Exchanging uncertainty. Interoperable geostatistics?

Matthew Williams, Dan Cornford, Lucy Bastin, Ben Ingram

Knowledge Engineering Group, School of Engineering and Applied Science, Aston University, Birmingham B4 7ET, UK

Summary. Traditionally, geostatistical algorithms are contained within specialist GIS and spatial statistics software. Such packages are often expensive, with relatively complex user interfaces and steep learning curves, and cannot be easily integrated into more complex process chains. In contrast, Service Oriented Architectures (SOAs) promote interoperability and loose coupling within distributed systems, typically using XML (eXtensible Markup Language) and Web services. Web services provide a mechanism for a user to discover and consume a particular process, often as part of a larger process chain, with minimal knowledge of how it works. Wrapping current geostatistical algorithms with a Web service layer would thus increase their accessibility, but raises several complex issues.

This paper discusses a solution to providing interoperable, automatic geostatistical processing through the use of Web services, developed in the INTAMAP project (INTeroperability and Automated MAPping). The project builds upon Open Geospatial Consortium standards for describing observations, typically used within sensor webs, and employs Geography Markup Language (GML) to describe the spatial aspect of the problem domain. Thus the interpolation service is extremely flexible, being able to support a range of observation types, and can cope with issues such as change of support and differing error characteristics of sensors (by utilising descriptions of the observation process provided by SensorML).

XML is accepted as the de facto standard for describing Web services, due to its expressive capabilities which allow automatic discovery and consumption by 'naïve' users. Any XML schema employed must therefore be capable of describing every aspect of a service and its processes. However, no schema currently exists that can define the complex uncertainties and modelling choices that are often present within geostatistical analysis. We show a solution to this problem, developing a family of XML schemata to enable the description of a full range of uncertainty types. These types will range from simple statistics, such as the kriging mean and variances, through to a range of probability distributions and non-parametric models, such as realisations from a conditional simulation. By employing these schemata within a Web Processing Service (WPS) we show a prototype moving towards a truly interoperable geostatistical software architecture.

1 Introduction

Uncertainty in geographic information is ubiquitous, be it from measurement error, observation operator error or modelling error. It is how we process and propagate this uncertainty that is of importance, especially when high-risk decisions are to be made based on such information [2, 5, 10]. In the field of geostatistics, uncertainty from multiple sources is routinely encountered. Consider, for example, a user in the field collecting soil samples. Inputting the data onto a small footprint machine (e.g. PDA) the user is able to store the data for lab processing, or alternatively, to submit the data for processing to a Web service. A typical process in this scenario might use the available data to predict where the user should optimally next sample, or to provide an estimate of the soil properties at an unsampled location. Errors in the original measurements, stemming from systematic sensor effects and random fluctuations, will combine with errors in the models used to process and interpolate the data, to produce significant levels of uncertainty (which must be explicitly estimated and quantified) in the final predictions. Traditionally, the soil data in this example would be processed from start to finish within a single software package to produce, for example, an interpolated map of heavy metal concentration, with estimation uncertainty represented as variance at each predicted location. The uncertainty in prediction might also be crystallised as exceedance probabilities, showing the likelihood that a critical threshold is exceeded at any location, or as sets of realised samples from the predicted distribution. While traditional geostatistical applications recognise and model the uncertainty at the end of the analysis, a conceptual model for describing and communicating uncertainties is of less importance, since the data is not usually shared with other applications. Uncertainty at the intermediate stages of analysis is therefore rarely explicitly characterised. However, if different processing steps (e.g. outlier detection, data harmonisation, parameter estimation, interpolation) are delegated to separate Web services, it becomes necessary for each service to receive an understandable summary of the uncertainty inherent in the sample data, and introduced by the intervening processing steps. Currently, there is a trend in software engineering to move away from tightly coupled legacy systems and towards loosely coupled, interoperable, services [8] based on XML. A conceptual design which allows the communication of uncertain results is of foremost importance in the development of such an interoperable geostatistical application. This paper introduces a conceptual model of uncertainty and examples of how one might encode uncertainty in XML, motivated by examples arising within the INTAMAP project.

