequence>
         <element name="value" type="anySimpleType" minOccurs="0">
            <annotation>
               <documentation>
               The value property contains the actual value of a particular
parameter
               </documentation>
            </annotation>
         </element>
      </sequence>
      <attribute name="definition" type="anyURI">
         <annotation>
            <documentation>
            The definition is a URI that resloves to an entry in a
dictionary to provide semantics to an uncertainty type
            </documentation>
         </annotation>
      </attribute>
   </complexType>
   <!-- ..................... -->
   <complexType name="ParameterPropertyType">
      <sequence>
         <element ref="un:Parameter" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <!-- ========================================================= -->
</schema>
```

10.3 statistics.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:un="http://www.uncertml.org"
xmlns:swe="http://www.opengis.net/swe/1.0"
targetNamespace="http://www.uncertml.org" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <!-- ========================================================= -->
   <!-- bring in other schemas-->
```

```
   <include schemaLocation="baseTypes.xsd"/>
   <import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <import namespace="http://www.opengis.net/swe/1.0"
schemaLocation="http://schemas.opengis.net/sweCommon/1.0.0/aggregateTypes.x
sd"/>
   <!-- ========================================================== -->
   <!-- ========================================================== -->
   <!-- ====== Statistic ======-->
   <!-- ========================================================== -->
   <element name="Statistic" type="un:StatisticType"
substitutionGroup="un:AbstractUncertainty">
      <annotation>
         <documentation>
         A Statistic is a generic type that allows the description of most
statistics including mean, variance, standard deviation, mode etc
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="StatisticType">
      <complexContent>
         <extension base="un:AbstractUncertaintyType">
            <sequence>
               <element name="parameters" minOccurs="0">
                  <annotation>
                     <documentation>A Statistic may have several
parameters</documentation>
                  </annotation>
                  <complexType>
                     <sequence>
                        <element ref="un:Parameter" maxOccurs="unbounded"/>
                     </sequence>
                  </complexType>
               </element>
               <element name="value" type="anySimpleType" minOccurs="0">
                  <annotation>
                     <documentation>
                     The value of a statistic varies with the type of
statistic.

                     When referring to a Statistic or Quantile the value
represents the value of that variable and is in the same units of measure
as specified by the variable.

                     Probability and DiscreteProbabilty types are different
in that the value of these types represents a probability (from 0.0 - 1.0
inclusive) and as such they do not have units of measure.

                     It is important that this distinction is understood
when working with Statistics
                     </documentation>
                  </annotation>
               </element>
            </sequence>
            <attribute name="degreesOfFreedom" type="int">
               <annotation>
                  <documentation>The degrees of freedom is the number of
independent pieces of information that go into the estimation of a
parameter, or statistic</documentation>
               </annotation>
```

```
            </attribute>
         </extension>
      </complexContent>
   </complexType>
   <!-- ..................... -->
   <complexType name="StatisticPropertyType">
      <sequence>
         <element ref="un:Statistic"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <!-- ====== Probability ======-->
   <!-- ========================================================= -->
   <element name="Probability" type="un:ProbabilityType"
substitutionGroup="un:Statistic">
      <annotation>
         <documentation>
         A Probability is a specific type of statistic that may require
several properties to specify bounds, e.g. the probability of a variable
being greater than 34 and less than 56 is 0.23
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="ProbabilityType">
      <complexContent>
         <extension base="un:StatisticType">
            <group ref="un:ProbabilityGroup"/>
         </extension>
      </complexContent>
   </complexType>
   <!-- ..................... -->
   <complexType name="ProbabilityPropertyType">
      <sequence>
         <element ref="un:Probability"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <!-- ====== Moment ======-->
   <!-- ========================================================= -->
   <element name="Moment" type="un:MomentType"
substitutionGroup="un:Statistic">
      <annotation>
         <documentation>
            A Moment is used to describe a random quantity. It may be
defined as a moment or a centred moment
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="MomentType">
      <complexContent>
         <extension base="un:StatisticType">
            <sequence>
               <element name="order" type="int">
                  <annotation>
                     <documentation>
                        The order of the moment. E.g. 1st moment, 2nd
moment...
