Describing and Communicating Uncertainty within the Semantic Web

Matthew Williams[1] (williamw@aston.ac.uk), Dan Cornford[1], and Lucy Bastin[1]

Knowledge Engineering Group, Aston University, Birmingham, United Kingdom

Abstract. The Semantic Web relies on carefully structured, well defined, data to allow machines to communicate and understand one another. In many domains (e.g. geospatial) the data being described contains some uncertainty, often due to incomplete knowledge; meaningful processing of this data requires these uncertainties to be carefully analysed and integrated into the process chain. Currently, within the Semantic Web there is no standard mechanism for interoperable description and exchange of uncertain information, which renders the automated processing of such information implausible, particularly where error must be considered and captured as it propagates through a processing sequence. In particular we adopt a Bayesian perspective and focus on the case where the inputs / outputs are naturally treated as random variables.
This paper discusses a solution to the problem in the form of the Uncertainty Markup Language (UncertML). UncertML is a conceptual model, realised as an XML schema, that allows uncertainty to be quantified in a variety of ways i.e. realisations, statistics and probability distributions. UncertML is based upon a soft-typed XML schema design that provides a generic framework from which any statistic or distribution may be created. Making extensive use of Geography Markup Language (GML) dictionaries, UncertML provides a collection of definitions for common uncertainty types. Containing both written descriptions and mathematical functions, encoded as MathML, the definitions within these dictionaries provide a robust mechanism for defining any statistic or distribution and can be easily extended. Universal Resource Identifiers (URIs) are used to introduce semantics to the soft-typed elements by linking to these dictionary definitions.
The INTAMAP (INTeroperability and Automated MAPping) project provides a use case for UncertML. This paper demonstrates how observation errors can be quantified using UncertML and wrapped within an Observations & Measurements (O&M) Observation. The interpolation service uses the information within these observations to influence the prediction outcome. The output uncertainties may be encoded in a variety of UncertML types, e.g. a series of marginal Gaussian distributions, a set of statistics, such as the first three marginal moments, or a set of realisations from a Monte Carlo treatment. Quantifying and propagating uncertainty in this way allows such interpolation results to be consumed by other services. This could form part of a risk management chain or a decision support system, and ultimately paves the way for complex data processing chains in the Semantic Web.

1 Introduction

As the Semantic Web evolves, increasing quantities of data are being formatted to allow distribution, discovery and consumption by machines operating over networks. This approach requires clear conceptualisation of real-world objects and phenomena, their attributes and relationships, to allow rich datasets to be fully exploited by automated parsers and processes. Uncertainty in measurement (for example of objects' bounds and parameters) is sometimes considered as a part of the metadata taxonomy, but rarely in any significant detail, and currently no uniform standard exists for capturing and communicating the errors and uncertainties which are inherent in almost all real-world datasets. We would argue that data without quantified uncertainty has no effective information content, and believe that a Bayesian probabilistic framework offers a complete, principled mechanism to conceptualise this.

Currently, there is a trend in software engineering to move away from tightly coupled legacy systems and towards loosely coupled, interoperable, services [1] based on XML. The Web Services approach, whereby functionality is exposed and consumed over networks, is a particular context where standardised descriptions of capabilities and outputs ensures interoperability, and allows data to be passed sequentially through Services in processing chains. As Semantic Web Services evolve, these descriptions will become richer, but even now there is a need for uncertainty information to be passed between and 'understood' by automated processes. This is especially important where error propagates through a processing sequence — for example, in the case of automatically monitored and interpolated temperature data, where sensor error and random noise in the original measurements can combine with artefacts from the techniques used to characterise and interpolate the data, to produce significant levels of posterior uncertainty. This uncertainty should ideally be explicitly estimated and quantified, either as simple means and variances or as fully-characterised probability distributions, over all inputs, parameters and the final outputs. It is also critical to communicate data uncertainty where the outputs are to be used for decision-making — for example, where national radiation data is used to plan for evacuations after a critical incident. In this case, the uncertainty in predicted radiation at any location might be represented as exceedance probabilities, showing the probability that a critical threshold is exceeded at any location, or as sets of realised samples from the predicted distribution.