2 XML, Web Services and SOAs

Interoperability is defined as "the ability of two or more systems or components to exchange information and to use the information that has been exchanged" [1]. This section provides an overview of several technologies and concepts that provide the foundations of an 'interoperable' application.

XML [15] is a structured language that allows metadata to be integrated with content, thus adding a layer of intelligence to information [7]. XML is implemented by defining a set of *elements* and *attributes* that are unique to a particular context, or domain. A collection of such elements and attributes is often referred to as a *vocabulary.* Vocabularies can be defined formally using a *schema definition language*, typically 'XML Schema' language [9], but is not a requirement of XML. The descriptive nature and extensibility of XML are two key ingredients that contribute towards it being a suitable language for interoperability.

The concept of a 'service' in software engineering is not a new term and typically refers to an independent building block within a larger application environment, or distributed system [7]. A Web service is an implementation of a service that uses XML to describe the operations available including the data inputs and outputs. There are other types of Web service (RESTful) which do not rely so heavily on XML, however, we do not discuss these in this paper and from hereon the term *Web service* refers specifically to an XML Web service.

Communication of data to and from a Web service is encoded as XML and transported via an Internet protocol (this is usually HTTP). Adhering to these requirements provides an interoperable framework that allows software applications, written in different languages and on different platforms, to communicate seamlessly. A collection of these services, 'loosely coupled', forms the basis of a design philosophy called Service Oriented Architecture.

The term Service Oriented Architecture has many definitions, perhaps one or the more concise is defined in [12] as:

> "SOA is an architectural paradigm for dealing with business processes distributed over a large landscape of existing and new heterogeneous systems that are under the control of different owners."

A SOA is usually realised as a collection of Web services, that may be governed by different owners, communicating with one another to form a processing chain. In context this could be a risk management or decision support chain.

3 The INTAMAP Project

Introducing interoperability into the field of geostatistics, INTAMAP seeks to provide a fully automated interpolation service implementing a Web Processing Service interface [14]. A WPS is a restriction on a normal Web service, governed by the Open Geospatial Consortium, that is suited to processing of geospatial data. Simply, a Web Processing Service can be thought of as a function that can be called over the Web. Within INTAMAP we are also

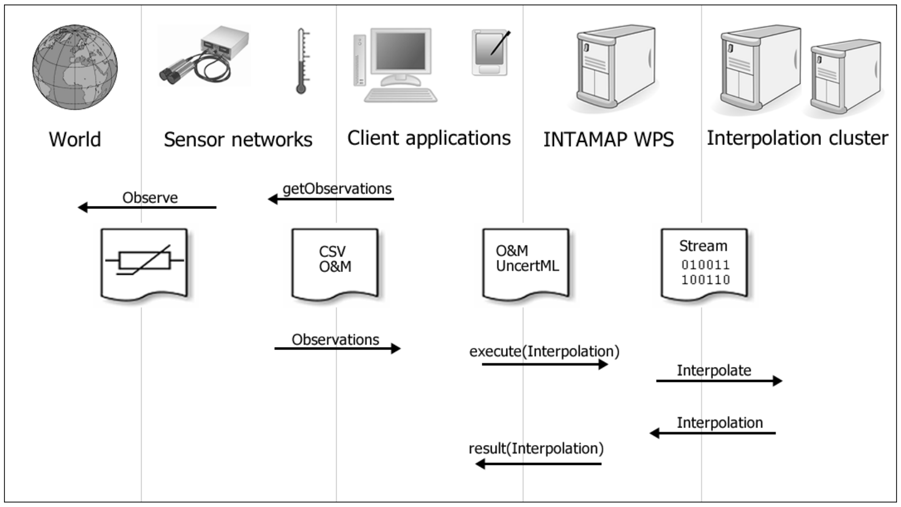

Fig. 1. Example workflow for an interpolation request in the INTAMAP project. All geostatistical processes are carried out using R and C++ on a separate server to the WPS. A client may obtain observations from multiple sensor systems before submitting them to INTAMAP for processing. Clients may also be services; chaining of services in this way underpins the foundation of Service Oriented Architectures