                     </documentation>
                  </annotation>
```

```
            </element>
         </sequence>
      </extension>
   </complexContent>
</complexType>
<!-- ..................... -->
<complexType name="MomentPropertyType">
   <sequence>
      <element ref="un:Moment"/>
   </sequence>
</complexType>
<!-- ========================================================= -->
<!-- ====== DiscreteProbability ======-->
<!-- ========================================================= -->
<element name="DiscreteProbability" type="un:DiscreteProbabilityType"
substitutionGroup="un:Statistic">
   <annotation>
      <documentation>
      A DiscreteProbability is a more specific type of statistic that
requires a discrete category to be provided
      </documentation>
   </annotation>
</element>
<!-- ..................... -->
<complexType name="DiscreteProbabilityType">
   <complexContent>
      <extension base="un:StatisticType">
         <sequence>
            <element name="category" minOccurs="0">
               <annotation>
                  <documentation>
                  The category of a DiscreteProbability may be any type
but typically this will be a string representing a particular category
within a set. E.g. the probability that the NO2 levels in the United
Kingdom are high
                  </documentation>
               </annotation>
               <complexType>
                  <complexContent>
                     <extension base="anyType">
                        <attribute name="definition" type="anyURI"/>
                     </extension>
                  </complexContent>
               </complexType>
            </element>
         </sequence>
      </extension>
   </complexContent>
</complexType>
<!-- ..................... -->
<complexType name="DiscreteProbabilityPropertyType">
   <sequence>
      <element ref="un:DiscreteProbability"/>
   </sequence>
</complexType>
<!-- ========================================================= -->
<!-- ====== Quantile ======-->
<!-- ========================================================= -->
<element name="Quantile" type="un:QuantileType"
substitutionGroup="un:Statistic">
   <annotation>
```

```
         <documentation>
         A Quantile is a more specific type of statistic that requires a
level property be supplied to specify the particular quantile of interest,
e.g. 0.05 or 0.95
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="QuantileType">
      <complexContent>
         <extension base="un:StatisticType">
            <attribute name="level" use="optional">
               <annotation>
                  <documentation>
                  The level attribute specifies the quantile of interest.
This is a number that ranges from 0.0 -- 1.0 inclusive
                  </documentation>
               </annotation>
               <simpleType>
                  <restriction base="double">
                     <minInclusive value="0.0"/>
                     <maxInclusive value="1.0"/>
                  </restriction>
               </simpleType>
            </attribute>
         </extension>
      </complexContent>
   </complexType>
   <!-- ..................... -->
   <complexType name="QuantilePropertyType">
      <sequence>
         <element ref="un:Quantile"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <!-- ====== StatisticsRecord ======-->
   <!-- ========================================================= -->
   <element name="StatisticsRecord" type="un:StatisticsRecordType"
substitutionGroup="un:AbstractUncertainty">
      <annotation>
         <documentation>
         A StatisticsRecord groups several statistics together to form a
set of statistics that may provide a summary of a particular variable
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="StatisticsRecordType">
      <complexContent>
         <extension base="un:AbstractUncertaintyType">
            <sequence>
               <element name="field" type="un:AnyStatisticPropertyType"
minOccurs="0" maxOccurs="unbounded">
                  <annotation>
                     <documentation>
                     Each field represents another Statistic. It may be any
statistic from within the AnyStatistic union. This may be a StatisticsArray
or another StatisticsRecord
                     </documentation>
                  </annotation>
               </element>
```

```
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ..................... -->
   <complexType name="StatisticsRecordPropertyType">
      <sequence>
         <element ref="un:StatisticsRecord"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <!-- ====== StatisticsArray ======-->
   <!-- ========================================================= -->
   <element name="StatisticsArray" type="un:StatisticsArrayType"
substitutionGroup="un:AbstractUncertainty">
      <annotation>
         <documentation>
         A StatisticsArray encodes numerous values for a particular
statistic (or group of statistics) in an efficient encoding block
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="StatisticsArrayType">
      <complexContent>
         <extension base="un:AbstractUncertaintyType">
            <sequence>
               <element name="elementType"
type="un:AnyStatisticPropertyType">
                  <annotation>
                     <documentation>
                     The elementType defines the type of statistic encoded