The above two examples have a spatial component, and indeed many of the existing Web Service standards are designed to handle geospatial data (for example, the OGC Web Coverage Service, Web Feature Service and Web Processing Service standards). It is also the case that error propagation in geospatial data and geostatistics has been particularly well documented, particularly in natural resources and decision making contexts [2–4]. However, there is a pressing need, in the context of the Semantic Web, to represent uncertainty far more generically, using a clear and flexible standard which can be incorporated into a variety of existing ontologies and schemata. Our proposal is UncertML, an XML schema designed for communicating uncertainty in an interoperable way, based on a conceptual model which allows data uncertainty to be flexibly represented in combination with any other structured data model, including commonly-used XML schemata such as O&M (Observations and Measurements) and GML (Geography Markup Language). These uncertainty representations currently include sets of summary statistics, marginal or joint distributions, and sets of realisations generated by sampling, and it is anticipated that they will be extended to other representations such as fuzzy sets.

In order to maintain flexibility and extensibility within UncertML, we have made considerable use of Uniform Resource Identifiers (URIs) in combination with a weak-typed design pattern to allow elements such as statistical distributions and algorithmic sampling techniques to be fully described in dictionaries, rather than encoded as concrete types. These dictionaries could be written in GML (the current option within UncertML), Resource Definition Framework (RDF) or Web Ontology Language (OWL). This paper describes the conceptual model for UncertML, with examples of how one might encode uncertainty in XML, illustrated by examples arising within the INTAMAP project.

2 UncertML Conceptual Model

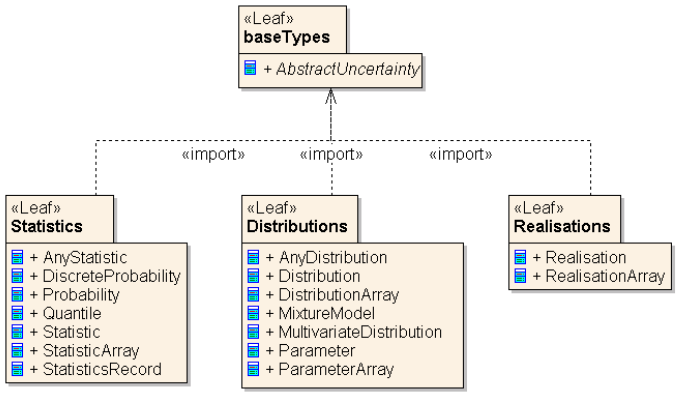

Fig. 1. Package overview of UncertML. Each package contains a set of elements for describing uncertainty.

UncertML is divided into three distinct packages. Each package is tailored toward describing uncertainty using a specific mechanism; either through realisations, statistics or probability distributions. Sections 2.1– 2.3 introduce the conceptual outline for each package and discuss the component types.

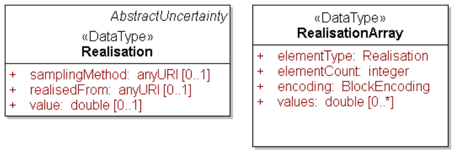

Fig. 2. Realisations can either be encoded singly using the `Realisation` type or aggregated in the `RealisationArray`.

2.1 Realisations

In some situations the user may not be able to parametrically describe uncertainties in their data. Typically, in such a situation they may provide a sample, often using Markov Chain Monte Carlo methods, from the probability distribution of the data which allows the uncertainties to be described implicitly. However, a sufficiently large sample of data is required to properly assess uncertainty, therefore efficient encapsulation of large data volumes is an important issue for UncertML.

Realisation. As with all uncertainty types, a `Realisation` inherits the `definition` property from the `AbstractUncertainty` type. In this instance the URI should resolve to a definition of the concept of a realisation. Greater information about any particular realisation may be included with the `realisedFrom` and `samplingMethod` properties. Both these properties are URIs that link to dictionaries, providing information about the distribution the sample was realised from and the method by which the data was sampled, respectively. The final property of a `Realisation` is the `value`. This property contains the actual value of the realisation; i.e. the number generated by the sampling mechanism.

RealisationArray. Working with large arrays of realisations is more common practice, since we are often dealing with joint distributions. UncertML provides the `RealisationArray` type for such purposes. As with all other array types in UncertML, the `RealisationArray` is based around the SWE Common `DataArray` type [5]. The `elementType` property describes the element that is contained within the array, in this instance it is a `Realisation`. The `elementCount` property is an integer value that defines the number of elements, or realisations, within the array.

The SWE Common encoding schema 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`, or realisations.

2.2 Statistics

There is an extensive range of options available in UncertML for describing 'summary statistics'. Such statistics are used to provide a summary of a random variable, ranging from measures of location (mean, mode, median etc) to measures of dispersion (range, standard deviation, variance etc). 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. It should be noted that providing a location value which is explicitly defined as the mean conveys significantly more information than simply providing a value, since the value might represent many things including the mean, mode, median or even a realisation.

