Open GIS Consortium Inc.

Date: 2002-04-19

Reference number of this OpenGIS© Project Document: **OGC 02-027**

Version: 0.86

Category: OpenGIS© OGC Interoperability Program Report-Engineering Specification

Editor: Simon Cox

Observations and Measurements

Document type:	OpenGIS© Discussion Paper
Document subtype:	Engineering Specification
Document stage:	Draft
Document language:	English

Table of Contents

i. Preface

ii. Submitting Organizations

This Interoperability Program Report, Engineering Specification, is being submitted to the OGC Interoperability Program by the following organizations:

CSIRO Australia

Galdos Systems Inc

iii. Document Contributor Contact Points

All questions regarding this submission should be directed to the editor or the submitters:

CONTACT	COMPANY	ADDRESS	PHONE/FAX	EMAIL
Simon Cox	CSIRO Australia	ARRC PO Box 1130 Bentley WA 6102 Australia	+61 (8) 6436 8639	Simon.Cox@csiro.au
Ron Lake	Galdos Systems Inc.	Suite 200, 1155 West Pender Street Vancouver, BC V6E 2P4 Canada	+1 (604) 484-2750	rlake@galdosinc.com

iv. Revision history

Date	Release	Author	Paragraph modified	Description
2001-12-20	0.3	SJDC		
2001-12-24	0.4	SJDC	Sections 6, 7 and Annex A	Big reorganisation – separated model from XML, more and better examples, simpleContent schemas included
2002-01-02	0.5	SJDC	Sections 6, 7 and Annex A, B	Small changes in terminology; some parameters promoted (optionally) to collection containers; fixed the "redefine" structure in Y3observation.xsd; explicit "recommendation" section
2002-01-04	0.6	SJDC, RL	Sections 6, 7 and Annex A, B	Measurement promoted to be used for the measurement-event feature. Observed value used for an abstract typed value. Definitions inserted in section 4.
2002-01-10	0.65	SJDC	Sections 6, 7 and Annex A, B	Measurand changed to observable. Members parameter added to measurementArray. ValueBase moved into gml namespace. Placeholders for Galdos ObservableType dictionary inserted
2002-01-17	0.7	SJDC		Reintroduce simpleContent encoding of scalar values. Various fixes to arrays and collections – some impacting on content-model sequences. Make most element declarations global, attribute declarations local. Change namespace "swe" to "ows".
2002-01-30	0.76	SJDC		Introduce _ValueList form for homogeneous arrays.
2002-01-31	0.76.1	SJDC	Section 7, Annex B	Corrected schemaLocation values
2002-04-04	0.8-0.85	SJDC		Replaced XML with "simpleContent" version Conforms with draft GML3 Emphasised weak typing approach Added more and more diverse instance examples to Annex B Added issues and future work section Extensive rewriting
2002-04-19	02-027 (.86)	HAN	Various	Final OWS-1 review and edit; minor changes. Produced OGC 02-027.

v. Changes to the OpenGIS® Abstract Specification

The OpenGIS© Abstract Specification requires changes to accommodate the technical contents of this document.

The following is a list of the required changes: TBD

Foreword

Attention is drawn to the possibility that some of the elements of this part of OGC 02-027 may be the subject of patent rights. The Open GIS Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

This specification was developed under the OWS 1.1 initiative.

Introduction

We describe a framework and encoding for measurements and observations. This is required specifically for the Sensor Collection Service and related components of an OGC Sensor Web capability, and also for general support for OGC compliant systems dealing in technical measurements in science and engineering.

The aim is to define a number of terms used for measurements, and the relationships between them. This proposal discusses **Observation, Measurement, Observed Value, Coverage, SensorInstance, ObservableType,** and related terms, presented using UML class diagrams and in equivalent GML conformant XML serialisations. The scope is restricted to measurements whose results are expressed as quantities, categories, and boolean values.

We discuss how the observation and measurement model is used in the context of SensorWeb is included. The discussion notes how different parts of the information model would be provided by different services. However, the details remain to be resolved and might depend on specific use-cases.

.

Observations and Measurements Engineering Specification

1 Scope

We describe a framework and encoding for measurements and observations.

Within the context of the OGC Feature Model we define a *measurement* to be an observation feature whose value property is a value of some natural phenomenon. A measurement usually refers to the measuring device and procedure used to determine the value, such as a sensor or observer, analytical procedures, simulations or other numerical processes. Measurement binds the observed value to the (spatiotemporal) location where it was made.

A measurement results in an estimate of the value of a property. Various classes of *observed values* are required, including nominal category, ordered category, quantity, count and position, and also *value collections*, including tuples, value arrays and lists, and composite values including vectors and tensors. An observed value may be semantically typed according to the subject or observable, which is known as measurand in the case of measured values. The value normally requires a reference system to provide the context for its interpretation, such as the unit of measure for a quantity, or a dictionary or "code space" for a category.

We discuss how the observation and measurement model is used in the context of SensorWeb, noting that different parts of the information model may be provided by different services, though details remain to be resolved and might depend on specific use-cases.

Not covered in this report are:

- Sensor Model Language (SensorML) & Sensor Instance/Sensor Type registries
- Reference System definitions (CRS, frames, units of measure, dictionaries & category lists).
- Measurements whose result is geometric or temporal information.

2 Conformance

Conformance and Interoperability Testing for this OGC Interoperability Program Report may be checked using all the relevant tests specified in Annex A (normative). The framework, concepts, and methodology for testing, and the criteria to be achieved to claim conformance are specified in ISO 19105: Geographic information — Conformance and Testing.

3 Normative References

The following normative documents contain provisions which, through reference in this text, constitute provisions of this part of OGC 02-027. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of OGC 02-027 are encouraged to investigate the possibility of applying the most recent editions of the normative documents indicated below. For undated references, the latest edition of the normative document referred to applies.

ISO 191XX (all parts), *Geographic Information*

Abstract Specification Topic 0: Overview, OGC document 99-100r1

Guidelines for Successful OGC Interface Specifications, OGC document 00-014r1

Sensor Model Language IPR, OGC 02-026.

Sensor Collection Service IPR, OGC 02-028.

4 Terms and Definitions

The following terms and definitions apply.

4.1
Observed Value
A value describing a natural phenomenon, which may use one of a variety of scales including nominal, ordinal, ratio and interval. The term is used regardless of whether the value is due to an instrumental observation, a theoretical model, subjective assignment or some other method of estimation or assignment.

4.2
Observable
The particular phenomenon to which values are assigned, such as temperature, gravity, chemical concentration, orientation, number-of-individuals. The equivalent term is **measurand** when the values are determined by measurement .

4.3
Measurement (feature)
An instance of a procedure to estimate of the value of a natural phenomenon, typically involving an instrument or sensor [VIM]. This is implemented as a dynamic feature type, which has a property containing the result of the measurement. The measurement feature also has a location, time, and reference to the method used to determine the value. A measurement feature effectively binds a value to a location and to a method or instrument.

5 Conventions

5.1 Symbols (and abbreviated terms)

The following symbols and abbreviated terms are used in this document.

API	Application Program Interface
COTS	Commercial Off The Shelf
DCE	Distributed Computing Environment
GML	OGC Geography Markup Language
ISO	International Organization for Standardization
OGC	Open GIS Consortium
PSVI	Post schema-validated information set (for XML documents)
UML	Unified Modeling Language
XML	eXtended Markup Language
XSD	W3C XML Schema Definition Language
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional

5.2 UML Notation

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

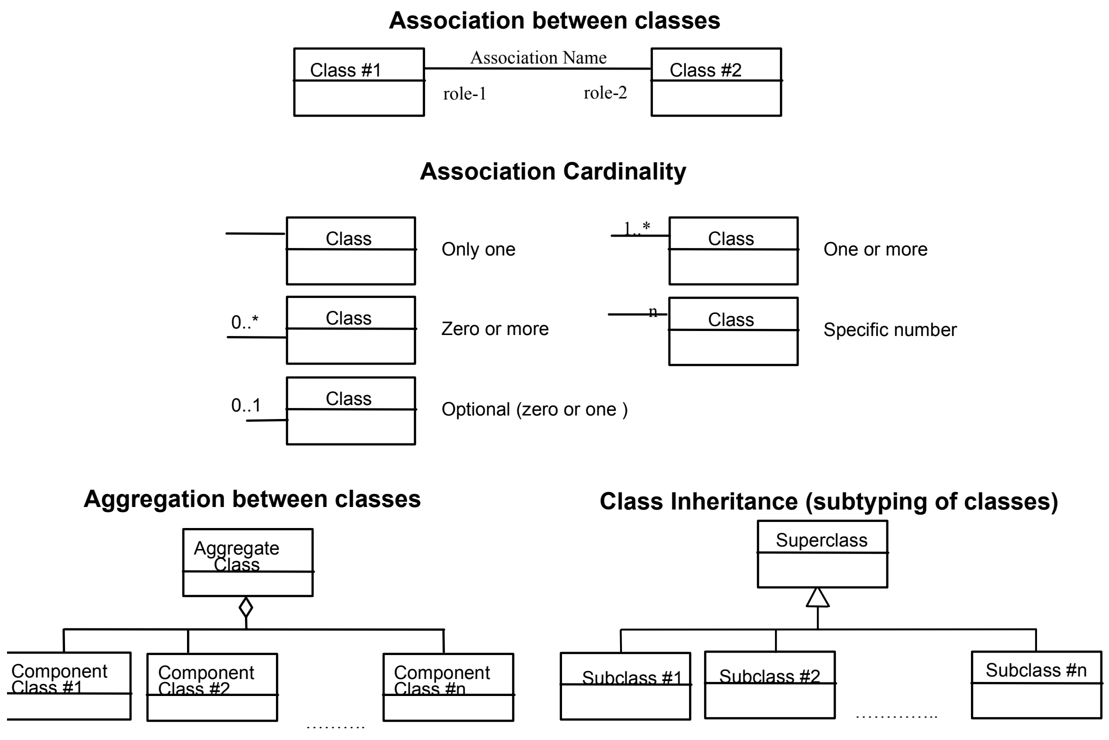

Figure 1. UML notation

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

a) <<Interface>> A definition of a set of operations that is supported by objects having this interface. An Interface class cannot contain any attributes.

b) <<DataType>> A descriptor of a set of values 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.

c) <<CodeList>> is a flexible enumeration that uses string values for expressing a list of potential values.

In this document, the following standard data types are used:

a) CharacterString – A sequence of characters

b) Integer – An integer number

c) Double – A double precision floating point number

d) Float – A single precision floating point number

6 Observations and Measurements Model

In this section we describe a model for measurements and associated components. The analysis is presented as an information model using UML static class diagrams.

We start in section 6.1 with a summary of the "Feature" which is the basic item in the OGC model for geographic information. This is followed in section 6.2 with a very brief discussion of an alternative view of geographic information based on the Coverage, but focussing particularly on the shared information content between these views. Understanding this equivalence allows us to base the subsequent analysis on the Feature model with a clear understanding of how to transform the same information into a Coverage.

Descriptions follow of models for Measurements (section 6.3), for Observed Values (section 6.4), and for collections of both Observed Values and Measurements, and the dependencies between these (section 6.5). Collections are critical in transforming between the Feature and Coverage views.

Finally, in section 6.6 we review the complete model, and introduce some associated components: sensors and observable types. This leads to a discussion of some operational factors of a Sensor Collection Service, particularly focussing on the supply of different information items by different services through various interfaces.

6.1 The Feature Model

The Feature model underlies the ISO/OGC view of geographic information [OGCAS5]. The term `Feature` (Figure 2) is used for an object located in geographic space, and also for its model representation in software. A Feature will have a number of `properties`, some of which may be geometric and spatial. The latter is important: in contrast to traditional GIS approaches, a Feature is *not* defined primarily as a geometric object, but rather as a conceptually meaningful object, one or more of whose properties may be geometric.

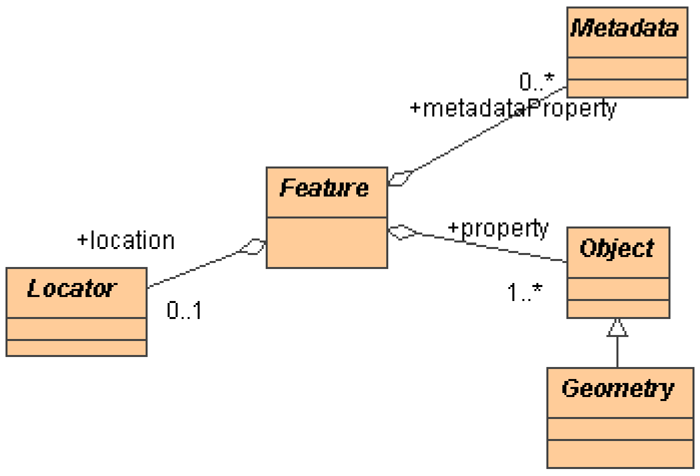

Figure 2. The Generic Feature Model

A specific Feature Type is defined primarily in terms of the set of named properties associated with that type [ISO19110]. A Feature instance is thus the map of its property values.

6.2 Relation to Coverages

Coverage is an alternative model for spatial data, and is useful in some cases. A Coverage is "… a set of attribute values (its *range*) associated to position within a bounded space (its spatiotemporal *domain*) …" [ISO19123]. The range is homogeneous in type. A coverage is normally considered to be a map of the range onto the domain.

The information associated with a collection of Features *of uniform type* may be represented as a discrete spatial Coverage, where the set of *location* properties provide the domain of the coverage, and the set of the *other* properties provide the range. A Coverage as a whole can thus be considered to have a "type" related to the type of the member Features of the equivalent Feature Collection: "Feature type" is essentially defined by the set of properties, which thus determines the structure of the information in the range of the equivalent coverage.

Note that this type is distinct from typing a coverage according to the representation of its spatiotemporal domain, such as grid, TIN, etc.

This equivalence is discussed further in Section 6.5.2 and illustrated in Figure 9. For the purposes of the discussion in the document, we assume a method for transforming any homogeneous Feature Collection into a discrete Coverage is available, and vice versa.

6.3 Measurement Model

We define a **`Measurement`** as a specific Feature type (Figure 3), in which one of its properties is **`resultOf`**, the value of which is an **`Observed Value`** describing some natural phenomenon.

Measurement has a **`measuredBy`** property, the value of which is a particular instrument, sensor, observer, or procedure. A proposed model and encoding for the description of sensors is presented in another IPR arising from work done in this project [SensorML]. A measurement is **`measuredAt`** a particular location and has a **`timeStamp`**, so the Measurement feature is effectively a "measurement event" or "measurement incident". A measurement will frequently be related to other features, such as a measurement collection, a survey tie-point, a previous measurement from the same location, a previous measurement which is the basis for a new measurement (e.g. a *classification* may be based on an earlier set of quantitative measurements), etc. We implement this through the Measurement feature having a general association with other Features using a **`relatedFeature`** association class, with a **`role`** attribute, allowing the association to be (weakly) typed.