within the values property. This may be any statistic from the AnyStatistic
union so you can have arrays of records and even arrays of arrays
                     </documentation>
                  </annotation>
               </element>
               <element name="elementCount" type="integer">
                  <annotation>
                     <documentation>
                     The elementCount property is an integer value that
contains the number of elements in the array
                     </documentation>
                  </annotation>
               </element>
               <group ref="swe:EncodedValuesGroup"/>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ..................... -->
   <complexType name="StatisticsArrayPropertyType">
      <sequence>
         <element ref="un:StatisticsArray"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <!-- ========================================================= -->
   <group name="AnyStatistic">
      <annotation>
         <documentation>
```

```
         A union type that provides a choice between any statistic - used
in StatisticsRecord and StatisticsArray
         </documentation>
      </annotation>
      <choice>
         <element ref="un:StatisticsRecord"/>
         <element ref="un:StatisticsArray"/>
         <element ref="un:Statistic"/>
      </choice>
   </group>
   <!-- ========================================================== -->
   <!-- ========================================================== -->
   <complexType name="AnyStatisticPropertyType">
      <sequence minOccurs="0">
         <group ref="un:AnyStatistic"/>
      </sequence>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
   </complexType>
   <!-- ========================================================== -->
   <!-- ========================================================== -->
   <group name="ProbabilityGroup">
      <annotation>
         <documentation>Group for probabilities containing greater than,
less than, equal to, greater than or equal to and less than or equal to.
All are optional and may be used in combination to create upper and lower
bounds
         </documentation>
      </annotation>
      <sequence>
         <element name="gt" type="double" minOccurs="0">
            <annotation>
               <documentation>
            The probability that a random quantity is greater than this
number. The acutal probability is contained within the value
property</documentation>
            </annotation>
         </element>
         <element name="lt" type="double" minOccurs="0">
            <annotation>
               <documentation>
            The probability that a random quantity is less than this
number. The actual probability is contained within the value
property</documentation>
            </annotation>
         </element>
         <element name="eq" type="double" minOccurs="0">
            <annotation>
               <documentation>
            The probability that a random quantity is equal to this number.
The actual probability is contained within the value property
            </documentation>
            </annotation>
         </element>
         <element name="ge" type="double" minOccurs="0">
            <annotation>
               <documentation>
            The probability that a random quantity is greater than or equal
to this number. The actual probability is contained within the value
property</documentation>
            </annotation>
         </element>
```

```
         <element name="le" type="double" minOccurs="0">
            <annotation>
               <documentation>
            The probability that a random quantity is less than or equal to
this number. The actual probability is contained within the value
property</documentation>
            </annotation>
         </element>
      </sequence>
   </group>
   <!-- ========================================================== -->
   <!-- ========================================================== -->
</schema>
```

10.4 realisations.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:un="http://www.uncertml.org"
xmlns:swe="http://www.opengis.net/swe/1.0"
targetNamespace="http://www.uncertml.org" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <!-- ========================================================== -->
   <!-- bring in other schemas-->
   <include schemaLocation="baseTypes.xsd"/>
   <import namespace="http://www.opengis.net/swe/1.0"
schemaLocation="http://schemas.opengis.net/sweCommon/1.0.0/aggregateTypes.x
sd"/>
   <!-- ========================================================== -->
   <!-- ========================================================== -->
   <!-- ====== Realisations types ======-->
   <!-- ========================================================== -->
   <element name="Realisations" type="un:RealisationsType"
substitutionGroup="un:AbstractUncertainty">
      <annotation>
         <documentation>
         A Realisations is used to group numerous samples into an efficient
encoding
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="RealisationsType">
      <complexContent>
         <extension base="un:AbstractUncertaintyType">
            <sequence>
               <element name="realisationsCount" type="int">
                  <annotation>
                     <documentation>realisationsCount specifies the number
of realisations for any given phenomena at any given domain
point</documentation>
                  </annotation>