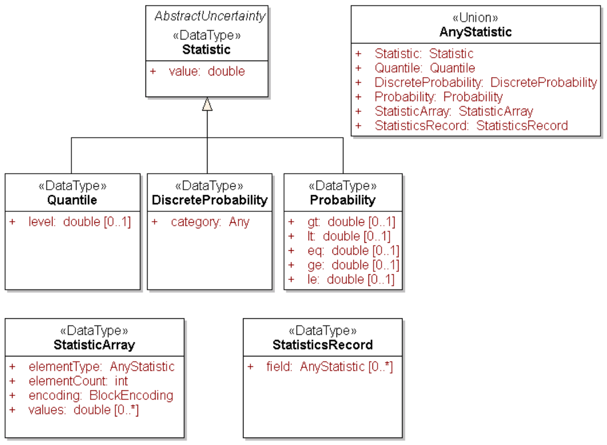

Fig. 3. UncertML model for summary statistics.

The `Statistic` type extends the `AbstractUncertainty` type, inheriting a `definition` property, which in this instance should resolve to a definition of the particular statistic, e.g. mean, variance or mode etc. The other property of a `Statistic` is the `value` which contains the actual value of the statistic, encoded as a double. This generic and concise concept of a statistic allows *most* statistics to be encoded, but for certain statistics more information is required.

One such example is a quantile; here the user needs to know which quantile is being referred to. UncertML provides a specific `Quantile` type which extends the `Statistic` type and provides an additional property, `level`. Continuous and discrete probabilities follow a similar pattern; extending the `Statistic` type with additional properties, and allowing encoding of histograms, exceedance probabilities and discrete random variables.

StatisticsRecord. A grouped set of summary statistics provides a mechanism for summarising a particular variable's uncertainty. 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 [5].

A `StatisticsRecord` consists of a number of `field` properties. Each `field` of a `StatisticsRecord` may be a `Statistic`, `Quantile`, `DiscreteProbability`, `Probability`, `StatisticsArray` or `StatisticsRecord`.

StatisticsArray. Arrays of statistics are useful when describing a variable at several locations, or several variables at a given location. The `StatisticsArray` type in UncertML, closely modelled on the `DataArray` of SWE Common, provides such a mechanism. Unlike the `RealisationArray` type, the `elementType` property of a `StatisticsArray` may be any type from within the `AnyStatistic` union. This flexibility allows arrays of single statistics, or an array of `Statistics-Records` to provide multiple summaries. More complex structures such as two dimensional arrays are also possible.

2.3 Distributions

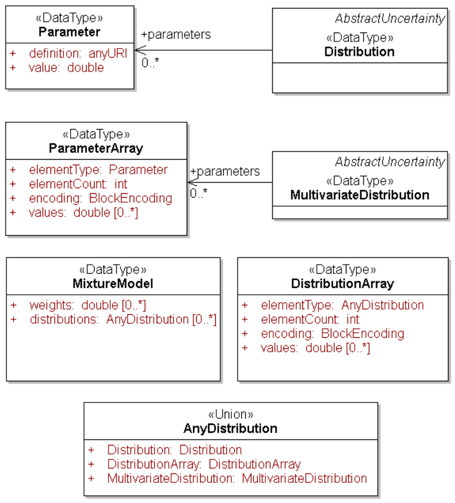

Fig. 4. Distributions in UncertML are encoded using one of the types above.

When the uncertainties of a dataset are more clearly 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 encapsulation of *all* probability distributions without sacrificing the simplicity of UncertML.

Distribution. In the simplest case, where a user wishes to describe the probability distribution of a single variable, UncertML provides the `Distribution` type. In the case of distributions the definition may contain both a textual description, and a complex mathematical description of the distribution's functions (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.

Complementing the `definition` property is a `parameters` property that contains a number of `Parameter` types. Each `Parameter` of a distribution is not considered to be an uncertainty type, however, 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.

DistributionArray. The `DistributionArray` type is similar to both the `StatisticsArray` and `RealisationArray`. However, in this instance the `elementType` property is realised as a type from the `AnyDistribution` union. The rest of the properties remain the same as in the `StatisticsArray` & `RealisationArray`, but one subtle difference exists. Distributions often have numerous parameters that help describe them (e.g. a Gaussian distribution has both a mean and a variance parameter). In this instance the `Distribution` contained within the `elementType` property acts as a form of 'record'. Therefore, when encoding the distributions within the `values` property, care should be taken in interpretation to clearly understand which values refer to which parameter.