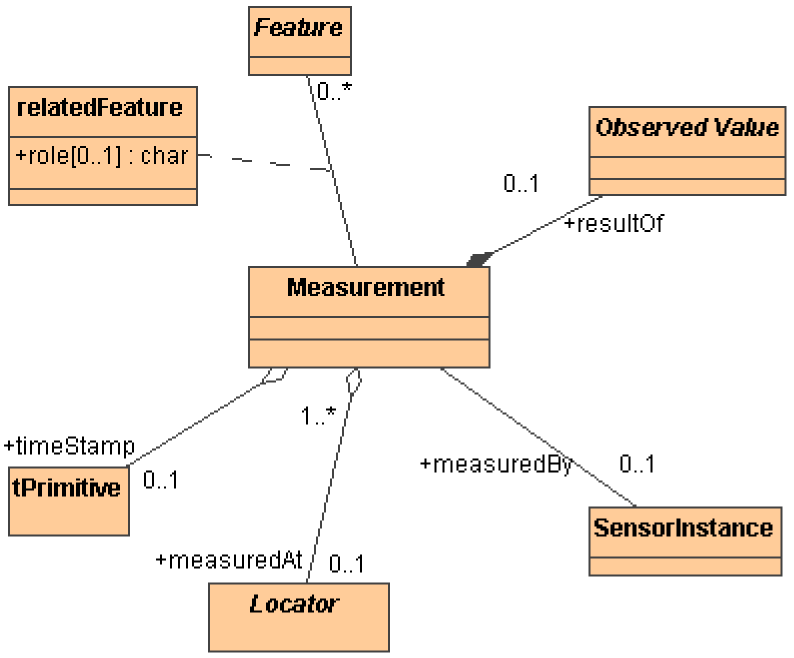

Figure 3. The Measurement Feature

In the context of GML, we inherit several of the properties from the `Observation` feature (Figure 4). Measurement specialises Observation by adding the `measuredBy` property, and changing the `valueOf` property name to `resultOf`, and `location` to `measuredAt`. Note that the value of the location might be another feature, such as a measurement "station" or a "specimen".

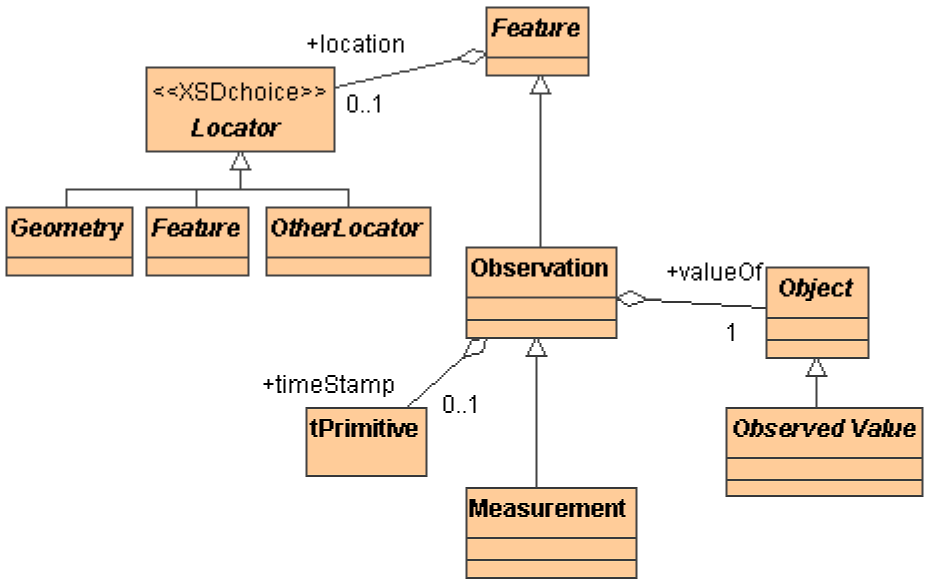

Figure 4. Relationship with the GML Observation Feature

It is through its association with a Measurement feature that an Observed Value is bound to a geospatial location, to a time instant or period, and to the `SensorInstance` responsible for the measurement. The Measurement might be considered to be the *map* of an Observed Value and (spatiotemporal) location.

6.4 Value Model

In engineering and natural sciences, "value" refers to the magnitude of a quantity expressed as a unit of measurement multiplied by a number [VIM]. In this document we generalise the concept to also include values using other scales, such as terms selected from a controlled list, which may be ordered, and positions within a frame. We refer to these in the singular as `Observed Value`, though in the model this is an abstract type which is never instantiated.

The equivalent concept for spatial values is the spatial reference system, including coordinate reference systems. A temporal value relates to a temporal reference system, such as a calendar.

6.4.1 Scalar Values

Simple or scalar values can occur on several scales, including nominal, ordinal, interval and ratio scales [SUP1963]. We model these as a small number of generic DataTypes `Category`, `OrderedCategory`, `Quantity`, `Position`, `Count` and `Boolean`, derived from an abstract superclass `ScalarValue` (Figure 5).

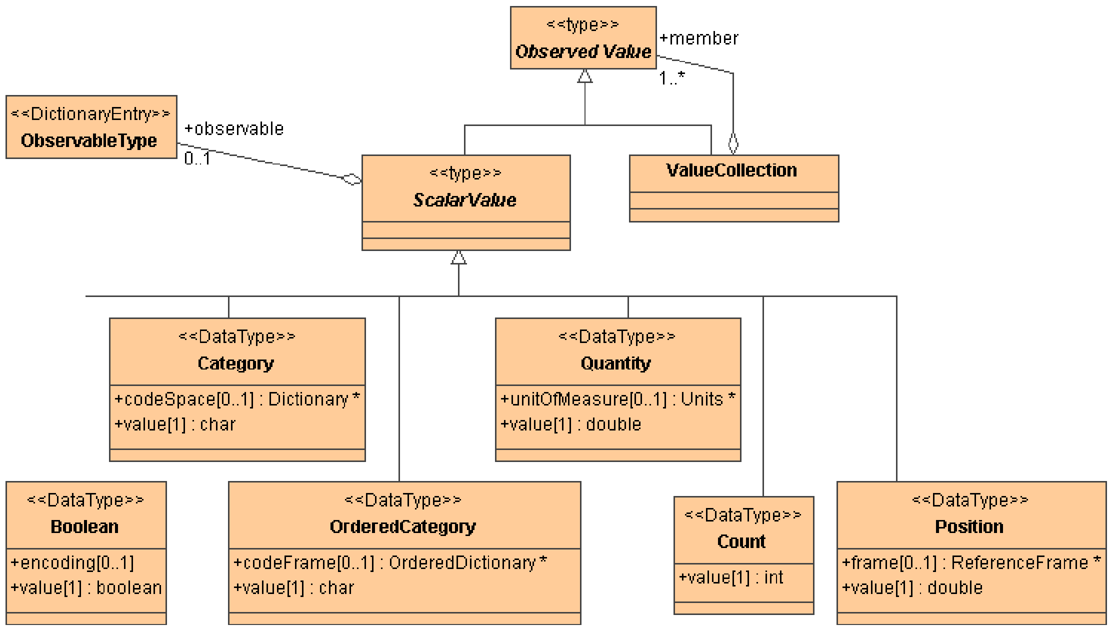

Figure 5. Generic Observed Value Model

A similar pattern is used for each of the scalar value types, comprising a numerator or value using the appropriate base data type, and a denominator defining the context or base units for the value, implemented as a pointer to the reference system. An optional `observable` parameter provides a slot to indicate some semantics of the nature of the phenomenon being observed.

The model for **`Quantity`** is the same as that discussed in the OGC project document on Units of Measure [UOM2001] except that here we add **`observable`**.

In a related analysis based on his work in the medical domain, Fowler uses the term *Observation* instead of Observed Value, but only includes **`Category`** and **`Quantity`** [FOW1998].

Issue Name: Need for Position and OrderedCategory types (SJDC, 2002-04-12)]

Issue Description: The need for distinct datatypes for OrderedCategory and Position has been reconsidered. The fact that a number is on an interval vs ration scale, or that a term is on an ordinal vs nominal scale is important in determining the valid operations. However, this information will be apparent from the definition of the reference system. Furthermore, it is necessary to inspect the definition of the reference system in order to recover other information required for performing operations on the value, such as scale. Thus, only one piece of information is indicated by changing two tokens in the model – i.e. the name of the data type and the reference system label – and this information is incomplete for the use case anyway.

Resolution: OrderedCategory and Position will be removed from the model in future. (SJDC, 2002-04-12)]

6.4.2 Typing the Observed Values

In the model shown in Figure 5, Observed Values are **strongly typed** by their **content** model – i.e. whether it is a number or text, and whether the scale is ordered or has an arbitrary origin. But we follow Fowler [FOW1998] in that the Observed Values are **weakly typed** in their **semantics**, using the parameter **`observable`**, which takes a value drawn from an Observable Type dictionary or registry.

Parameterised typing allows a new Observed Value type to be created by an 'Authority' by adding an entry to the Observable Type dictionary. A new type may even be made when a measurement is instantiated, provided the Observable Type is text-valued and can be defined locally. In the latter case, however, the new type is unlikely to be comprehensible by reading applications and is therefore not interoperable.

There are limitations associated with weak typing. In particular, it is necessary to ensure consistency between the ObservableType and the Observed Value class, so that (for example) a "Quantity" is not given the observable "speciesName", which would be an error. In an implementation based on Fowler's model, Yoder et al. [YOD] proposed that an ObservationType class handle the necessary validation.

Alternatively, strongly typed Observed Values can be derived by sub-classing. The value of observable is given a fixed value (which may become the name of the data type), so the observable type is bound to a content model, ensuring at least this level of validity. Two levels of derivation are illustrated in Figure 6.

i. moderately typed Values might have additional parameters for further sub-classification. In the classes above the separator in Figure 6 the attributes `in` and `of` are examples of additional axes for sub-typing, which take a value from the appropriate domain;

ii. strongly typed Values may be defined for every possible combination of type parameter values. In this case all the additional axes for sub-classification are implicitly bound into the observable type. This maintains the clarity of the basic model introduced in Figure 5, which has a single observable parameter. But it leads to a combinatorial explosion of typed Values, especially when dealing with classes with large taxonomies, such as chemical and biological species.

So even if strong-typing of the Observed Value is desired, there is likely to still be a role for parameterised sub-types.

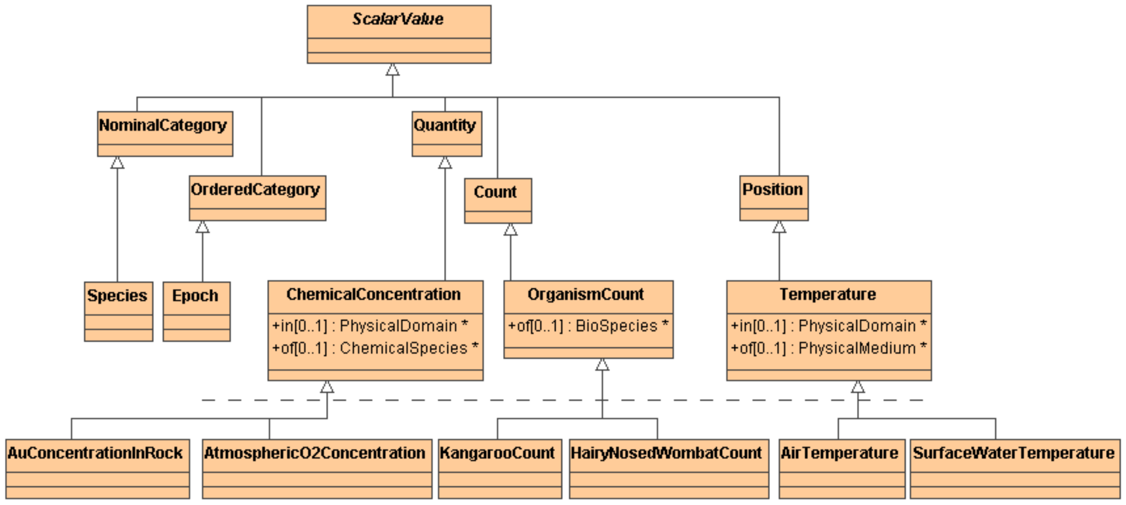

Figure 6. Specific value classes

6.4.3 Typing the Measurement

In some cases a short description of an observable (e.g. using entries from a dictionary) will be enough to capture the information necessary to ensure that values are comparable. For example, a term like “wind-speed” is usually unambiguous.

However, in other cases some of the necessary details of the observable are in the sensor description. For example, in remote sensing the band of a radiance measurement is best modelled using sensitivity information from the sensor. And in analysis of trace-elements in a rock specimen, a description of the sample preparation method is often required to interpret the result.

This flexibility is accommodated in the model by having two slots where the observable semantics are attached. Note that on `Observed Value` the `observable` parameter is optional. Information covering similar ground is also available through the `measuredBy` property of the `Measurement` with which this `Observed Value` is associated (see section 6.3 above).

The value of `measuredBy` is normally the description of a `SensorInstance`. The sensor description includes reference to an observable type, to instrumental parameters such as sensitivity ranges, etc. and can also include information concerning the processing of the data used to generate meaningful values. The sensor description thus provides slots for an unlimited set of classifiers required to fully parameterise the observable.

It may also be useful for `measuredBy` to identify other types of object. For example, a field survey may be carried out by a *person*. An assay of a rock sample is usually carried out by a commercial *laboratory,* using one of several *procedures* pre-defined according to standards applicable at the time the measurement is made. Estimates of values of phenomena may be made by *software* such as a numerical simulation system or differential equation solvers, using finite elements, finite differences, etc. To cover each of these cases it will probably be necessary to define additional feature types.

6.5 Collections

6.5.1 Value Collections

Collections of Observed Values are frequently required, for example a record [ISO19103, ISO19107] or tuple. Following the "composite" pattern we treat `ValueCollection` itself as a Value, to get a pattern for building complex values or tuples recursively from other Observed Values (Figure 7). A complex value built this way has a tree structure, which ultimately resolve to leaf nodes at possibly varying levels each bearing a `ScalarValue`.

We define two special cases of ValueCollection.

A `CompositeValue` is used for a vector result of measurements made by a *SensorPackage* at one discrete time and place. The members are `components`*,* each of which has an optional distinguishing `axis`. The `observable` parameter on the container should be used for the collective name for the complex value, e.g. "AirQuality" or "Assay", which complements the `observable` parameter on each of the component values.

In a `ValueArray` each member is of the same type. A ValueArray may be used to record values produced by a single sensor may produce in a time-series, or by an array of similar sensors at a number of locations. Thus the information in a ValueArray may form the range set of a discrete Coverage. The type of the members is indicated by the `observable` parameter. The optional attribute `referenceSystem` is abstract: when the members of the array are `ScalarValue`s this is replaced by `codeSpace`, `codeFrame`, `unitOfMeasure` or `frame`, as appropriate. On a `ValueArray` the values of both `observable` and `referenceSystem` on the container are *inherited* by the member values.

Apart from being homogeneous in type, there are no other restrictions on the membership of a Value Array. Arrays of Composite Values may form a Value Array, such as a series of wind-velocity values. Arrays themselves may be nested, so a set of values corresponding to time-series of the same Observed Value from different locations can form an array of Value Arrays.