developing a range of novel automatic mapping algorithms including Bayesian trans-Gaussian kriging, fast anisotropy detection, data harmonisation for heterogeneous networks and fast approximate techniques that can deal with multiple sensor and error characteristics [11]. Key to all the methods we employ is a description of the uncertainties on the inputs and outputs of the interpolation process. Currently no such XML vocabulary, or *schema*, exists to allow the description of uncertainty, hence our development of UncertML. The inputs to the INTAMAP Web service are XML files describing the observations, with UncertML being used to characterise the observation errors (see Section 5). The results produced by INTAMAP contain inherent, and additional, uncertainty introduced by the interpolation process which must be communicated for the results to be of any subsequent utility. The rest of this paper discusses a solution to the problem, UncertML, and investigates the integration into INTAMAP, providing ‘interoperable geostatistics’.

4 Describing Uncertainty in XML

In this section we discuss the design of an XML language for describing uncertainty, UncertML, depicted using the Unified Modeling Language (UML). The UML diagrams used within this paper are static structure diagrams, whose notation is clearly defined in Sect. 5.4 of [13]. Examples in XML are given, where necessary, to illustrate how it may be used.

4.1 Conceptual Model and examples

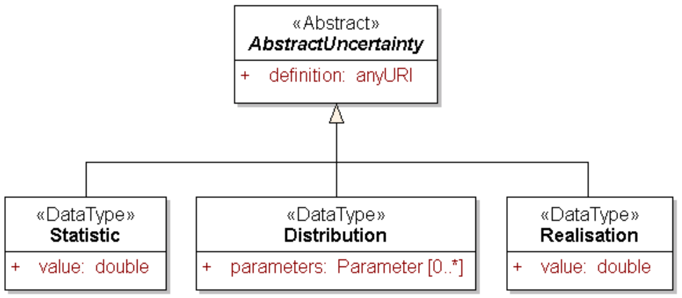

Fig. 2. Conceptual overview of UncertML. Three main types extend the abstract uncertainty type; 'Statistic', 'Distribution' and 'Realisation'. Other types are also available and discussed in more detail later

The core design of UncertML is split into three distinct sections; summary statistics, distributions and realisations (Fig. 2). Aggregate types for statistics, distributions and realisations also exist where deemed necessary. It is important to note that UncertML does not provide a framework for describing phenomena or their units of measure, nor does it provide any geospatial attributes. Removing this level of detail allows UncertML to be integrated into a diverse range of domains.

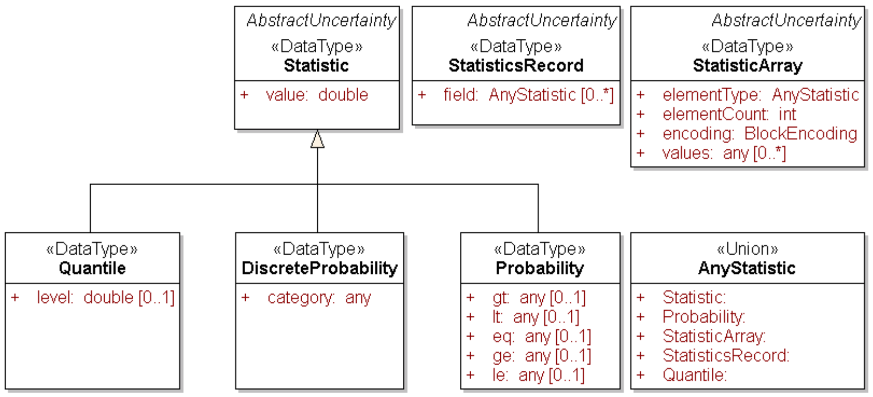

Fig. 3. UncertML model for summary statistics.