               </element>
               <element name="elementCount" type="integer">
                  <annotation>
                     <documentation>
                     elementCount is an integer value that specifies the
number of realistaions contained within the values property
                     </documentation>
                  </annotation>
               </element>
```

```
            <group ref="swe:EncodedValuesGroup"/>
         </sequence>
         <attribute name="samplingMethod" type="anyURI"/>
         <attribute name="realisedFrom" type="anyURI"/>
      </extension>
   </complexContent>
</complexType>
<!-- ..................... -->
<complexType name="RealisationsPropertyType">
   <sequence>
      <element ref="un:Realisations"/>
   </sequence>
</complexType>
<!-- ========================================================= -->
<!-- ========================================================= -->
<group name="AnyRealisation">
   <choice>
      <element ref="un:Realisations"/>
   </choice>
</group>
<!-- ========================================================= -->
<!-- ========================================================= -->
</schema>
```

10.5 distributions.xsd

```
<schema xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:un="http://www.uncertml.org"
xmlns:swe="http://www.opengis.net/swe/1.0"
targetNamespace="http://www.uncertml.org" elementFormDefault="qualified"
attributeFormDefault="unqualified">
   <!-- ========================================================= -->
   <!-- bring in the other schemas-->
   <include schemaLocation="baseTypes.xsd"/>
   <import namespace="http://www.opengis.net/gml"
schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/gml.xsd"/>
   <import namespace="http://www.opengis.net/swe/1.0"
schemaLocation="http://schemas.opengis.net/sweCommon/1.0.0/aggregateTypes.x
sd"/>
   <!-- ========================================================= -->
   <!-- ========================================================= -->
   <!-- ====== Distribution ======-->
   <!-- ========================================================= -->
   <element name="Distribution" type="un:DistributionType"
substitutionGroup="un:AbstractUncertainty">
      <annotation>
         <documentation>
         A Distribution is used to represent any parameteric distribution.
The definition attribute links to a dictionary that provides details about
the cumulative distribution function.
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="DistributionType">
      <complexContent>
         <extension base="un:AbstractUncertaintyType">
            <sequence>
               <element name="parameters" type="un:ParameterPropertyType"
minOccurs="0">
```

```
                  <annotation>
                     <documentation>
                     Most distributions contain 1 or more parameters that
define a specific instance of that distribution
                     </documentation>
                  </annotation>
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ..................... -->
   <complexType name="DistributionPropertyType">
      <sequence>
         <element ref="un:Distribution"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <!-- ====== MultivariateDistribution ======-->
   <!-- ========================================================= -->
   <element name="MultivariateDistribution"
type="un:MultivariateDistributionType"
substitutionGroup="un:AbstractUncertainty">
      <annotation>
         <documentation>
         A MultivariateDistribution is used for joint distributions.
ParameterArrays are utilised to encode the numerous parameter values
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="MultivariateDistributionType">
      <complexContent>
         <extension base="un:AbstractUncertaintyType">
            <sequence>
               <element name="parameters"
type="un:ParameterArrayPropertyType" minOccurs="0">
                  <annotation>
                     <documentation>
                     The parameters property of a MultivariateDistribution
contains multiple arrays of parameters. In the instance of a multivariate
Gaussian distribution this would be an array of mean values and an array of
covariance values
                     </documentation>
                  </annotation>
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ..................... -->
   <complexType name="MultivariateDistributionPropertyType">
      <sequence>
         <element ref="un:MultivariateDistribution"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <!-- ====== ParameterArray ======-->
   <!-- ========================================================= -->
   <element name="ParameterArray" type="un:ParameterArrayType">
      <annotation>
```

```
         <documentation>
         A ParameterArray is used by MultivariateDistributions and contains
an efficient encoding for numerous values of the same parameter
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="ParameterArrayType">
      <sequence>
         <element name="elementType" type="un:ParameterPropertyType">
            <annotation>
               <documentation>
               The elementType contains a Parameter type that defines the
parameter that is encoded within the values property
               </documentation>
            </annotation>
         </element>
         <element name="elementCount" type="integer">
            <annotation>
               <documentation>