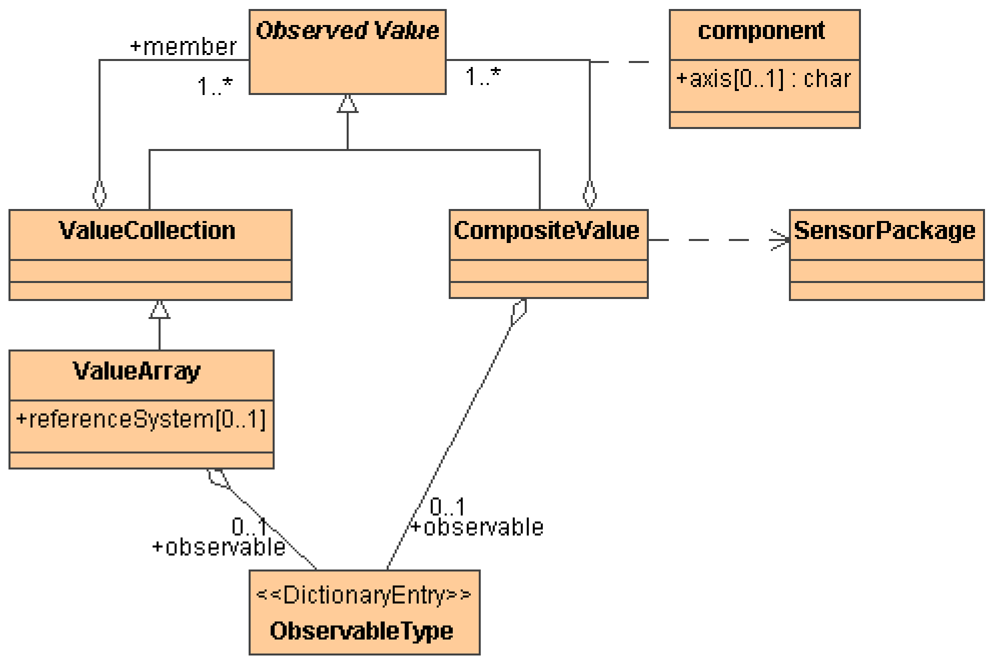

Figure 7. Model for Various Complex Value Types

For arrays of Scalar Values we define an additional structure, the `ValueList`, which has multiple values for the same reference system and observable (Figure 8). This is information-equivalent to a `ValueArray` with Scalar Values as the content, but leads to a more compact encoding.

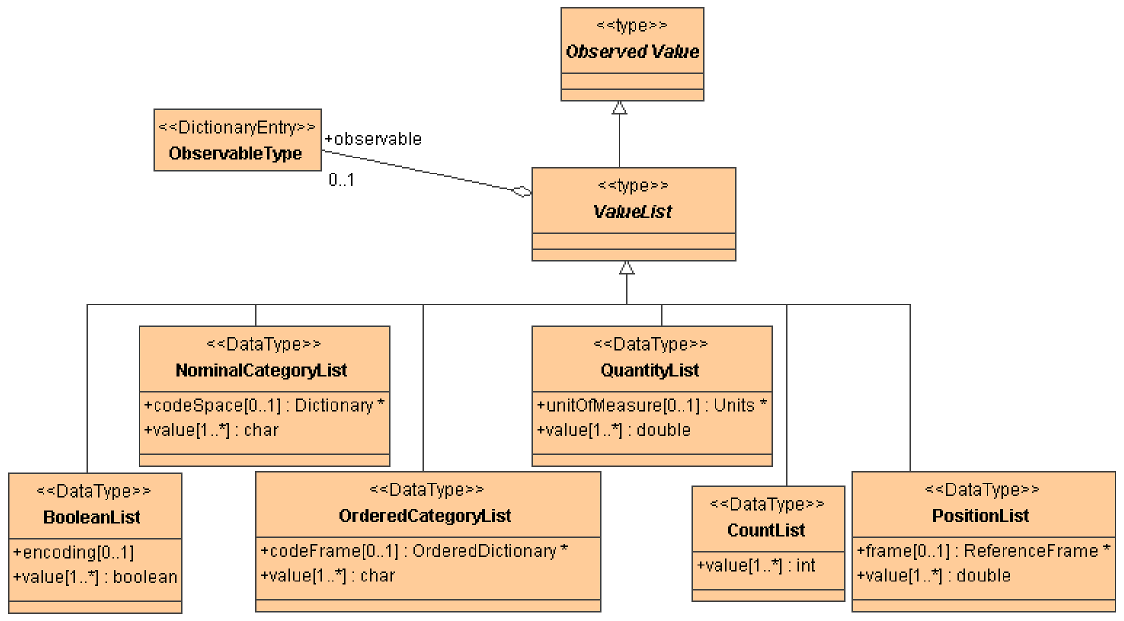

Figure 8. Model for Value Lists – Arrays of Scalar Values

6.5.2 Measurement Collections

Measurements may be aggregated into a Collection (Figure 9). Using the *composite* pattern again, a **`MeasurementCollection`** may itself be treated as a Measurement.

A MeasurementCollection consisting of homogeneous Measurements is a **`MeasurementArray`**. A parameter **`members`** provides weak typing of the MeasurementArray. Its value is a name corresponding to the type of the member Measurements. This provides a basic validation check..

A **`MeasurementArray`** can also be realised using a discrete measurement coverage. In a coverage model, the values in the range are mapped to their spatiotemporal domain collectively, using a rule or formula, rather than individually by being properties of a Feature within the domain. A single sensor can be associated with Measurements in a temporal coverage or **`MeasurementSeriesCoverage`**. A spatial array of similar sensors sampled at a single instant or measuring a time-invariant phenomenon can produce a spatial coverage or **`SpatialMeasurementCoverage`**.

A **`Measurement`** binds a single **`Observed Value`** to a single sensor. However, a **`SensorInstance`** can be a **`SensorCollection,`** constructed in a similar way to the other collections. The nature of the sensor or sensor collection determines the structure of the value, as well as its type. Thus, a **`SensorPackage`** can yield a measurement whose result is a **`CompositeValue`**, while a **`SensorArray`** can yield a **`ValueArray`**.

Note that the **`MeasurementArray`** and **`DiscreteMeasurementCoverage`** are dual views of the same information with completely equivalent content. They are useful in different contexts, which often map onto different phases of the information acquisition/exploitation process. For example, information is often collected in single events at discrete spatio-temporal locations. Thus a single **`Measurement`** is the natural view of the information at this phase, a number of which then build a **`MeasurementArray`**. However, at analysis phase, it is common to use many values of a single theme or observable, and perform operations based on their mutual relationships. This requirement is usually better addressed by presenting the information in an array, with the other values and spatio-temporal locations each available as separate arrays. This is the **`Coverage`** view.

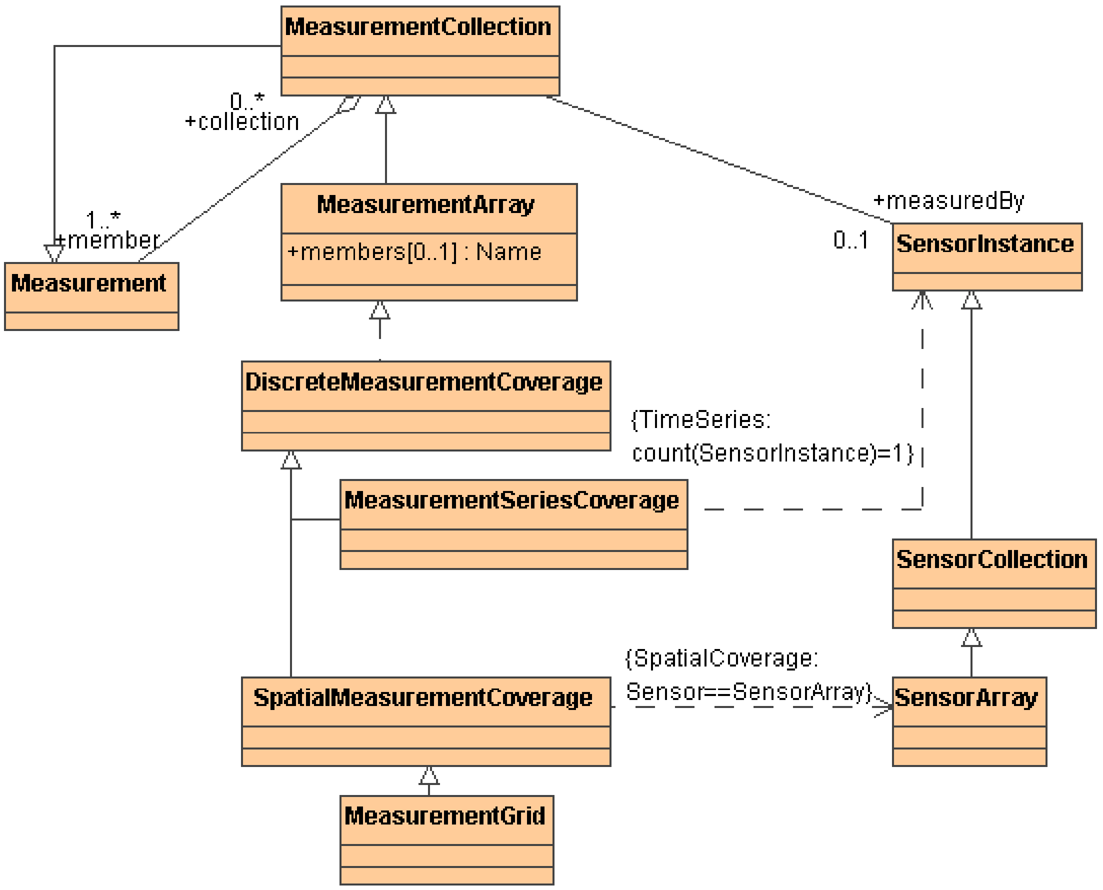

Figure 9. Measurement Collection and Measurement Coverage

6.6 Sensor Web and the Information Model

In order to build a system using the information model shown in Figure 3, we must consider the full set of dependencies and relationships. In Figure 10 we include the ObservableType service, which is used by both Observed Value and SensorInstance, and also note the association between SensorInstance and Locator. An individual sensor service would not normally provide all the information shown here at one time. Rather, various services and operations within Sensor Web will refer to various components of the model.

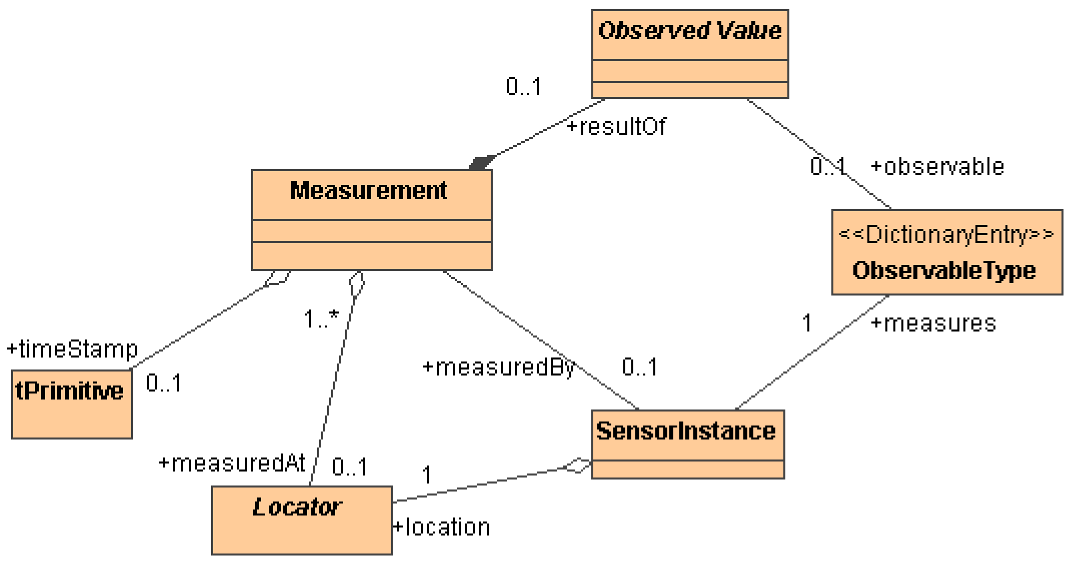

Figure 10. Measurement Model Including Additional Validation Dependencies

6.6.1 Measurements from a Sensor Collection Service

Several interfaces are expected to be available onto a Sensor Collection Service [SCS]. Observed Values may be encapsulated within Coverages or Measurement Features, which are available through the GetCoverage and GetFeature interfaces (respectively). Observed Values may also be available directly through the GetObservation interface, in which case the meaning of the measurements must be known from the context.

Examples:

- A SCS may supply *Observed Values* or a *ValueArray* from a single sensor (or several sensors) to a client directly, in response to a *GetObservation* request. The client would also need knowledge of the related sensor and location information: either directly through other interfaces onto the sensor service, or indirectly from a registry. The client must specify the timing of the measurement.

- A SCS may package information from the description of a single sensor (location, timing, sensor description) together with an *Observed Value* into a *Measurement* in response to a *GetFeature* request.

- A SCS may provide multiple measurements from a single sensor in a time-series: either as a *Coverage* in response to a *GetCoverage* request, or as a homogeneous *Feature collection* (*MeasurementArray*) in response to a *GetFeature* request (N.B. Feature Collections are Features).

- A SCS might aggregate *Observed Values* from (a collection of) individual sensors, and add the contextual information, such as location, and then present these to the client as a *Feature Collection*, a *Coverage*, or as a *ValueArray*, in response to a *GetFeature*, *GetCoverage*, or *GetObservation* request, respectively.

Thus, data provided by a SCS might be in the form of instance documents describing Measurements (i.e. including timing and location, plus other information) or might include only Observed Values (i.e. no explicit timing or location in the data document).

For example, in the case of fixed in-situ sensors, the location may be known in advance, as part of the SensorInstance description, and does not change. In the case of dynamic sensors, locational information might be computed, as a function of time, using parameters provided in the SensorInstance description.

Furthermore, when timing is specified in the request it is not needed explicitly in the data document, and an Observed Value may be returned. A single Observed Value or a ValueArray might be presented, representing the one or many time instants specified in the request.

But if, for example, the request is for any measurements that are available within a certain time interval, then the data returned will need to contain the specific timing information as well, and thus be encoded as Measurements.

6.6.2 Observable Type

An *Observed Value* has a *observable* property, whose value is an *Observable Type*, usually taken from a controlled source. The *SensorInstance* associated with the same *Measurement* has a *measures* property, which should have the same value of *ObservableType.* This is a basic validity check that can be made on any Measurement by a client application.

The Observable Type definition will normally include consideration of semantics, and dimensionality (e.g. expressed in the conventional $L^aM^bT^c$, form) of the phenomenon on which the measurement is being made. In some cases, some details of the observable will be recorded as part of the sensor description: for example, band information for radiation measurements are most appropriately modelled as part of the instrument sensitivity parameters. The meaning of the measured value may require explicit reference to the sensor description as well as the observable type definition.

The GML schema `ObservablesDictionary.xsd` proposed by Galdos Inc. provides a partial solution. It defines a simple structure for an observables dictionary. Fellah has discussed the limitation of using a syntactic approach, and has suggested that a more formally semantic or ontological language, such as RDF or DAML, will be required for at least part of the definition. These are complex questions and the details will not be resolved here. We will simply assume that an appropriate definition is available, and may be referenced using a URI.

6.6.3 Reference Systems for Values

Various Reference Systems are required to describe the taxonomy, scale, or frame for an ObservedValue. We call the controlled list (possibly hierarchical) of terms for category measurements (e.g. a list of biological species) a “code space”. If these are ordered in some way (e.g. a ranking of excellent, good, fair, poor; or geological eras which are understood to have possibly unquantified time-relationships with each other) it is a “code frame”. A scale used as the multiplier for a numerical measurement (e.g. kilograms, metres, litres) is a “unitOfMeasure”. A scale with a specified origin within some other conventionally defined system (e.g. Celsius; bearing in degrees relative to magnetic north; coordinate reference system; various historical time systems) is a “frame”.