Throughout UncertML we follow a weak-typed design pattern that offers improved extensibility at the cost of strict validation. Weak-typing works by providing generic types with generic properties, in contrast to a strongly-typed design with concrete types and well-defined properties. Figure 3 introduces the hierarchy for summary statistics; the base type is a generic ‘Statistic’ that can be used for most summary statistics such as mean, median, variance or standard deviation. The value of a statistic is given through the ‘value’ property that holds a single real number. All types within UncertML extend the ‘AbstractUncertainty’ type and therefore inherit the ‘definition’ property. Accepting any Uniform Resource Identifier (URI) as a value, the ‘definition’ property provides a level of semantics to the weak-typed elements. Typically the URIs resolve to a dictionary entry describing the uncertainty type of interest, however, other methods of description may be used such as ontologies.

```
<Statistic definition="Mean">    <Mean>
  <value>34.5</value>              <value>34.5</value>
</Statistic>                     </Mean>
```

Fig. 4. Comparison of a weak-typed (left) and strong-typed (right) representation of a mean value. Weak-typing is more generic and provides greater extensibility, however, strong-typing provides easier validation

Certain summary statistics require additional information than the generic ‘Statistic’ type provides. A ‘Quantile’ is used for describing quantiles where a ‘level’ property, accepting a value between 0.0 and 1.0, defines the quantile of interest. Probabilities offered through either the ‘DiscreteProbability’ or ‘Probability’ types. The former provides a ‘category’ property which may contain any information, and the latter offers a range of properties including ‘equal to’, ‘greater than’ and ‘less than’; a combination of which may be used. It should be noted that probabilities differ from other summary statistics in that their ‘value’ property contains a *probability* (0.0–1.0) rather than an actual *value* (with units of measure etc).

```
<Statistic definition="Mean">
  <value>26.5</value>
</Statistic>

<Probability definition="Probability" gt="23.4" lt="33.4">
  <value>0.34</value>
</Probability>
```

Fig. 5. Two XML instances, the first represents a mean value while the latter shows the probability that a value falls between 23.4 and 33.4

It is often the case that one would wish to describe a collection of individual statistics to provide a summary of a particular variable. A ‘StatisticsRecord’ is used for this exact purpose and groups different statistics into a unified structure. When dealing with multiple instances of the same statistic it is more appropriate to use the 'StatisticArray' type. The flexibility of UncertML allows any combination of records and arrays to be created including arrays of records and records of arrays. All aggregate types within UncertML utilise the Sensor Web Enablement (SWE) common encoding schema [3] to provide an extensive list of options for encoding the data, including most MIME types.

```
<StatisticRecord>
  <field>
    <Statistic definition="Mean">
      <value>34.5</value>
    </Statistic>
  </field>
  <field>
    <Statistic definition="Standard_Deviation>
      <value>12.4</value>
    </Statistic>
  </field>
</StatisticRecord>
```

Fig. 6. A collection of individual statistics can be grouped into a 'StatisticRecord' to provide a summary of a variable. This example shows a mean and standard deviation