MixtureModel. A `MixtureModel` is a specialised form of record. When describing a variable using a mixture of distributions, a specific weight is assigned to each distribution specifying the relative importance of that distribution. 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.

MultivariateDistribution. The final type provided by UncertML is the `MultivariateDistribution` type. A typical use case for a multivariate (or joint) distribution is when two variables are correlated. As this scenario (usually) requires the inclusion of a covariance matrix the `DistributionArray` is not sufficient to describe the variable.

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.

3 XML Examples

In this section we demonstrate how the UncertML schemata can be used to encode uncertainty in a variety of ways.

3.1 Realisations

```xml
<un:RealisationArray>
   <un:elementType>
      <un:Realisation definition="http://www.intamap.org/
         uncertml/realisations" realisedFrom="http://www.
         intamap.org/uncertml/distributions/gaussian"
         samplingMethod="http://www.intamap.org/uncertml/
         realisations/methods/directSampling">
      </un:Realisation>
   </un:elementType>
   <un:elementCount>5</un:elementCount>
   <swe:encoding>
      <swe:TextBlock decimalSeparator="." blockSeparator="␣"
         tokenSeparator=","/>
   </swe:encoding>
   <swe:values>45.6 47.6 87.6 23.4 35.6</swe:values>
</un:RealisationArray>
```

Listing 1.1. An array of realisations may be encoded using the `RealisationArray` type.

This example demonstrates how a sample of data can be encoded as a `RealisationArray`. 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. For more information about the other forms of encoding available via the SWE Common `EncodedValuesGroup` we refer to the SWE Common specification [5].

3.2 Statistics

Due to the soft-typed approach of UncertML all simple statistics will look identical. What separates a ‘mean’ from a ‘median’ is the URI (and definition upon resolving) of the `definition` property. 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://www.intamap.org/uncertml/
    statistics/mode">
   <un:value>34.67</un:value>
</un:Statistic>
```

Listing 1.2. A `Statistic` type in UncertML maintains a very simplistic structure.

Grouping statistics into a single structure can be an efficient mechanism for describing the uncertainty surrounding a particular variable. An example of a `StatisticsRecord` may be seen in Listing 1.3, demonstrating how the mean and the probability that a variable exceeds a certain threshold can be grouped.

```
<un:StatisticsRecord>
   <un:field>
      <un:Statistic definition="http://www.intamap.org/
          uncertml/statistics/mean">
         <un:value>34.5</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>
```

Listing 1.3. `StatisticsRecord`s allow individual statistics to be grouped into a meaningful structure.

3.3 Distributions

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 a complex notion such as a Gaussian distribution to be encoded in a simple framework. Listing 1.4 demonstrates the ease with which such a distribution may be encoded.

```
<un:Distribution definition="http://www.intamap.org/uncertml/
    distributions/gaussian">
   <un:parameters>
      <un:Parameter definition="http://www.intamap.org/
          uncertml/distributions/gaussian/mean">
```