See also the note Need for Position and OrderedCategory Types in Section 6.4.1.

The GML3 schema **`units.xsd`** develops components to define a particular units-of-measuure for Quantities. It concerns conversion of the desired units of measure to a "standard" units system (see also Figure 6 in YOD).

The GML3 schema **`temporal.xsd`** develops components to define and describe temporal reference systems.

The GML3 schemas covering coordinate reference systems and transformations are a comprehensive solution to the related area of spatial (primarily geodetic) reference systems.

However, there are currently no solutions proposed for the other classes of "reference systems" required for generalised observed-values. These are:

- *dictionary* ("code space"), which may need to be cross-referenced through a thesaurus. Note that a *gazetteer* is a specialised dictionary which binds a code to a location.
- *ordered dictionary* ("code frame") which is very complex when you consider all the ways in which things can be ordered (relative, numerical, fuzzy/partial) and also the relationships between code-frames which purport to cover the same space.

The Reference System must be commensurate with the Observable Type: e.g. Temperature should use Celsius, Fahrenheit or Kelvin frames; Speed should use Metres per second or Miles per Hour, etc. This is managed within the definition of the units-of-measure through the **`measurementType`** parameter. Providing the dictionary used by the units directory is the same as the dictionary of observable types used by the measurement then validation will be supported. However, if the measurement is using a dictionary of strong types, then it will have to support some kind of (for example) hierarchical taxonomy so that (for example) a value with an **`observable`** of “WaterTemperature” is valid using units of measure whose **`measurementType`** is simply “Temperature”.

It is a necessary condition that the “units-family” or dimensionality of the units of measure or for the scale component of a frame should match the dimensionality of the Observable Type. However, this is not a sufficient condition, particularly when considering dimensionless quantities such as *concentration*.

Note that we do not expect that this condition can or will be checked through XML Schema validation, so it is the responsibility of application processes to ensure validity using other methods.

Beyond these general model implications, the definition or syntax used for Reference Systems is outside scope here. In the rest of the report we merely assume that an appropriate definition is available, and may be referenced using a URI.

6.6.4 Sensor Instances and Other Descriptions of Measurement Methodology

Information about the Sensor Instance, encoded using SensorML, would usually be available from a SensorInstance registry, which would acquire this data when a related Sensor Web service is *published*. The client would probably access this information prior to acquiring observations or measurements from a specific Sensor Web service.

The SensorInstance description might also be available *directly* in response to a DescribeSensor request. The description will include information about the sensor calibration and accuracy, the sensor frame, mount and platform. Thus, the description of the sensor should allow an estimate of the error on the measurement to be made. The definition of SensorML, the model and language used to describe sensors, is the subject of another IPR [SensorML], so is not described further here.

The SensorInstance description might also include a description of the structure of measurements and observations that can be delivered by the sensor. This might include the schema according to which data is available in reponse to a request. Or this schema might be delivered in response to a DescribeObservation request.

It is the latter – i.e. schemas specifying the structure of Observed Values and Measurements delivered by the sensor – that is the primary subject of the remainder of this document.

7 XML Encoding of the Observation and Measurement Models

In this section we describe the implementation of the model for Measurements and Observed Values in XML. The analysis is presented through a set of extracts from XML Schema documents, with example instances for each case. The *measurement* schemas **`<include>`** various *value* schemas, so the latter are described first. The examples focus particularly on Measurement Collections and Value Collections, which will be important in building Coverages.

We start in section 7.1 with a brief description of the XML idiom used to serialise the models presented in section 6. Section 7.2 and 7.3 walk through the application of the generic Observed Value, and Measurement schemas, respectively. In Section 7.4 we briefly describe a method for deriving strongly typed values, based on fixing the observable and other parameters. The *instance documents* shown in these sections are all examples of how data streams from sensor services and encoded as XML documents might appear. Finally, in section 7.5 we discuss how the various instance and schema documents might be used in a Sensor Web deployment.

7.1 GML Mapping of Model Components into XML

OGC's Geography Markup Language [GML] defines a consistent XML serialisation for information modelled in UML. The style used results in very explicit XML instance documents: UML attribute names and association rolenames are used for the names of XML elements representing properties, and UML class names are used for the names of XML elements representing objects which may have identity.

Thus, a basic Feature modeled according to Figure 2 appears:

Code sample (i)

```
<Feature xmlns="http://www.opengis.net/gml" … >
  <description>A basic located feature for illustration</description>
  <name>Fred</name>
  <metadataProperty><Metadata>Some structured metadata </Metadata></metadataProperty>
  <location xlink:href="http://your.domain.org/locations#point3"/>
  <simpleProperty>Some property value</simpleProperty>
</Feature>
```

`description`, **`name`**, **`metadataProperty`**, **`location`**, and **`simpleProperty`** are all properties of this feature. In this example, three properties have a value that is encoded as a text-string, **`metadataProperty`** contains an element which would contain some structured metadata, and **`location`** uses **`href`** to point to a Point description available elsewhere.

The encoding used in this recommendation generally follows the GML idiom, so the default mapping is that *properties* are encoded as *elements* in content models. Thus "measuredAt", "measuredBy" and "resultOf", "amount", "unitOfMeasure", etc., are XML elements. Exceptions are made for a few parameters that indicate *refined semantics* of element types without affecting the content model. These are encoded as XML attributes on the parent element. Thus "observable" and "of", and also "members" are XML attributes. In the UML model we indicate this with the stereotype **`<<XSDattribute>>`** to override the default property mapping. We also use XML attributes in the standard way for identification, references and hyperlinks. .

7.1.1 DataTypes

By applying the GML pattern consistently the XML elements representing Quantity, Position, NominalCategory, OrderedCategory and Count will have XML Schema *complexContent*, with the "numerator" and "referenceSystem" appearing as sub-elements of the Value element.

However, in developing a model, there is an argument for recognising some classes as "primitive", as denoted by the <<DataType>> stereotype in the UML model. We then use a custom encoding for these classes. For example, the OGC paper on Units of Measure [UOM2001] recommends a more compact encoding for Quantity, in which the "amount" is promoted to the XML Schema *simpleContent* and the "units of measure" moved to an XML *attribute*.

In this paper, classes with the <<DataType>> stereotype have a simpleContent encoding. This approach is also consistent with using XML list structures for the compact list encoding for arrays.

Note that earlier versions of this recommendation used the complexContent syntax, and this was the syntax used in the OWS-1.1 demo. However, technical problems which inhibited us going straight with the simpleContent syntax have now been resolved. In this paper we present what we consider to be the preferred option.

7.2 XML schema for Values

The Value model presented in section 6.4 is used as the basis for a GML conformant XML Schema `value.xsd` (Annex A.2.1). This schema will be included amongst the GML 3 base schemas and contributes components to the GML namespace, so it has `targetNamespace="http://www.opengis.net/gml"`. The GML 3 schemas were still under development during the study reported here, so earlier versions of the report have illustrated instances which contain some variations which reflected earlier versions of the GML 3 schemas. The schemas and instances have been updated to reflect the latest available version of GML. GML 3 is scheduled for a code-freeze 26th April 2002, so the versions shown now are likely to be stable.

We use the built-in XML Schema simple types boolean, string, integer and double for the values as appropriate. GML provides several different null values through `gml:NullType`: viz. *inapplicable*, *missing*, *unknown* and *withheld*. The scalar types in `value.xsd` are based on XML schema "union" types defined in `basicTypes.xsd`, which make the null values available.

7.2.1 Scalar Values

The generic element `Position` from `value.xsd` is used for recording a temperature:

Code sample (ii)

```
<gml:Quantity observable="http://www.opengis.net/ows/observableTypes#waterTemperature"
uom="http://www.opengis.net/ows/CRS#Celsius">21.2</gml:Quantity>
```

or if the value is a null:

Code sample (iii)

```
<gml:Quantity observable="#waterTemperature" >missing</gml:Quantity>
```

7.2.2 Reference Values for Measurement Parameters

The values of the `observable`, `uom` and `frame` parameters are references to other resources, given as URIs. The dereferenced resource should contain enough information to identify and describe the semantics of the relevant concept.

The fully qualified URIs quoted above do not currently resolve to anything useful, but imply a suite of dictionaries, registries and other services that are required as part of a Sensor Web deployment.

The second example uses the "abbreviated XPointer syntax" [XPointer] in the value of `observable`. This allows us to refer to an elements with this identifier within the local document. Implicitly the "#" has the URL of the current document prepended in order to construct a fully qualified URI. The abbreviated style is used for clarity in most subsequent examples.

7.2.3 Value Collections

Unrelated values can be grouped into a `ValueCollection`:

Code sample (iv)

```
<gml:ValueCollection>
  <gml:valueMember>
    <gml:Position observable="#airTemperature" frame="#Celsius">21.2</gml:Position>
  </gml:valueMember>
  <gml:valueMember>
    <gml:Quantity observable="#barometricPressure" uom="#hPa">990.</gml:Quantity>
  </gml:valueMember>
</gml:ValueCollection>
```

Note that any of the members of a collection might themselves be collections, arrays, or lists. Complex values can be constructed by nesting these as necessary.

7.2.4 Composite Values

Wind velocity is a vector quantity, requiring two numbers for the horizontal component. `CompositeValue` can be used to record various decompositions:

Code sample (v)

```
<gml:CompositeValue observable="#windVelocity">
  <gml:component>
    <gml:Quantity observable="#Speed" uom="#mPs">14.7</gml:Quantity>
  </gml:component>
  <gml:component>
    <gml:Position observable="#Direction" frame="#degreesEast">315</gml:Position>
  </gml:component>
</gml:CompositeValue>
```

Code sample (vi)

```
<gml:CompositeValue observable="#windVelocity">
```

```
    <gml:component axis="north">
      <gml:Quantity observable="#Speed" uom="#mPs">10.4</gml:Quantity>
    </gml:component>
    <gml:component axis="east">
      <gml:Quantity observable="#Speed" uom="#mPs">-10.4</gml:Quantity>
    </gml:component>
  </gml:CompositeValue>
```

In the second example it is necessary to use the **`axis`** parameter to distinguish between the two Speed components.

7.2.5 Value Arrays

An array of temperature values may be encoded using the **`ValueArray`** element:

Code sample (vii)

```
<gml:ValueArray observable="#waterTemperature" uom="#Celsius">
  <gml:valueMembers>
    <gml:Quantity>16.5</gml:Quantity>
    <gml:Quantity>17.2</gml:Quantity>
    <gml:Quantity>18.4</gml:Quantity>
    <gml:Quantity>missing</gml:Quantity>
    <gml:Quantity>21.2</gml:Quantity>
  </gml:valueMembers>
</gml:ValueArray>
```

Note that since the members of the array are all scalar values of the same observable, and use the same frame, the values of both the **`observable`** and **`frame`** parameters are inherited from the container element with no ambiguity.

Since the members of this array are scalar values, we may also use a ValueList:

Code sample (viii)

```
<gml:QuantityList observable="#waterTemperature" uom="#Celsius">16.5 17.2 18.4 missing 21.2</gml:QuantityList>
```

Finally, the **`arrayMembers`** element may point to an external source for the values, using **`href`**. In this case the **`observable`** parameter on the array container should be used. This indicates the type of the member Values in the source, even if they are not marked up as XML.

Code sample (ix)

```
<gml:ValueArray observable="#Temperature">
  <gml:valueMembers xlink:href="externalFile.dat"/>
</gml:ValueArray>
```

This form is particularly useful if the **`ValueArray`** is used for the range-set of a coverage, for which the observed values are available in compact form in an external file.

7.3 XML for Measurements

The Measurement model, presented in section 6.3, is used as the basis for a GML conformant XML Schema `measurement.xsd` (Annex A.2.2). This schema `<include>`s the components from the `value.xsd` schema document. The measurement schema is *not* a GML 3 base schema. Currently it contributes components to the OWS namespace, so it has `targetNamespace="http://www.opengis.net/ows"`.

7.3.1 Simple Measurement – Single Value

To illustrate the encoding, we show a complete example of a `Measurement` of Dissolved Solids made at a particular sampling station, for which there is also a previous measurement available:

Code sample (x)

```
  <ows:Measurement gml:id="OD654">
    <ows:measuredAt xlink:href="http://www.opengis.net/ows/stations#s432"/>
    <gml:timeStamp>
      <gml:tInstant><gml:tPosition>2001-12-12</gml:tPosition></gml:tInstant>
    </gml:timeStamp>
    <ows:resultOf>
      <gml:Quantity observable="http://www.opengis.net/ows/observableTypes#DissolvedSolids"
uom="http://www.opengis.net/ows/uom#gpL">72.1</gml:Quantity>
    </ows:resultOf>
    <ows:measuredBy xlink:href="http://www.opengis.net/ows/sensors#TDS"/>
    <ows:relatedFeature xlink:role="previous observation" xlink:href="http://www.opengis.net/ows/archive#OD321"/>
  </ows:Measurement>
```

7.3.2 Measurement Collection – Heterogeneous Observed Values

An example of a `MeasurementCollection` including several `Measurements` with some different observables, but all made at the same location, might appear as follows:

```
<?xml version="1.0" encoding="UTF-8"?>
<ows:MeasurementCollection xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows  ./measurement.xsd">
  <gml:boundedBy><gml:null>unknown</gml:null></gml:boundedBy>
  <ows:measurementMember>
    <ows:Measurement gml:id="OD654">
      <ows:resultOf>
        <gml:Quantity observable="#DissolvedSolids" uom="#gpL">72.1</gml:Quantity>
      </ows:resultOf>
      <ows:measuredBy xlink:href="#TDS"/>
      <ows:relatedFeature xlink:role="previous observation" xlink:href="http://www.opengis.net/ows/archive#OD321"/>
    </ows:Measurement>
  </ows:measurementMember>
  <ows:measurementMember>
    <ows:Measurement>
      <ows:resultOf>
        <gml:CompositeValue observable="#windVelocity">
          <gml:valueComponent>
            <gml:Quantity observable="#Speed" uom="#mPs">14.7</gml:Quantity>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Position observable="#Direction" frame="#degreesEast">315.</gml:Position>
          </gml:valueComponent>
        </gml:CompositeValue>
      </ows:resultOf>
      <ows:measuredBy xlink:href="#WVL"/>
    </ows:Measurement>
  </ows:measurementMember>
  <ows:measurementMember>
    <ows:Measurement>
      <ows:resultOf>
        <gml:Position observable="#waterTemperature" frame="#Celsius">21.2</gml:Position>
      </ows:resultOf>
      <ows:measuredBy xlink:href="#TEMPC"/>
    </ows:Measurement>
  </ows:measurementMember>
  <ows:measuredAt xlink:href="http://www.opengis.net/ows/stations#s432"/>
  <gml:timeStamp>
    <gml:tInstant><gml:tPosition>2001-12-12</gml:tPosition></gml:tInstant>
  </gml:timeStamp>
  <gml:dictionaryReference gml:id="TDS" xlink:href="http://www.opengis.net/ows/sensors#TDS"/>
  <gml:dictionaryReference gml:id="WVL" xlink:href="http://www.opengis.net/ows/sensors#WVL"/>
  <gml:dictionaryReference gml:id="TEMPC" xlink:href="http://www.opengis.net/ows/sensors#TEMPC"/>
  <gml:dictionaryReference gml:id="gpL" xlink:href="http://www.opengis.net/ows/uom#gpL"/>
  <gml:dictionaryReference gml:id="mPs" xlink:href="http://www.opengis.net/ows/uom#mPs"/>
  <gml:dictionaryReference gml:id="degreesEast" xlink:href="http://www.opengis.net/ows/RS#degreesEast"/>
  <gml:dictionaryReference gml:id="Celsius" xlink:href="http://www.opengis.net/ows/RS#Celsius"/>
  <gml:dictionaryReference gml:id="DissolvedSolids"
xlink:href="http://www.opengis.net/ows/observableTypes#DissolvedSolids"/>
  <gml:dictionaryReference gml:id="windVelocity"
xlink:href="http://www.opengis.net/ows/observableTypes#windVelocity"/>
  <gml:dictionaryReference gml:id="Speed" xlink:href="http://www.opengis.net/ows/observableTypes#Speed"/>
  <gml:dictionaryReference gml:id="Direction" xlink:href="http://www.opengis.net/ows/observableTypes#Direction"/>
  <gml:dictionaryReference gml:id="waterTemperature"
xlink:href="http://www.opengis.net/ows/observableTypes#waterTemperature"/>
</ows:MeasurementCollection>
```

The **`dictionaryReference`** elements collect the external references together and allow abbreviated XPointers to be used locally within the document.