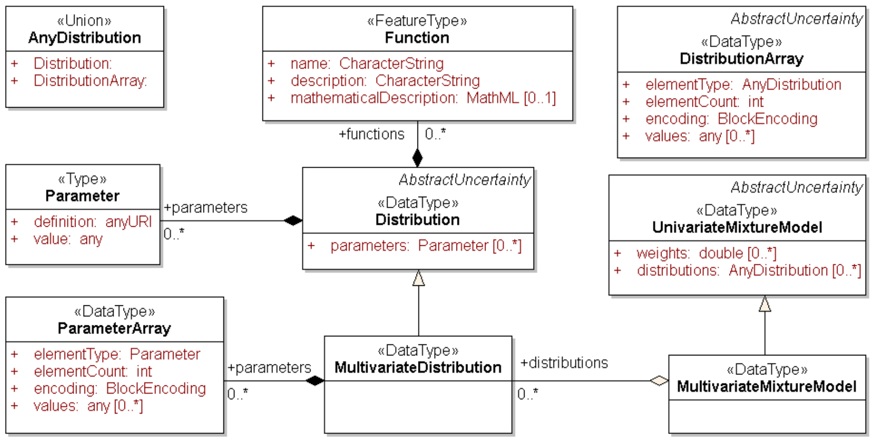

Fig. 7. UncertML model for distributions and other related types. The base 'Distribution' type is similar to the 'StatisticsRecord' discussed earlier, however, the addition of 'functions' provides a mechanism for describing a cumulative distribution function

A 'Distribution' type in UncertML follows a similar pattern to a 'StatisticsRecord' (Fig. 7)due to it containing a collection of parameters. An example encoding of a distribution is shown in Fig. 8 and consists of a reference to a dictionary through the 'definition' property as well the distribution parameters and their values. When a link to such a definition is not available a 'Distribution' can be extended to include a set of functions inline, encoded in MathML [4], such as a cumulative distribution function, probability density function or other arbitrary functions that may be performed on a distribution. Such flexibility allows users to work with distributions about which they have no prior knowledge.

```
<Distribution definition="Gaussian_Distribution">
  <parameters>
    <Parameter definition="Mean">
      <value>23.4</value>
    </Parameter>
    <Parameter definition="Variance">
      <value>56.7</value>
    </Parameter>
  </parameters>
</Distribution>
```

Fig. 8. A typical distribution encoded in UncertML. Reference to a dictionary entry is made through the 'definition' property which provides a complete description of a distribution, including its cumulative distribution function

There are many instances where a single distribution is not sufficient or where you wish to work with multivariate distributions, interpolation being one such example. The 'DistributionArray' type takes the form of an 'array of records' mentioned earlier and allows multiple instances of a particular distribution to be encoded efficiently using the SWE encoding schema. The 'MultivariateDistribution' type shown in Fig. 7 is an extension of the base 'Distribution' type, differentiated by the inclusion of a number of 'ParameterArray' properties. This is due to the nature of multivariate distributions having more than a single value for each parameter. UncertML provides two mixture model types that may be used for encoding a collection of distributions, each of which describe a variable by different amounts. Conceptually the 'UnivariateMixtureModel' is similar to the standard 'DistributionArry', however, an additional property yields an array of values between 0.0–1.0 to indicate the relative fraction, or weight, of each distribution, the total of which must sum to 1. A 'MultivariateMixtureModel' is a restriction on the univariate model that only allows a collection of multivariate distributions.

The final strand of UncertML is concerned with realisations, or samples, seen in Fig. 9. A single realisation is encoded using the 'Realisation' type which is identical to a 'Statistic', however, we feel it necessary to make a conceptual distinction between the two. Typically one would not wish to work with single realisations, instead preferring to encode large arrays; UncertML provides the ‘RealisationArray’ type as a solution. A ‘RealisationArray’ utilises the SWE encoding block to provide an efficient means of encoding vast quantities of data, a small example can be seen in Fig. 10.