```
            <un:value>34.564</un:value>
        </un:Parameter>
        <un:Parameter definition="http://www.intamap.org/
            uncertml/distributions/gaussian/variance">
            <un:value>67.45</un:value>
        </un:Parameter>
    </un:parameters>
</un:Distribution>
```

Listing 1.4. A Gaussian distribution is simple to encode in UncertML using just the two parameters mean and variance.

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

4 Integrating UncertML into Existing Taxonomies - the INTAMAP Example

The INTAMAP project aims to provide sophisticated functionality across the Web, exposing data cleaning, outlier detection and geostatistical interpolation functions via a Web Processing Service. The approach prioritises interoperability, with a particular focus on the future consumption of data from automatic monitoring networks via Sensor Observation Services. Our specific case study involves the processing of radiation data from stations across Europe (the European Radiological Data Exchange Platform (EURDEP)) whose spacing, sensitivity and error characteristics are patchy and heterogeneous, and real-time prediction of radiation values to unknown locations between the sampling locations by specialised methods such as Projected Process Kriging [6]. In this context, it is vital that the uncertainty in the monitoring data and the predicted outputs is clearly and fully characterised and communicated.

Several XML schemata exist which are of value in representing this data as it is collected: The Observations & Measurements schema [7] allows results recorded from a sensing instrument to be encoded along with information on the observation time, the specific phenomenon being observed and the spatial extent of the feature of interest. Two important pieces of XML can be used as property values to enrich the information encoded in an Observation. Firstly, an UncertML type, rather than a simple value, can be given as the 'result' property of the Observation, to describe the uncertainty inherent in observed values. This allows a wide range of uncertainty information to be supplied, from a simple marginal mean and variance to a joint distribution with full covariance information. Secondly, the 'procedure' property will typically contain a sensor model encoded in SensorML [5] allowing users a fuller understanding of the physical methods by which the observation was collected.

The INTAMAP Web Processing Service interface accepts requests for interpolation, each of which includes a collection of observations, encoded in the

O&M schema. The availability of both the error characteristics of a sensor and the observation uncertainty in a machine-parsable form allows us to employ flexible, powerful techniques that take into account the different characteristics and uncertainties, based on a Bayesian framework, to perform the interpolation request. Depending on user preferences made in the request, the result of an interpolation can take several forms. The bulk of the data will be encoded in any one of the uncertainty types within UncertML and additional information may be added by separate schemata. A typical result may consist of a regular grid, possibly defined in GML [8], of some variable defined by a series of Gaussian distributions encoded in UncertML.

In the INTAMAP example, geographic information (usually in the form of GML) is added to the Observation as a separate layer. Uncertainty in the spatial location of an Observation could, in theory, be added by nesting UncertML records within an adapted form of GML. Our intention is to make UncertML generic and usable within a large variety of applications, which can replace existing value types with UncertML types.

5 Conclusion

As the Semantic Web and Web Services evolve into a loosely coupled, interoperable framework, sophisticated processing functions such as the geo-processing example described here will become more widely available, along with detailed and rich datasets for analysis. Machine-readable summaries of data quality will become increasingly important, both as a metric on which 'discovered' datasets can be judged for their suitability, and as statistical inputs into analyses where the risks of being wrong need to be quantified. Already, sophisticated users recognise that a single summary of error or precision across an entire dataset (for example, the Root Mean Square registration error commonly supplied as the 'accuracy' metadata on a registered aerial photograph) is rarely representative, and that excellent use may be made of more detailed error estimates, stratified by time, space, measurement instrument or even by individual measurements. In order to filter and judge the wealth of data which will become available via the Web in coming years, clear and standardised semantic descriptions of data uncertainty are vital, and we believe that UncertML can fulfil this need.

However, for true interoperability, several areas require greater attention. A conceptual model for extending the use of UncertML to random functions is under way, and further work on conditional distributions (or graphical models / belief networks) is envisaged. Other extensions to the UncertML model will include the addition of fuzzy memberships.

Currently, we are undergoing discussions with the Open Geospatial Consortium with the view of making the UncertML specification an official, governed, standard. A working interpolation service using UncertML will be available for testing online shortly. More information and latest developments can be found at the INTAMAP website (http://www.intamap.org).

Acknowledgements

This work is funded by the European Commission, under the Sixth Framework Programme, by Contract 033811 with DG INFSO, action Line IST-2005-2.5.12 ICT for Environmental Risk Management.

References

1. Erl, T.: Service-Oriented Architecture : Concepts, Technology, and Design. Prentice Hall PTR (August 2005)
2. Atkinson, P.M.: Geographical information science: geostatistics and uncertainty. Progress in Physical Geography **23** (1999) 134–142
3. Couclelis, H.: The Certainty of Uncertainty: GIS and the Limits of Geographic Knowledge. Transactions in GIS **7**(2) (2003) 165–175
4. Heuvelink, G.B.M., Goodchild, M.F.: Error Propagation in Environmental Modelling with GIS. CRC Press (1998)
5. Botts, M., Robin, A.: OpenGIS Sensor Model Language (SensorML) Implementation Specification. OpenGIS standard 07-000, Open Geospatial Consortium Inc (July 2007) http://www.opengeospatial.org/standards/sensorml.
6. Ingram, B., Cornford, D., Evans, D.: Fast algorithms for automatic mapping with space–limited covariance functions. Stochastic Environmental Research and Risk Assessment (2007)
7. Cox, S.: Observations and Measurements – Part 1 - Observation schema. OpenGIS standard 07-022r1, Open Geospatial Consortium Inc (December 2007) http://www.opengeospatial.org/standards/om.
8. Portele, C.: OpenGIS Geography Markup Language (GML) Encoding Standard. OpenGIS standard 07-036, Open Geospatial Consortium Inc (August 2007) http://www.opengeospatial.org/standards/gml.