7.3.3 Measurement Array – Time Series

An array of measurements of temperature comprising a time-series might appear:

Code sample (xi)

```
<?xml version="1.0" encoding="UTF-8"?>
<ows:MeasurementArray xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows ./measurement.xsd"
observable="http://www.opengis.net/ows/observableTypes#Temperature"
frame="http://www.opengis.net/ows/RS#Celsius">
  <gml:boundedBy>
    <gml:Envelope gml:srsName="http://opengis.org/epsg#4326">
      <gml:coordinates>-92.0,33.0 -91.0,34.0</gml:coordinates>
    </gml:Envelope>
  </gml:boundedBy>
  <ows:measurementMembers>
    <ows:Measurement>
      <gml:timeStamp>
        <gml:tInstant><gml:tPosition>2001-12-03</gml:tPosition></gml:tInstant>
      </gml:timeStamp>
      <ows:resultOf>
        <gml:Position>16.5</gml:Position>
      </ows:resultOf>
    </ows:Measurement>
    <ows:Measurement>
      <gml:timeStamp>
        <gml:tInstant><gml:tPosition>2001-12-04</gml:tPosition></gml:tInstant>
      </gml:timeStamp>
      <ows:resultOf>
        <gml:Position>17.2</gml:Position>
      </ows:resultOf>
    </ows:Measurement>
    <ows:Measurement>
      <gml:timeStamp>
        <gml:tInstant><gml:tPosition>2001-12-05</gml:tPosition></gml:tInstant>
      </gml:timeStamp>
      <ows:resultOf>
        <gml:Position>18.3</gml:Position>
      </ows:resultOf>
    </ows:Measurement>
  </ows:measurementMembers>
  <ows:measuredAt xlink:href="http://www.opengis.net/ows/stations#s432"/>
  <ows:measuredBy xlink:href="http://www.opengis.net/ows/sensors#TEMPCb6"/>
</ows:MeasurementArray>
```

As with any **`MeasurementArray`** containing scalar observed values, the values of the **`observable`** and **`frame`** parameters may be inherited from the container. Since this is a time series of measurements at the same location using the same sensor, there is a single **`measuredAt`** and **`measuredBy`** element in the **`MeasurementArray`**, while each member **`Measurement`** has a specific time **`tInstant`**.

7.3.4 Measurement Array – Spatial Sampling

On the other hand, a spatial array of O2 concentration in water samples might appear:

Code sample (xii)

```
<?xml version="1.0" encoding="UTF-8"?>
<ows:MeasurementArray xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows ./measurement.xsd"
observable="http://www.opengis.net/ows/observableTypes#ChemicalConcentration"
uom="http://www.opengis.net/ows/uom#percent">
  <gml:boundedBy>
    <gml:Envelope gml:srsName="http://opengis.org/epsg#4326">
      <gml:coordinates>-92.0,33.0 -91.0,34.0</gml:coordinates>
    </gml:Envelope>
  </gml:boundedBy>
  <ows:measurementMembers>
    <ows:Measurement>
      <ows:measuredAt xlink:href="http://www.opengis.net/ows/stations#s432"/>
      <ows:resultOf>
        <gml:Quantity>0.05</gml:Quantity>
      </ows:resultOf>
    </ows:Measurement>
    <ows:Measurement>
      <ows:measuredAt xlink:href="http://www.opengis.net/ows/stations#s434"/>
      <ows:resultOf>
        <gml:Quantity>0.17</gml:Quantity>
      </ows:resultOf>
    </ows:Measurement>
    <ows:Measurement>
      <ows:measuredAt xlink:href="http://www.opengis.net/ows/stations#s532"/>
      <ows:resultOf>
        <gml:Quantity>0.075</gml:Quantity>
      </ows:resultOf>
    </ows:Measurement>
  </ows:measurementMembers>
  <gml:timeStamp>
    <gml:tInstant><gml:tPosition>2001-12-03</gml:tPosition></gml:tInstant>
  </gml:timeStamp>
  <ows:measuredBy xlink:href="http://www.opengis.net/ows/sensors#DOXP"/>
</ows:MeasurementArray>
```

In this example, since the locations of the measurements are all different, the **`measuredAt`** element is on each of the member **`Measurements,`** while the collection as a whole has a **`tInstant`** to indicate the timing.

The results of a survey of animals caught in a set of traps can be encoded:

Code sample (xiii)

```
<?xml version="1.0" encoding="UTF-8"?>
<ows:MeasurementArray xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows ./measurement.xsd"
observable="http://www.opengis.net/ows/observableTypes#BettongCount">
  <gml:dictionaryReference gml:id="JAtkinson">Jacqueline Atkinson</gml:dictionaryReference>
  <gml:boundedBy>
    <gml:Envelope gml:srsName="http://opengis.org/epsg#4326">
      <gml:coordinates>-92.0,33.0 -91.0,34.0</gml:coordinates>
    </gml:Envelope>
  </gml:boundedBy>
  <ows:measurementMembers>
    <ows:Measurement>
      <ows:measuredAt xlink:href="http://www.opengis.net/ows/traps#n432"/>
      <ows:resultOf>
        <gml:Count>2</gml:Count>
      </ows:resultOf>
    </ows:Measurement>
    <ows:Measurement>
      <ows:measuredAt xlink:href="http://www.opengis.net/ows/traps#n43"/>
      <ows:resultOf>
        <gml:Count>19</gml:Count>
      </ows:resultOf>
    </ows:Measurement>
    <ows:Measurement>
      <ows:measuredAt xlink:href="http://www.opengis.net/ows/traps#n32"/>
      <ows:resultOf>
        <gml:Count>0</gml:Count>
      </ows:resultOf>
    </ows:Measurement>
    <ows:Measurement>
      <ows:measuredAt xlink:href="http://www.opengis.net/ows/traps#n32"/>
      <ows:resultOf>
        <gml:Count observable="http://www.opengis.net/ows/observableTypes#SkinkCount">3</gml:Count>
      </ows:resultOf>
    </ows:Measurement>
  </ows:measurementMembers>
  <gml:timeStamp>
    <gml:tInstant><gml:tPosition>2001-12-03</gml:tPosition></gml:tInstant>
  </gml:timeStamp>
  <ows:measuredBy xlink:href="#JAtkinson"/>
</ows:MeasurementArray>
```

The value of the **`measuredBy`** property for the whole array identifies the person who performed the count. The **`measuredAt`** property for each measurement identifies a **`Feature`**, the trap, where the measurement was made. All of the Observed Values inherit the **`observable`** value **`BettongCount`** from the container, except in one case where a local value of **`SkinkCount`** overrides this.

7.3.5 Extended Examples

In Annex B we present some additional examples. These show the flexibility of the Measurement and Value schemas, and illustrate some patterns for applying them to specific examples. The examples include:

- Measurements of several weather parameters made at an array of stations
- Time series of measurements of air quality made at an array of stations
- Spatial survey of gravity measurements
- Set of multi-element assays on samples from an exploration drillhole
- Alternative (compact) encoding of the assay data

7.4 XML Schema Derivation of Typed Measurements

7.4.1 Fixing the Values of the Classifiers

It is also possible to derive more typed observed values from the generic components. `TypedValue.xsd` (Annex C.1) is built using the pattern shown in Figure 6. It declares a number of measurements used in environmental monitoring. Named value types are defined by doing three things:

i. assign a base Observed Value type to the measurement name. This is a structural pattern chosen from Boolean, Category, OrderedCategory, Quantity, Count and Position;

ii. bind the `observable` to a value from the ObservableType registry.

iii. (optionally) extend the type with classifiers representing axes for further sub-typing.

Thus a `TemperatureType` is defined by starting with `PositionType`, fixing the value of `observable` to point to "Temperature" taken from a suitable dictionary, and adding an `of` classifier so that the user can indicate the medium being measured. The `Temperature` element is then declared using this type.

Code sample (xiv)

```
 <xs:element name="Temperature" type="ows:TemperatureType" substitutionGroup="gml:Position"/>
 <xs:complexType name="TemperatureBaseType">
   <xs:annotation>
     <xs:documentation>fix the value of observable</xs:documentation>
   </xs:annotation>
   <xs:simpleContent>
     <xs:restriction base="gml:PositionType">
       <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#Temperature"/>
     </xs:restriction>
   </xs:simpleContent>
 </xs:complexType>
 <xs:complexType name="TemperatureType">
   <xs:annotation>
```

```
      <xs:documentation>add "of" and "in" to support subtyping - the dictionary used must make sense ...
</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="ows:TemperatureBaseType">
        <xs:attribute name="of" type="xs:anyURI" use="optional"/>
        <xs:attribute name="in" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
```

Using this schema, the information in the first example Code sample (ii) would appear:

Code sample (xv)

```
<ows:Temperature of="http://www.opengis.net/ows/domains#water"
frame="http://www.opengis.net/ows/CRS#Celsius">21.2</ows:Temperature>
```

The element name has changed, the `observable` is omitted, and an axis for sub-typing has appeared called `of`.

A `compositeValue` might be strongly typed in a similar way to produce a typed vector value. For example, this schema fragment defines the necessary components to build a Velocity element:

Code sample (xvi)

```
  <!-- Speed -->
  <xs:complexType name="SpeedType">
    <xs:simpleContent>
      <xs:restriction base="gml:QuantityType">
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#Speed"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="Speed" type="ows:SpeedType" substitutionGroup="gml:Quantity"/>
  <xs:complexType name="SpeedPropertyType">
    <xs:complexContent>
      <xs:restriction base="gml:ValueComponentType">
        <xs:sequence>
          <xs:element ref="ows:Speed"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
  <!-- Direction -->
  <xs:complexType name="DirectionType">
    <xs:simpleContent>
      <xs:restriction base="gml:PositionType">
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#Direction"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="Direction" type="ows:DirectionType" substitutionGroup="gml:Position"/>
  <xs:complexType name="DirectionPropertyType">
    <xs:complexContent>
      <xs:restriction base="gml:ValueComponentType">
        <xs:sequence>
          <xs:element ref="ows:Direction"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
  <!-- Velocity -->
  <xs:element name="Velocity" type="ows:VelocityType" substitutionGroup="gml:_Value"/>
  <xs:complexType name="VelocityBaseType">
    <xs:complexContent>
      <xs:restriction base="gml:CompositeValueType">
        <xs:choice>
          <xs:sequence>
            <xs:element name="magnitude" type="ows:SpeedPropertyType"/>
            <xs:element name="azimuth" type="ows:DirectionPropertyType"/>
            <xs:element name="inclination" type="ows:DirectionPropertyType" minOccurs="0"/>
          </xs:sequence>
          <xs:sequence>
            <xs:element name="velocityComponent" type="ows:SpeedPropertyType" minOccurs="2" maxOccurs="3"/>
          </xs:sequence>
        </xs:choice>
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#Velocity"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="VelocityType">
    <xs:complexContent>
      <xs:extension base="ows:VelocityBaseType">
        <xs:attribute name="of" type="xs:anyURI"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
```

which would lead to

Code sample (xvii)

```
    <ows:Velocity of="#wind">
      <ows:magnitude>
        <ows:Speed uom="#mPs">14.7</ows:Speed>
      </ows:magnitude>
      <ows:azimuth>
        <ows:Direction frame="#degreesEast">315</ows:Direction>
      </ows:azimuth>
    </ows:Velocity>
```

Going still further, other parameters and the additional classifiers can also be bound to fixed values for still stronger typing. This can be done in two ways.

Define a new type, such as `CelsiusWaterTemperatureType`, which binds "Celsius" to the `frame` parameter, and "Water" to the `of` classifier, and declare a new element `CelsiusWaterTemperature` using this type. Within the schema, this looks like

Code sample (xviii)

```
  <xs:include schemaLocation="./typedValueAndMeasurement.xsd"/>
  <xs:element name="CelsiusWaterTemperature" type="ows:CelsiusWaterTemperatureType"
substitutionGroup="gml:Position"/>
  <xs:complexType name="CelsiusWaterTemperatureType">
    <xs:simpleContent>
      <xs:restriction base="ows:TemperatureType">
        <xs:attribute name="frame" type="xs:anyURI" use="optional" fixed="http://www.opengis.net/ows/CRS#Celsius"/>
        <xs:attribute name="of" type="xs:anyURI" use="optional" fixed="http://www.opengis.net/ows/domains#water"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
```

with the instance becoming

Code sample (xix)

```
<ows:CelsiusWaterTemperature>21.2</ows:CelsiusWaterTemperature >
```

Redefine **`TemperatureType`**, by binding the parameters as described. This method means that the type of the existing **`Temperature`** element is overridden by the new model. Within a schema this looks like:

Code sample (xx)

```
  <xs:redefine schemaLocation="./typedValueAndMeasurement.xsd">
    <xs:complexType name="TemperatureType">
      <xs:simpleContent>
        <xs:restriction base="ows:TemperatureType">
          <xs:attribute name="frame" type="xs:anyURI" use="optional" fixed="http://www.opengis.net/swe/CRS#Celsius"/>
          <xs:attribute name="of" type="xs:anyURI" use="optional" fixed="http://www.opengis.net/swe/domains#water"/>
        </xs:restriction>
      </xs:simpleContent>
    </xs:complexType>
  </xs:redefine>
```

with the instance becoming

Code sample (xxi)

```
<ows:Temperature>21.2</ows:Temperature >
```

This method of type derivation – defining a type by binding fixed values to free parameters provided by its supertype – is very flexible. However, it requires that the definition of the immediate supertype has included all the axes that may be restricted.

7.4.2 Where is the Information?

In the schema examples given, the attributes whose values are fixed are **`use="optional"`**. This means that the attributes can be omitted from the instance document, but the *post-schema-validated information set* (PSVI) still includes the information. For example, in the PSVI any occurrence of the element **`CelsiusWaterTemperature`** includes the information

```
(observable, http://www.opengis.net/ows/observableTypes#Temperature)
```

from **`typedValue.xsd`**, and

```
(of, http://www.opengis.net/ows/domains#water)
(frame, http://www.opengis.net/ows/RS#Celsius)
```

from **`Y2measurement.xsd`**, even though it is not seen in the instance document.