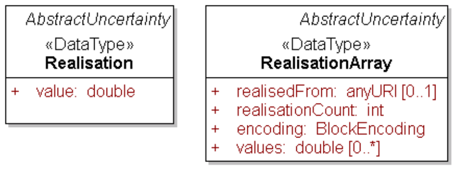

Fig. 9. A single realisation may be encoded using the ‘Realisation’ type, however, a typical user would wish to encode multiple realisations for which scenario a ‘RealisationArray’ is provided

```
<RealisationArray realisedFrom="Gaussian_Distribution">
  <realisationCount>5</realisationCount>
  <swe:encoding>
      <swe:TextBlock tokenSeparator="," tupleSeparator=" "
      decimalSeparator="." />
  </swe:encoding>
  <values>
    53.2,58.4,51.3,42.9,60.02
  </values>
</RealisationArray>
```

Fig. 10. Realisations (or samples) can be encoded using the ‘RealisationArray’ type. If the distribution from which these samples were realised is known then the ‘realisedFrom’ property may be used. A ‘tokenSeparator’ is used to identify individual values within a tuple and a ‘tupleSeparator’ is used to separate tuples

5 Integrating UncertML into the INTAMAP Project

The Observations & Measurements schema [6] provides an extensive model for describing the act of observing. Accompanying the ‘result’ property, this model may include properties for documenting the observation time (‘samplingTime’), the feature or location (‘featureOfInterest’), the property being measured (‘observedProperty’) and the procedure or instrument used to generate the result (‘procedure’). Typically the ‘procedure’ property will contain a sensor model encoded in SensorML [3] which can describe the error characteristics of a sensor (e.g. bias).

Within INTAMAP a request for interpolation is made by sending a collection of observations, encoded in the O&M schema, to the Web Processing Service interface. UncertML is used within the 'result' of an observation (Fig. 11) to describe the uncertainty inherent in observed values. Utilising both the error characteristics of a sensor and the observation uncertainty allows us to employ the arbitrary likelihood estimation techniques mentioned biefly in Sect. 3.

Due to UncertML types not encoding phenomena or geospatial attributes it is envisaged that a three layered architecture, seen in Fig. 12, will be employed, where each layer adds an extra level of detail. It should be stressed that this chain is not a part of UncertML, nor is it mandatory that UncertML be implemented in this way, it is simply an abstract notion of how one may wish to use UncertML when dealing with geographic data.

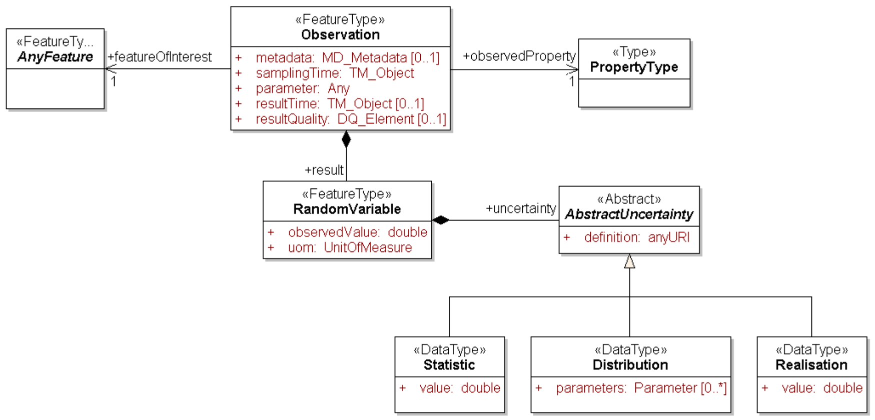

Fig. 11. An observation model within the O&M schema. The result can be of any type, in this instance it is a 'RandomVariable' which uses any uncertainty type from the UncertML schema to encode the value

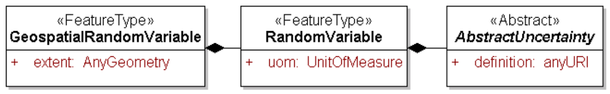

Fig. 12. Three layered implementation of UncertML. At it's simplest, UncertML only encodes the values of an uncertainty type. A 'RandomVariable' type adds a link to a phenomena and its units of measure and a 'GeospatialRandomVariable' adds further detial with an attached geometry. These random variable types are not included in UncertML and only represent one possible implementation

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 [13], of some variable defined by a series of Gaussian distributions encoded in UncertML. Figure 1 in Sect. 3 displays the lineage of an interpolation request in INTAMAP.