               The elementCount property is an integer value that defines
the number of elements within the values property
               </documentation>
            </annotation>
         </element>
         <group ref="swe:EncodedValuesGroup"/>
      </sequence>
   </complexType>
   <!-- ..................... -->
   <complexType name="ParameterArrayPropertyType">
      <sequence>
         <element ref="un:ParameterArray" maxOccurs="unbounded"/>
      </sequence>
   </complexType>
   <!-- ========================================================== -->
   <!-- ====== MixtureModel ======-->
   <!-- ========================================================== -->
   <element name="MixtureModel" type="un:MixtureModelType"
substitutionGroup="un:AbstractUncertainty">
      <annotation>
         <documentation>
         A MixtureModel is like a collection of multiple distributions with
each one having a relative influence on the variable
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="MixtureModelType">
      <complexContent>
         <extension base="un:AbstractUncertaintyType">
            <sequence>
               <element name="weights" type="gml:doubleList" minOccurs="0">
                  <annotation>
                     <documentation>
                     The weights property contains a list of double values
that represent the relative weights of each distribution instance. There
should be the same number of values within the weights property as there
are distributions within the distributions property. Extra care should be
taken when working with multivariate distributions
                     </documentation>
                  </annotation>
```

```
               </element>
               <element name="distributions"
type="un:AnyDistributionPropertyType" minOccurs="0">
                  <annotation>
                     <documentation>
                     The distributions property contains 1 or more
distribution types from the AnyDistribution union which includes
Distributions, MultivariateDistributions and DistributionArrays
                     </documentation>
                  </annotation>
               </element>
            </sequence>
         </extension>
      </complexContent>
   </complexType>
   <!-- ..................... -->
   <complexType name="MixtureModelPropertyType">
      <sequence>
         <element ref="un:MixtureModel"/>
      </sequence>
   </complexType>
   <!-- ========================================================= -->
   <!-- ====== DistributionArray ======-->
   <!-- ========================================================= -->
   <element name="DistributionArray" type="un:DistributionArrayType"
substitutionGroup="un:AbstractUncertainty">
      <annotation>
         <documentation>
         A DistributionArray is used to encode multiple distributions of
the same type
         </documentation>
      </annotation>
   </element>
   <!-- ..................... -->
   <complexType name="DistributionArrayType">
      <complexContent>
         <extension base="un:AbstractUncertaintyType">
            <sequence>
               <element name="elementType"
type="un:AnyDistributionPropertyType">
                  <annotation>
                     <documentation>
                     The elementType property contains the distribution
that is encoded within the values property. The distribution within this
property is taken from the AnyDistribution union and can be a Distribution,
MultivariateDistribution or another DistributionArray
                     </documentation>
                  </annotation>
               </element>
               <element name="elementCount" type="integer">
                  <annotation>
                     <documentation>
                     The elementCount property is an integer value that
represents the number of elements within the values property
                     </documentation>
                  </annotation>
               </element>
               <group ref="swe:EncodedValuesGroup"/>
            </sequence>
         </extension>
      </complexContent>
```

```
   </complexType>
   <!-- ..................... -->
   <complexType name="DistributionArrayPropertyType">
      <sequence>
         <element ref="un:DistributionArray"/>
      </sequence>
   </complexType>
   <!-- ========================================================== -->
   <!-- ========================================================== -->
   <group name="AnyDistribution">
      <annotation>
         <documentation>
         The AnyDistribution union is used by the DistributionArray and
MixtureModel types
         </documentation>
      </annotation>
      <choice>
         <element ref="un:MultivariateDistribution"/>
         <element ref="un:Distribution"/>
         <element ref="un:DistributionArray"/>
         <element ref="un:MixtureModel"/>
      </choice>
   </group>
   <!-- ========================================================== -->
   <!-- ========================================================== -->
   <complexType name="AnyDistributionPropertyType">
      <sequence maxOccurs="unbounded">
         <group ref="un:AnyDistribution"/>
      </sequence>
      <attributeGroup ref="gml:AssociationAttributeGroup"/>
   </complexType>
   <!-- ========================================================== -->
   <!-- ========================================================== -->
</schema>
```