For typed measurements, the parameter binding applying to the XML documents may occur in several places. At a schema level this happens successively, as each schema builds on types defined in the preceding schema. Each parameter value is a URI, which is expected to be a pointer to an entry in a standard dictionary or registry. In the examples shown above:

- `measurement.xsd` is the generic measurement schema, using Observed Value elements with only structural typing from `value.xsd`. Examples that use only this schema must set all parameters *in the instance document*.
- `typedValue.xsd` is a community schema derived from the generic value schema. XML element names have `observable` bound to an entry in an ObservableTypes registry *in the schema*. All other parameters are set *in the instance document*.
- The schema fragment in Code sample (xviii) inherits the community schema. New XML element names have additional bindings to entries in a referenceSystems registry, and to other measurement-specific registries *in the schema*. New element names are used. No parameters are set in the instance document.
- The schema fragment in Code sample (xx) inherits the community schema. The XML types are redefined to have additional bindings to entries in a referenceSystems registry, and to other measurement-specific registries *in the schema*. The element names from the community schema are used unchanged. No parameters are set in the instance document.

In order to build the PSVI for any instance document, it is strictly necessary to have access to the schema document. The standard XML Schema method is to use the `schemaLocation` on the root element of the document. In the final example Code sample (xxi) this might appear:

Code sample (xxii)

```
    <ows:Temperature
            xmlns:ows="http://www.opengis.net/ows"
            xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
            xsi:schemaLocation="http://www.opengis.net/ows
http://www.opengis.net/ows/registries/measurements/schemas/Y3measurement.xsd">21.2</ows:Temperature>
```

i.e., this indicates that a schema document found at
`http://www.opengis.net/ows/registries/measurements/schemas/Y3measurement.xsd`
should be used as the reference for all terms in the `http://www.opengis.net/ows` namespace.

7.4.3 Typed Values vs Observable Types

Note that the `typedValue` schema, and other schemas that specialise measurements, are *not* the same as an Observable Type registry or dictionary. The latter contains values which are referenced by the `observable` parameter.

An Observable Type registry is the authoritative source for *semantic* definitions of the observable of interest.

The `typedValue` schema binds the semantic definitions of observable types to the structural forms provided by the generic value schema.

We do not discuss the structure of the information in the registry further in this IPR.

7.5 Issues Outstanding and Future Work

The model and schemas that have been developed are suitable for deployment as is. But they do not cover the full range of requirements, and a number of associated components are outstanding. Briefly some of these are as follows:

1. The observed value schema does not include some important data types. For example, some measurements have results including **geometric** or **temporal** information (e.g. earthquakes). There are GML3 schemas for geometry and temporal, but elements from these cannot be used as Values or within Value Collections as they are currently defined.

2. The definition and encoding for **reference systems** is very limited. Some reference systems (classifications, non-spatial frames) are not covered at all.

3. Methods to describe **measurement methodologies** other than instruments and sensors are needed. For example, estimates of values may be generated by computational simulators or numerical process modeling system, and by human observers. All of these may be combined in an analysis. These non-sensor systems require a different parameterisation than sensors, but the basis for this is not well understood.

4. The relationship between the structure of a sensor package and the structure of a composite measurement has not been examined. In particular, we need to be able to relate **components** of a complex value to (real or virtual) elements within a sensor package.

5. ValueArrays and ValueLists were designed to hold sets of values that might constitute the range set of a discrete coverage. However, the details of the implementation of an encoding for **coverages** in GML, or in a WCS, have not been fully explored.

6. An SCS depends on many information components: schemas and instances for values and measurements, for sensors, for observable types, for reference systems. Some of these components will need to be made available by registry services, and some by the sensor services themselves. Which components are managed by which service is not fully understood. Who is responsible for maintenance, access control, etc are significant architectural issues, which in turn will feed back to the **factoring** of information between the various components to accomplish real use cases.

7. Information **validation** depends on interactions between various components. Who or which components are responsible for validation has barely been considered.

8. The use of XML Schema based languages allows us to impose reasonable order on the structure of the information. However, significant **semantics** are associated with various aspects, which requires either implicit community understanding and approval, or associated formal semantic layers, probably using other language tools which are more appropriate.

9. Furthermore, XML Schemas constrain the validity of instance documents,but they only very weakly prescribe the pattern of application in a deployed system. Typically there are several options for encoding a particular instance, some more efficient and transparent than others. There is a need for more experiments applying the measurement schemas to varied real data, in order to determine best practice and enable the compilation of rules or guidelines.

Annex A
(normative)

XML Schemas for Measurements

A.1 General

The XML schemas included here are fully conformant GML Application Schemas for Measurements. The `schemaLocation` hints will work unchanged if the schemas are placed in a subdirectory created within the `applications` directory in the standard GML3 distribution.

A.2 Generic Schemas

A.2.1 Utility Schemas

The utility schema `observationAndValue.xsd` includes two component schema documents in the GML namespace. It is used to import the required components from the GML namespace into derived schemas such as `measurement.xsd`.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.opengis.net/gml" xmlns:gml="http://www.opengis.net/gml"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:annotation>
    <xs:documentation>
    observationAndValue.xsd
    Utility schema which simply includes both observation and value</xs:documentation>
  </xs:annotation>
  <!-- ==============================================================
      includes and imports
     ============================================================== -->
  <xs:include schemaLocation="../../base/observation.xsd"/>
  <xs:include schemaLocation="../../base/value.xsd"/>
</xs:schema>
```

A.2.1 Value Schema

In this schema we derive XML types for six generic Value types and several generic Value collections. Elements declared in this schema can be used directly in data instances, or may be used as components of Measurements.

Note that the definitions of several primitive simpleContent types, such as `doubleOrNull`, are now found in the GML 3 schema `basicTypes.xsd`.

See also the note Need for Position and OrderedCategory Types in Section 6.4.1.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.opengis.net/gml" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:sch="http://www.ascc.net/xml/schematron" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified" attributeFormDefault="unqualified"
version="0WS1">
 <xs:annotation>
  <xs:appinfo>value.xsd</xs:appinfo>
  <xs:documentation>
  Copyright (c) 2002 OGC, All Rights Reserved.

GML conformant schema for Observed Values in which the
* scalar Value types have their values recorded in simpleContent elements
* complex Value types are built recursively

Compact "list" types are also provided.

2002-02-26 added ValueRange
2002-03-11 properties based on AssociationType,  moved *OrNull and *List types to gmlBase
2002-03-14 removed observable attribute ...
2002-03-15 put it back as a local attribute
2002-03-28 create referenceSystem attribute group.
2002-04-10 re-define Observed(Value)Types as extensions of basic types

All are unified into a single hierarchy using substitutionGroups using some abstract head elements
</xs:documentation>
 </xs:annotation>
 <!-- ====================================================================== -->
 <xs:include schemaLocation="./gmlBase.xsd"/>
 <xs:include schemaLocation="./geometry.xsd"/>
 <xs:include schemaLocation="./temporal.xsd"/>
 <!-- should abstract substitution group heads _Geometry and _tObject be moved to gmlBase so that this schema does
not have to import those complete schemas? -->
 <!-- ====================================================================== -->
 <xs:element name="_Value" abstract="true" substitutionGroup="gml:_Object">
  <xs:annotation>
   <xs:documentation>
   head Value element is abstract -
   this means that potentially either simpleContent or complexContent elements can substitute
   </xs:documentation>
  </xs:annotation>
 </xs:element>
 <!-- ====================================================================== -->
 <!-- ================= Scalar Values ======================== -->
 <xs:element name="_ScalarValue" abstract="true" substitutionGroup="gml:_Value">
  <xs:annotation>
   <xs:documentation>Should be type="xs:anySimpleType" except MSXML does not recognise
anySimpleType</xs:documentation>
  </xs:annotation>
 </xs:element>
 <!-- ====================================================================== -->
 <xs:element name="_ValueList" abstract="true" substitutionGroup="gml:_Value">
  <xs:annotation>
   <xs:documentation>Use these elements for a compact encoding of arrays of scalar values</xs:documentation>
  </xs:annotation>
 </xs:element>
 <!-- ====================================================================== -->
 <!-- ====================== Boolean ======================= -->
 <xs:complexType name="ObservedBooleanType">
  <xs:annotation>
   <xs:documentation>true/false or 1/0.  Adds an observable parameter for weak semantic typing.   </xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
   <xs:extension base="gml:booleanOrNull">
    <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
   </xs:extension>
  </xs:simpleContent>
 </xs:complexType>
 <xs:element name="Boolean" type="gml:ObservedBooleanType" substitutionGroup="gml:_ScalarValue"/>
 <xs:complexType name="ObservedBooleanListType">
```

```
  <xs:simpleContent>
    <xs:extension base="gml:stringOrNullList">
      <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:element name="BooleanList" type="gml:ObservedBooleanListType" substitutionGroup="gml:_ValueList"/>
<!-- ===================================================================== -->
<!-- ======================= Category ========================= -->
<xs:complexType name="ObservedCategoryType">
  <xs:annotation>
    <xs:documentation>Term from a controlled codeSpace.
    Adds an observable parameter to NominalType for weak semantic typing.  </xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="gml:CategoryType">
      <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:element name="Category" type="gml:ObservedCategoryType" substitutionGroup="gml:_ScalarValue"/>
<xs:complexType name="ObservedCategoryListType">
  <xs:simpleContent>
    <xs:extension base="gml:CategoryListType">
      <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:element name="CategoryList" type="gml:ObservedCategoryListType" substitutionGroup="gml:_ValueList"/>
<!-- ===================================================================== -->
<!-- ========================== Quantity =========================== -->
<xs:complexType name="ObservedQuantityType">
  <xs:annotation>
    <xs:documentation>Number with a scale.
    Adds an observable parameter to MeasureType for weak semantic typing.
    The name Quantity is used since this conforms with general usage.  </xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="gml:QuantityType">
      <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:element name="Quantity" type="gml:ObservedQuantityType" substitutionGroup="gml:_ScalarValue"/>
<xs:complexType name="ObservedQuantityListType">
  <xs:simpleContent>
    <xs:extension base="gml:QuantityListType">
      <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:element name="QuantityList" type="gml:ObservedQuantityListType" substitutionGroup="gml:_ValueList"/>
<!-- ===================================================================== -->
<!-- ========================= Count ========================= -->
<xs:complexType name="ObservedCountType">
  <xs:annotation>
    <xs:documentation>Integer amount with no scale.
    Adds an observable parameter for weak semantic typing.   </xs:documentation>
  </xs:annotation>
  <xs:simpleContent>
    <xs:extension base="gml:integerOrNull">
      <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:simpleContent>
</xs:complexType>
<xs:element name="Count" type="gml:ObservedCountType" substitutionGroup="gml:_ScalarValue"/>
<xs:complexType name="ObservedCountListType">
  <xs:simpleContent>
    <xs:extension base="gml:integerOrNullList">
```

```
        <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="CountList" type="gml:ObservedCountListType" substitutionGroup="gml:_ValueList"/>
  <!-- =========================================================================== -->
  <!--                compound Value types                    -->
  <!-- ========================================================================= -->
  <!-- ========================= CompositeValue ======================== -->
  <xs:complexType name="CompositeValueType">
    <xs:annotation>
      <xs:documentation>A CompositeValue is conceptually a single Value,
      usually collected by some complex sensor or sensor package,
      but represented by several components - e.g. Vector, Tensor.

      The observable indicates the type of the complex value as a whole.
      The components will also have an observable to indicate their types.  </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractGMLType">
        <xs:sequence>
          <xs:element ref="gml:valueComponent" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="CompositeValue" type="gml:CompositeValueType" substitutionGroup="gml:_Value">
    <xs:annotation>
      <xs:documentation>Vector or tensor value</xs:documentation>
    </xs:annotation>
  </xs:element>
  <!-- ========================================================================= -->
  <!-- ==================== ValueCollection ========================= -->
  <xs:complexType name="ValueCollectionType">
    <xs:annotation>
      <xs:documentation>ValueCollection is built up from other Values.  No other restrictions.  No implications for ordering.
</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:AbstractGMLType">
        <xs:sequence>
          <xs:element ref="gml:valueMember" minOccurs="0" maxOccurs="unbounded"/>
          <xs:element ref="gml:valueMembers" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="ValueCollection" type="gml:ValueCollectionType" substitutionGroup="gml:_Value">
    <xs:annotation>
      <xs:documentation>Arbitrary combination of values</xs:documentation>
    </xs:annotation>
  </xs:element>
  <!-- ========================================================================= -->
  <!-- ======================== ValueArray ========================= -->
  <xs:complexType name="ValueArrayType">
    <xs:annotation>
      <xs:documentation>ValueArray is a collection of homogeneous values.
      The member values may be composite, collections or arrays.
      Since the members are homgeneous, the common parameters can be attached to the container, and inherited by the
members:
      referenceSystem - choose the appropriate one for the type of the members;
      observable - this is implicitly inherited by the immediate child member values,
      though if the children are themselves composite values then their children require their own observables for
disambiguation.
</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:ValueCollectionType">
```

```
        <xs:attributeGroup ref="gml:referenceSystem"/>
        <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="ValueArray" type="gml:ValueArrayType" substitutionGroup="gml:_Value">
    <xs:annotation>
      <xs:appinfo>
        <sch:pattern name="Check either codeSpace or uom not both">
          <sch:rule context="gml:ValueArray">
            <sch:report test="@codeSpace and @uom">ValueArray may not carry both a reference to a codeSpace and a
uom</sch:report>
          </sch:rule>
        </sch:pattern>
      </xs:appinfo>
      <xs:documentation>Use this element for homogeneous arrays of CompositeValues and other ValueCollections.
_ValueList is preferred for arrays of Scalar Values since this is more efficient. </xs:documentation>
    </xs:annotation>
  </xs:element>
  <!-- attribute group required for ValueArray -->
  <!-- since "choice" is not available for attribute groups, an external constraint (e.g. Schematron) is required to enforce the
selection of only one of these -->
  <xs:attributeGroup name="referenceSystem">
    <xs:attribute name="codeSpace" type="xs:anyURI" use="optional"/>
    <xs:attribute name="uom" type="xs:anyURI" use="optional"/>
  </xs:attributeGroup>
  <!-- =================================================================== -->
  <!-- ====================== ValueRange =========================== -->
  <xs:complexType name="ValueExtentType">
    <xs:sequence>
      <xs:element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="low" type="gml:ValuePropertyType"/>
      <xs:element name="high" type="gml:ValuePropertyType"/>
    </xs:sequence>
    <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
  </xs:complexType>
  <!-- -->
  <xs:element name="ValueExtent" type="gml:ValueExtentType" substitutionGroup="gml:_Value">
    <xs:annotation>
      <xs:documentation>Utility element to store 2-point ranges</xs:documentation>
    </xs:annotation>
  </xs:element>
  <!-- =================================================================== -->
  <!-- ==================== pieces needed for compositing ==================== -->
  <xs:complexType name="ValuePropertyType">
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:_Value"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- =================================================================== -->
  <xs:element name="valueProperty" type="gml:ValuePropertyType" substitutionGroup="gml:_property"/>
  <xs:element name="valueMember" type="gml:ValuePropertyType" substitutionGroup="gml:_property"/>
  <!-- =================================================================== -->
  <xs:complexType name="ValueComponentType">
    <xs:annotation>
      <xs:documentation>add "axis" attribute to indicate an index or ordering for vector and tensor values,
      in cases where the "observable" on the component value is insufficient</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:ValuePropertyType">
        <xs:attribute name="axis" type="xs:string" use="optional"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="valueComponent" type="gml:ValueComponentType" substitutionGroup="gml:valueProperty"/>
  <!-- =================================================================== -->
  <xs:complexType name="ValueArrayPropertyType">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
```

```
      <xs:element ref="gml:_Value"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <xs:element name="valueMembers" type="gml:ValueArrayPropertyType" substitutionGroup="gml:_property"/>
  <!-- ====================================================================== -->
  <!-- ====================== utility typed valueProperty types ===================  -->
  <xs:complexType name="BooleanPropertyType">
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:Boolean"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <xs:complexType name="CategoryPropertyType">
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:Category"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <xs:complexType name="QuantityPropertyType">
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:Quantity"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <xs:complexType name="CountPropertyType">
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:Count"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- ====================================================================== -->
  <!-- ================ specific metaData pieces ==================== -->
  <xs:complexType name="ValueMetaDataPropertyType">
    <xs:annotation>
      <xs:documentation> Use this to provide a gloss on the measurement(s)</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:extension base="gml:MetaDataPropertyType">
        <xs:sequence>
          <xs:element name="ValueDescription" type="gml:StringOrRefType" minOccurs="0"/>
        </xs:sequence>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <xs:element name="valueMetaData" type="gml:ValueMetaDataPropertyType"
substitutionGroup="gml:metaDataProperty"/>
  <!-- ====================================================================== -->
  <!-- OrderedCategory -->
  <xs:complexType name="OrderedCategoryType">
    <xs:annotation>
      <xs:documentation>Term from an ordered list or tree</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="gml:stringOrNull">
        <xs:attribute name="codeFrame" type="xs:anyURI" use="optional"/>
        <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="OrderedCategory" type="gml:OrderedCategoryType" substitutionGroup="gml:_ScalarValue"/>
  <xs:complexType name="OrderedCategoryListType">
    <xs:simpleContent>
      <xs:extension base="gml:stringOrNullList">
        <xs:attribute name="codeFrame" type="xs:anyURI" use="optional"/>
        <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="OrderedCategoryList" type="gml:OrderedCategoryListType" substitutionGroup="gml:_ValueList"/>
  <!-- Position -->
  <xs:complexType name="PositionType">
    <xs:annotation>
      <xs:documentation>Location defined by a number in a reference frame defining a scale and origin.  Value on an
interval scale.  </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="gml:doubleOrNull">
        <xs:attribute name="frame" type="xs:anyURI" use="optional"/>
        <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="Position" type="gml:PositionType" substitutionGroup="gml:_ScalarValue"/>
  <xs:complexType name="PositionListType">
    <xs:simpleContent>
      <xs:extension base="gml:doubleOrNullList">
        <xs:attribute name="frame" type="xs:anyURI" use="optional"/>
        <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name="PositionList" type="gml:PositionListType" substitutionGroup="gml:_ValueList"/>
  <!-- -->
  <xs:complexType name="OrderedCategoryPropertyType">
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:OrderedCategory"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <xs:complexType name="PositionPropertyType">
    <xs:sequence minOccurs="0">
      <xs:element ref="gml:Position"/>
    </xs:sequence>
    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
  </xs:complexType>
  <!-- -->
</xs:schema>
```