6 Conclusion

Embracing the ongoing evolution in software engineering to adopt a loosely coupled, interoperable, framework will make geostatistical methods available to a larger array of users. With the development of UncertML, as part of the INTAMAP project, a large step has been taken towards achieving this goal. The European Radiological Data Exchange Platform (EURDEP) provides a case study for the INTAMAP project and demonstrates a clear need for real-time interpolation across a Service Oriented Architecture.

However, for truly interoperable geostatistics, several areas require greater attention. A conceptual model for supporting the use of UncertML within geostatistical models will see the inclusion of variograms, covariance functions and other random functions. 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. This *may* be included as part of the OWS-6 request for quotation. A working interpolation service will be available for testing on-line 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. IEEE standard computer dictionary. a compilation of IEEE standard computer glossaries. *IEEE Std 610*, 18 Jan 1991.
2. Peter M. Atkinson. Geographical information science: geostatistics and uncertainty. *Progress in Physical Geography*, 23:134–142, 1999.
3. Mike Botts and Alexandre Robin. OpenGIS Sensor Model Language (SensorML) Implementation Specification. OpenGIS standard 07-000, Open Geospatial Consortium Inc, July 2007. http://www.opengeospatial.org/standards/sensorml.
4. David Carlisle, Robert Miner, Patrick Ion, and Nico Poppelier. Mathematical markup language (MathML) version 2.0 (second edition). W3C recommendation, W3C, October 2003. http://www.w3.org/TR/2003/REC-MathML2-20031021/.
5. Helen Couclelis. The Certainty of Uncertainty: GIS and the Limits of Geographic Knowledge. *Transactions in GIS*, 7(2):165–175, 2003.
6. Simon Cox. Observations and Measurements – Part 1 - Observation schema. OpenGIS standard 07-022r1, Open Geospatial Consortium Inc, December 2007. http://www.opengeospatial.org/standards/om.
7. Thomas Erl. *Service-Oriented Architecture: A Field Guide to Integrating XML and Web Services*. Prentice Hall PTR, Upper Saddle River, NJ, USA, 2004.
8. Thomas Erl. *Service-Oriented Architecture : Concepts, Technology, and Design*. Prentice Hall PTR, August 2005.
9. David C. Fallside and Priscilla Walmsley. XML schema part 0: Primer second edition. W3C recommendation, W3C, October 2004. http://www.w3.org/TR/2004/REC-xmlschema-0-20041028/.
10. Gerard B. M. Heuvelink and Michael F. Goodchild. *Error Propagation in Environmental Modelling with GIS*. CRC Press, 1998.
11. Ben Ingram, Dan Cornford, and Lehel Csató. Robust automatic mapping algorithms in a network monitoring scenario. *7th International Conference on Geostatistics for Environmental Applications (geoENV 2008)*, This volume, 2008.
12. Nicolai M. Josuttis. *SOA in Practice: The Art of Distributed System Design (Theory in Practice)*. O'Reilly Media, Inc., August 2007.
13. Clemens Portele. OpenGIS Geography Markup Language (GML) Encoding Standard. OpenGIS standard 07-036, Open Geospatial Consortium Inc, August 2007. http://www.opengeospatial.org/standards/gml.
14. Peter Schut. OpenGIS Web Processing Service 1.0.0. OpenGIS standard 05-007r7, Open Geospatial Consortium Inc, June 2007. http://www.opengeospatial.org/standards/wps.
15. François Yergeau, Eve Maler, Tim Bray, Jean Paoli, and C. M. Sperberg-McQueen. Extensible markup language (XML) 1.0 (fourth edition). W3C recommendation, W3C, August 2006. http://www.w3.org/TR/2006/REC-xml-20060816.