A.2.2 Measurement Schema

In this schema we derive XML types for measurements, which map values to location, timing and sensor. The schema imports the Value declarations.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.opengis.net/ows" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://www.opengis.net/ows"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified" attributeFormDefault="unqualified"
version="0.96">
  <xs:annotation>
    <xs:documentation>
measurement.xsd

A GML conformant schema
for Measurements
i.e. Time and Space located Features with properties containing Measurements made by Sensors

derived from gml3 Observations
restricts valueOf properties to be resultOf properties containing a Measurement
adds Measurement collections

SJDC  2002-04-01
</xs:documentation>
  </xs:annotation>
  <!-- ====================================================================== -->
```

```
<!-- bring in other schemas -->
<xs:import namespace="http://www.opengis.net/gml" schemaLocation="./observationAndValue.xsd"/>
<!-- ====================================================================== -->
<!-- Properties of Measurements -->
<xs:element name="measuredAt" type="gml:LocationPropertyType" substitutionGroup="gml:location"/>
<xs:element name="resultOf" type="gml:ValuePropertyType" substitutionGroup="gml:valueOf"/>
<xs:element name="errorIn" type="ows:ErrorPropertyType" substitutionGroup="gml:valueOf"/>
<!-- ====================================================================== -->
<!-- ====================================================================== -->
<!-- Measurement types -->
<xs:complexType name="MeasurementBaseType">
  <xs:annotation>
    <xs:documentation>restrict
    location to be measuredAt
    valueOf to be resultOf</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="gml:ObservationType">
      <xs:sequence>
        <xs:element ref="gml:metaDataProperty" minOccurs="0"/>
        <xs:element ref="gml:description" minOccurs="0"/>
        <xs:element ref="gml:name" minOccurs="0"/>
        <xs:element ref="gml:boundedBy" minOccurs="0"/>
        <xs:element ref="ows:measuredAt" minOccurs="0"/>
        <xs:group ref="gml:dynamicProperties"/>
        <xs:element ref="ows:resultOf"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="MeasurementType">
  <xs:annotation>
    <xs:documentation>   add errrorIn, measuredBy and  relatedFeature
          errorIn gives an estimate of the error of the result.
          measuredBy points to the sensor or party or software that generated the measurement
          relatedFeature is a pointer to another arbitrary feature

          do we need a "measuredOn" property (in addition to measuredAt)??

</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="ows:MeasurementBaseType">
      <xs:sequence>
        <xs:element ref="ows:errorIn" minOccurs="0"/>
        <xs:element ref="ows:measuredBy" minOccurs="0"/>
        <xs:element ref="ows:relatedFeature" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="gml:dictionaryReference" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="Measurement" type="ows:MeasurementType" substitutionGroup="gml:Observation"/>
<!-- ====================================================================== -->
<!-- Measurement Collection types -->
<xs:complexType name="MeasurementCollectionBaseType">
  <xs:annotation>
    <xs:documentation>restrict member types</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:restriction base="gml:FeatureCollectionType">
      <xs:sequence>
        <xs:element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="gml:description" minOccurs="0"/>
        <xs:element ref="gml:name" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="gml:boundedBy"/>
        <xs:element ref="ows:measurementMember" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="ows:measurementMembers" minOccurs="0"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
<xs:complexType name="MeasurementCollectionType">
  <xs:annotation>
    <xs:documentation>add measuredAt, dynamicProperties, measuredBy, relatedFeature</xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="ows:MeasurementCollectionBaseType">
      <xs:sequence>
        <xs:element ref="ows:measuredAt" minOccurs="0"/>
        <xs:group ref="gml:dynamicProperties"/>
        <xs:element ref="ows:measuredBy" minOccurs="0"/>
        <xs:element ref="ows:relatedFeature" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="gml:dictionaryReference" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="MeasurementCollection" type="ows:MeasurementCollectionType"
substitutionGroup="ows:Measurement"/>
<!-- ====================================================================== -->
<xs:complexType name="MeasurementArrayType">
  <xs:annotation>
    <xs:documentation>
    As MeasurementCollection, except has homogeneous measurementMembers
    optional "members" attribute for weak typing
    optional observable and referenceSystem attributes - inherited by ScalarValues of member measurements unless
overridden.
    </xs:documentation>
  </xs:annotation>
  <xs:complexContent>
    <xs:extension base="ows:MeasurementCollectionType">
      <xs:attribute name="members" type="xs:Name" use="optional"/>
      <xs:attributeGroup ref="gml:referenceSystem"/>
      <xs:attribute name="observable" type="xs:anyURI" use="optional"/>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="MeasurementArray" type="ows:MeasurementArrayType" substitutionGroup="ows:Measurement"/>
<!-- components of Measurement Collections -->
<xs:complexType name="MeasurementMemberType">
  <xs:complexContent>
    <xs:restriction base="gml:AssociationType">
      <xs:sequence>
        <xs:element ref="ows:Measurement" minOccurs="0"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="measurementMember" type="ows:MeasurementMemberType"
substitutionGroup="gml:featureMember"/>
<!-- -->
<xs:complexType name="MeasurementMembersType">
  <xs:complexContent>
    <xs:restriction base="gml:ArrayAssociationType">
      <xs:sequence>
        <xs:element ref="ows:Measurement" minOccurs="0" maxOccurs="unbounded"/>
      </xs:sequence>
    </xs:restriction>
  </xs:complexContent>
</xs:complexType>
<!-- -->
<xs:element name="measurementMembers" type="ows:MeasurementMembersType"
substitutionGroup="gml:featureMembers"/>
```

```
 <!-- ====================================================================== -->
 <xs:complexType name="ErrorPropertyType">
   <xs:annotation>
     <xs:documentation>add a "type" parameter</xs:documentation>
   </xs:annotation>
   <xs:complexContent>
     <xs:extension base="gml:ValuePropertyType">
       <xs:attribute name="errorType" type="xs:string"/>
     </xs:extension>
   </xs:complexContent>
 </xs:complexType>
 <!-- ====================================================================== -->
 <!-- Related Features -->
 <xs:complexType name="RelatedFeatureType">
   <xs:annotation>
     <xs:documentation>xlink:role compulsory in this context?</xs:documentation>
   </xs:annotation>
   <xs:sequence>
     <xs:element ref="gml:_Feature" minOccurs="0"/>
   </xs:sequence>
   <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
 </xs:complexType>
 <!-- -->
 <xs:element name="relatedFeature" type="ows:RelatedFeatureType"/>
 <xs:element name="measuredBy" type="ows:RelatedFeatureType" substitutionGroup="ows:relatedFeature">
   <xs:annotation>
     <xs:documentation>This element contains or points to a Sensor description in SensorML.
     A restricted type definition will be constructed when SensorML is available. </xs:documentation>
   </xs:annotation>
 </xs:element>
 <!-- -->
 <xs:complexType name="RelatedMeasurementType">
   <xs:complexContent>
     <xs:restriction base="ows:RelatedFeatureType">
       <xs:sequence>
         <xs:element ref="ows:Measurement" minOccurs="0"/>
       </xs:sequence>
     </xs:restriction>
   </xs:complexContent>
 </xs:complexType>
 <!-- -->
 <xs:element name="relatedMeasurement" type="ows:RelatedMeasurementType"
substitutionGroup="ows:relatedFeature"/>
 <!-- ====================================================================== -->
 <xs:complexType name="FeatureType">
   <xs:annotation>
     <xs:documentation>Concrete generic feature type.</xs:documentation>
   </xs:annotation>
   <xs:complexContent>
     <xs:extension base="gml:AbstractFeatureType"/>
   </xs:complexContent>
 </xs:complexType>
 <xs:element name="Station" type="ows:FeatureType" substitutionGroup="gml:_Feature"/>
 <xs:element name="Specimen" type="ows:FeatureType" substitutionGroup="gml:_Feature"/>
</xs:schema>
```

Annex B
(informative)

Extended examples

B.1 OWS 1.1 Examples

B.1.1 SAIC Weather Station Measurements

This example is a **`MeasurementArray`**, containing a set of member **`Measurements`**, the result of each of which is a **`CompositeValue`** comprising some weather observations. Alternatively the values could be packaged in a **`ValueCollection`**. The **`CompositeValue`** element is normally used for the value of a single **`measurement`** since it would have a single **`measuredBy`** parameter whose value would be a Sensor or SensorPackage. However, in this example the sensor is not identified.

```xml
<?xml version="1.0"?>
<ows:MeasurementArray xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows ./measurement.xsd" observable="#weather">
  <gml:name>SAIC Weather Monitoring System</gml:name>
  <gml:boundedBy>
    <gml:null>unknown</gml:null>
  </gml:boundedBy>
  <ows:measurementMembers>
    <ows:Measurement>
      <ows:measuredAt>
        <ows:Station gml:id="KSEE">
          <gml:name>San Diego / Gillespie, United States</gml:name>
          <gml:location>
            <gml:Point>
              <gml:coordinates>-116.96666666666667,32.833333333333336</gml:coordinates>
            </gml:Point>
          </gml:location>
        </ows:Station>
      </ows:measuredAt>
      <gml:timeStamp>
        <gml:tInstant>
          <gml:tPosition>2002-01-18T20:47:00.000</gml:tPosition>
        </gml:tInstant>
      </gml:timeStamp>
      <ows:resultOf>
        <gml:CompositeValue>
          <gml:valueComponent>
            <gml:Category observable="#cloudcover">BKN</gml:Category>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Quantity observable="#temperature">21.0</gml:Quantity>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Quantity observable="#humidity">23</gml:Quantity>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Quantity observable="#pressure">0.0</gml:Quantity>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Quantity observable="#windDirection">270</gml:Quantity>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Quantity observable="#windSpeed">6.0</gml:Quantity>
          </gml:valueComponent>
        </gml:CompositeValue>
      </ows:resultOf>
    </ows:Measurement>
    <ows:Measurement>
      <ows:measuredAt>
        <ows:Station gml:id="KCZZ">
          <gml:name>Campo, United States</gml:name>
          <gml:location>
            <gml:Point>
              <gml:coordinates>-116.46833333333333,32.626111111111108</gml:coordinates>
            </gml:Point>
          </gml:location>
        </ows:Station>
      </ows:measuredAt>
      <gml:timeStamp>
        <gml:tInstant>
          <gml:tPosition>2002-01-18T20:52:00.000</gml:tPosition>
        </gml:tInstant>
      </gml:timeStamp>
      <ows:resultOf>
        <gml:CompositeValue>
          <gml:valueComponent>
            <gml:Category observable="#cloudcover">missing</gml:Category>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Quantity observable="#temperature">13.0</gml:Quantity>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Quantity observable="#humidity">23</gml:Quantity>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Quantity observable="#pressure">1018.0</gml:Quantity>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Quantity observable="#windDirection">70</gml:Quantity>
          </gml:valueComponent>
          <gml:valueComponent>
            <gml:Quantity observable="#windSpeed">18.0</gml:Quantity>
          </gml:valueComponent>
        </gml:CompositeValue>
      </ows:resultOf>
    </ows:Measurement>
  </ows:measurementMembers>
</ows:MeasurementArray>
```

B.1.2 Polexis Air Monitoring System

This example is a **`MeasurementCollection`**, containing a set of member **`MeasurementCollections`**, in turn containing a set of **`MeasurementArrays`** of different air quality values, each comprising a time-series of **`Measurements`**.

```
<ows:MeasurementCollection xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/ows ./measurement.xsd">
  <gml:name>New York State Ambient Air Monitoring System</gml:name>
  <gml:boundedBy>
    <gml:null>unknown</gml:null>
  </gml:boundedBy>
  <ows:measurementMember>
    <ows:MeasurementCollection>
      <gml:boundedBy>
  <gml:null>unknown</gml:null>
</gml:boundedBy>
<ows:measurementMember>
  <ows:MeasurementArray observable="SO2" uom="ppm">
    <gml:boundedBy>
      <gml:null>unknown</gml:null>
    </gml:boundedBy>
    <ows:measurementMembers>
      <ows:Measurement>
        <gml:timeStamp>
          <gml:tInstant>
            <gml:tPosition>2002-01-14T12:00:00Z</gml:tPosition>
          </gml:tInstant>
        </gml:timeStamp>
        <ows:resultOf>
          <gml:Quantity>0.016</gml:Quantity>
        </ows:resultOf>
      </ows:Measurement>
      <ows:Measurement>
        <gml:timeStamp>
          <gml:tInstant>
            <gml:tPosition>2002-01-14T13:00:00Z</gml:tPosition>
          </gml:tInstant>
        </gml:timeStamp>
        <ows:resultOf>
          <gml:Quantity>0.016</gml:Quantity>
        </ows:resultOf>
      </ows:Measurement>
      <ows:Measurement>
        <gml:timeStamp>
          <gml:tInstant>
            <gml:tPosition>2002-01-14T14:00:00Z</gml:tPosition>
          </gml:tInstant>
        </gml:timeStamp>
        <ows:resultOf>
          <gml:Quantity>0.014</gml:Quantity>
        </ows:resultOf>
      </ows:Measurement>
    </ows:measurementMembers>
  </ows:MeasurementArray>
</ows:measurementMember>
<ows:measurementMember>
  <ows:MeasurementArray observable="24Hr SO2" uom="ppm">
    <gml:boundedBy>
      <gml:null>unknown</gml:null>
    </gml:boundedBy>
    <ows:measurementMembers>
      <ows:Measurement>
        <gml:timeStamp>
          <gml:tInstant>
            <gml:tPosition>2002-01-14T23:00:00Z</gml:tPosition>
          </gml:tInstant>
        </gml:timeStamp>
        <ows:resultOf>
          <gml:Quantity>missing</gml:Quantity>
        </ows:resultOf>
      </ows:Measurement>
      <ows:Measurement>
        <gml:timeStamp>
          <gml:tInstant>
            <gml:tPosition>2002-01-15T00:00:00Z</gml:tPosition>
          </gml:tInstant>
        </gml:timeStamp>
        <ows:resultOf>
          <gml:Quantity>0.015</gml:Quantity>
        </ows:resultOf>
      </ows:Measurement>
      <ows:Measurement>
        <gml:timeStamp>
```

```
            <gml:tInstant>
              <gml:tPosition>2002-01-15T01:00:00Z</gml:tPosition>
            </gml:tInstant>
          </gml:timeStamp>
          <ows:resultOf>
            <gml:Quantity>0.015</gml:Quantity>
          </ows:resultOf>
        </ows:Measurement>
      </ows:measurementMembers>
    </ows:MeasurementArray>
  </ows:measurementMember>
  <ows:measuredAt>
    <ows:Station gml:id="_709310">
      <gml:description xlink:href="http://www.dec.state.ny.us/website/dar/bts/airmon/709310site.htm"/>
      <gml:name>P.S. 59</gml:name>
      <gml:location>
        <gml:Point>
          <gml:coordinates>-73.96708,40.75982</gml:coordinates>
        </gml:Point>
      </gml:location>
    </ows:Station>
  </ows:measuredAt>
 </ows:MeasurementCollection>
</ows:measurementMember>
<ows:measurementMember>
 <ows:MeasurementCollection>
  <gml:boundedBy>
    <gml:null>unknown</gml:null>
  </gml:boundedBy>
  <ows:measurementMember>
    <ows:MeasurementArray observable="PM2.5" uom="UG/M3">
      <gml:boundedBy>
        <gml:null>unknown</gml:null>
      </gml:boundedBy>
      <ows:measurementMembers>
        <ows:Measurement>
          <gml:timeStamp>
            <gml:tInstant>
              <gml:tPosition>2002-01-14T12:00:00Z</gml:tPosition>
            </gml:tInstant>
          </gml:timeStamp>
          <ows:resultOf>
            <gml:Quantity>17.62</gml:Quantity>
          </ows:resultOf>
        </ows:Measurement>
        <ows:Measurement>
          <gml:timeStamp>
            <gml:tInstant>
              <gml:tPosition>2002-01-14T13:00:00Z</gml:tPosition>
            </gml:tInstant>
          </gml:timeStamp>
          <ows:resultOf>
            <gml:Quantity>13.9</gml:Quantity>
          </ows:resultOf>
        </ows:Measurement>
        <ows:Measurement>
          <gml:timeStamp>
            <gml:tInstant>
              <gml:tPosition>2002-01-14T14:00:00Z</gml:tPosition>
            </gml:tInstant>
          </gml:timeStamp>
          <ows:resultOf>
            <gml:Quantity>18.37</gml:Quantity>
          </ows:resultOf>
        </ows:Measurement>
      </ows:measurementMembers>
    </ows:MeasurementArray>
  </ows:measurementMember>
  <ows:measurementMember xlink:href="anotherArraySomewhere"/>
```

```
    </ows:MeasurementCollection>
  </ows:measurementMember>
  <ows:measuredAt>
    <ows:Station gml:id="_709315">
      <gml:description xlink:href="http//www.dec.state.ny.us/website/dar/bts/airmon/709315site.htm"/>
      <gml:name>IS 143</gml:name>
      <gml:location>
        <gml:Point>
          <gml:coordinates>-73.93094,40.84886</gml:coordinates>
        </gml:Point>
      </gml:location>
    </ows:Station>
  </ows:measuredAt>
</ows:MeasurementCollection>
```

B.1.3 Gravity Survey

A Gravity Survey is packaged as a `MeasurementArray`, whose members are `Measurements` of gravity at different locations. The locations are defined through `Stations`, a collection of which is collected in a `relatedFeature` of the survey.

```
<?xml version="1.0" encoding="UTF-8"?>
<ows:MeasurementArray gml:id="GravitySurvey2001" xmlns:xmml="http://www.ned.dem.csiro.au/XMML"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://www.opengis.net/ows"
xsi:schemaLocation="http://www.opengis.net/ows ./measurement.xsd" observable="#gravity">
  <gml:description>Small Gravity Survey</gml:description>
  <gml:name>2001 Survey 1</gml:name>
  <gml:boundedBy>
    <gml:Envelope>
      <gml:pos>115.88 -32.20 23.</gml:pos>
      <gml:pos>115.95 -31.95 40.</gml:pos>
    </gml:Envelope>
  </gml:boundedBy>
  <ows:measurementMembers>
    <ows:Measurement gml:id="OG1">
      <gml:description> ... comments - information on processing history, etc ... </gml:description>
      <ows:measuredAt xlink:href="#GravityStation1"/>
      <gml:timeStamp>
        <gml:tInstant>
          <gml:tPosition>2001-09-22</gml:tPosition>
        </gml:tInstant>
      </gml:timeStamp>
      <ows:resultOf>
        <gml:Quantity uom="#g2">9.87001</gml:Quantity>
      </ows:resultOf>
      <ows:measuredBy xlink:href="#gm1"/>
      <ows:relatedFeature xlink:role="previous measurement"
xlink:href="http://www.ned.dem.csiro.au/XMML/gravity/surveys/99#OG32"/>
      <ows:relatedFeature xlink:role="Parent Survey" xlink:href="#GravitySurvey2001"/>
    </ows:Measurement>
    <ows:Measurement gml:id="OG2">
      <gml:description> ... comments - information on processing history, etc ... </gml:description>
      <ows:measuredAt xlink:href="#GravityStation2"/>
      <gml:timeStamp>
        <gml:tInstant>
          <gml:tPosition>2001-09-22</gml:tPosition>
        </gml:tInstant>
      </gml:timeStamp>
      <ows:resultOf>
        <gml:Quantity uom="#g2">9.86678</gml:Quantity>
      </ows:resultOf>
      <ows:measuredBy xlink:href="#gm1"/>
      <ows:relatedFeature xlink:role="Parent Survey" xlink:href="#GravitySurvey2001"/>
    </ows:Measurement>
  </ows:measurementMembers>
  <ows:relatedFeature xlink:role="locations">
    <gml:FeatureCollection gml:id="GravityStations2001">
      <gml:description>List of stations occupied in this survey</gml:description>
      <gml:boundedBy>
        <gml:Envelope>
          <gml:pos>115.88 -32.20 23.</gml:pos>
          <gml:pos>115.95 -31.95 40.</gml:pos>
        </gml:Envelope>
      </gml:boundedBy>
      <gml:featureMember>
        <ows:Station gml:id="GravityStation1">
          <gml:description>road junction</gml:description>
          <gml:location>
            <gml:Point srsName="SRS1" gml:id="point1">
              <gml:coordinates>115.88,-31.95,23.</gml:coordinates>
            </gml:Point>
          </gml:location>
        </ows:Station>
      </gml:featureMember>
      <gml:featureMember>
        <ows:Station gml:id="GravityStation2">
          <gml:description>benchmark outside Post Office</gml:description>
          <gml:location>
            <gml:Point srsName="SRS1" gml:id="point2">
              <gml:coordinates>115.95,-32.20,40.</gml:coordinates>
            </gml:Point>
          </gml:location>
        </ows:Station>
      </gml:featureMember>
    </gml:FeatureCollection>
  </ows:relatedFeature>
  <ows:relatedFeature xlink:role="Reference Station" xlink:href="http://www.agso.gov.au/gravity/stations#ref045"/>
  <gml:dictionaryReference gml:id="gravity" xlink:href="http://www.ned.dem.csiro.au/XMML/observables#gravity"/>
  <gml:dictionaryReference gml:id="gm1" xlink:href="http://www.ned.dem.csiro.au/XMML/gravity/sensors/gm1"/>
  <gml:dictionaryReference gml:id="g2" xlink:href="http://www.ned.dem.csiro.au/XMML/gravity/frames/g2"/>
</ows:MeasurementArray>
```

B.1.4 Assay Samples from a Drillhole

A set of multi-element assays, made on samples taken from a drillhole. The results were **`measuredBy`** a particular laboratory using a specified procedure. The list of **`Specimens`** is collected in a **`relatedFeature`**. The location of each **`Specimen`** is given as an interval within the spatial reference system represented by the drillhole survey.

```
<?xml version="1.0" encoding="UTF-8"?>
<ows:MeasurementArray gml:id="ANA-67403" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:gml="http://www.opengis.net/gml"
xmlns:ows="http://www.opengis.net/ows" xsi:schemaLocation="http://www.opengis.net/ows ./measurement.xsd"
observable="#ChemicalConcentration">
  <gml:description>Composite assays for a set of samples</gml:description>
  <gml:boundedBy>
    <gml:Envelope srsName="#d722">
      <gml:pos>35.2</gml:pos>
      <gml:pos>36.7</gml:pos>
    </gml:Envelope>
  </gml:boundedBy>
  <ows:measurementMembers>
    <ows:Measurement gml:id="SR722410">
      <ows:measuredAt xlink:href="#DR722410"/>
      <gml:timeStamp>
        <gml:tInstant>
          <gml:tPosition>2001-10-12</gml:tPosition>
        </gml:tInstant>
  </gml:timeStamp>
  <ows:resultOf>
    <gml:CompositeValue>
      <gml:valueComponent axis="#As">
        <gml:Quantity uom="#ppm">185</gml:Quantity>
      </gml:valueComponent>
      <gml:valueComponent axis="#Cu">
        <gml:Quantity uom="#percent">0.86</gml:Quantity>
      </gml:valueComponent>
      <gml:valueComponent axis="#Ni">
        <gml:Quantity uom="#ppm">159</gml:Quantity>
      </gml:valueComponent>
    </gml:CompositeValue>
  </ows:resultOf>
</ows:Measurement>
<ows:Measurement gml:id="SR722411">
  <ows:measuredAt xlink:href="#DR722411"/>
  <gml:timeStamp>
    <gml:tInstant>
      <gml:tPosition>2001-10-12</gml:tPosition>
    </gml:tInstant>
  </gml:timeStamp>
  <ows:resultOf>
    <gml:CompositeValue>
      <gml:valueComponent axis="#As">
        <gml:Quantity uom="#ppm">190</gml:Quantity>
      </gml:valueComponent>
      <gml:valueComponent axis="#Cu">
        <gml:Quantity uom="#percent">0.86</gml:Quantity>
      </gml:valueComponent>
      <gml:valueComponent axis="#Ni">
        <gml:Quantity uom="#ppm">159</gml:Quantity>
      </gml:valueComponent>
    </gml:CompositeValue>
  </ows:resultOf>
</ows:Measurement>
<ows:Measurement gml:id="SR722412">
  <ows:measuredAt xlink:href="#DR722412"/>
  <gml:timeStamp>
    <gml:tInstant>
      <gml:tPosition>2001-10-13</gml:tPosition>
    </gml:tInstant>
  </gml:timeStamp>
  <ows:resultOf>
    <gml:CompositeValue>
      <gml:valueComponent axis="#As">
        <gml:Quantity uom="#ppm">17</gml:Quantity>
      </gml:valueComponent>
      <gml:valueComponent axis="#Cu">
        <gml:Quantity uom="#percent">missing</gml:Quantity>
      </gml:valueComponent>
      <gml:valueComponent axis="#Ni">
        <gml:Quantity uom="#ppm">5</gml:Quantity>
      </gml:valueComponent>
    </gml:CompositeValue>
  </ows:resultOf>
</ows:Measurement>
<ows:Measurement gml:id="parameters">
  <ows:measuredAt xlink:href="#null"/>
  <gml:timeStamp>
    <gml:tInstant>
      <gml:tPosition>2001-10-13</gml:tPosition>
    </gml:tInstant>
  </gml:timeStamp>
  <ows:resultOf>
    <gml:CompositeValue>
      <gml:valueComponent axis="#As">
        <gml:Quantity uom="#ppm"/>
      </gml:valueComponent>
```

```
          <gml:valueComponent axis="#Cu">
            <gml:Quantity uom="#percent"/>
          </gml:valueComponent>
          <gml:valueComponent axis="#Ni">
            <gml:Quantity uom="#ppm"/>
          </gml:valueComponent>
        </gml:CompositeValue>
      </ows:resultOf>
    </ows:Measurement>
  </ows:measurementMembers>
  <ows:measuredBy xlink:href="http://my.analytical.lab.com#procedure54"/>
  <ows:relatedFeature xlink:role="Specimens">
    <gml:FeatureCollection gml:id="SpecimenCollectionA">
      <gml:boundedBy>
        <gml:Envelope srsName="#d722">
          <gml:pos>35.2</gml:pos>
          <gml:pos>36.7</gml:pos>
        </gml:Envelope>
      </gml:boundedBy>
      <gml:featureMembers>
        <ows:Specimen gml:id="DR722410">
          <gml:location>
            <gml:Envelope srsName="#d722">
              <gml:pos>35.2</gml:pos>
              <gml:pos>35.7</gml:pos>
            </gml:Envelope>
          </gml:location>
        </ows:Specimen>
        <ows:Specimen gml:id="DR722411">
          <gml:location>
            <gml:Envelope srsName="#d722">
              <gml:pos>35.7</gml:pos>
              <gml:pos>36.2</gml:pos>
            </gml:Envelope>
          </gml:location>
        </ows:Specimen>
        <ows:Specimen gml:id="DR722412">
          <gml:location>
            <gml:Envelope srsName="#d722">
              <gml:pos>36.2</gml:pos>
              <gml:pos>36.7</gml:pos>
            </gml:Envelope>
          </gml:location>
        </ows:Specimen>
      </gml:featureMembers>
    </gml:FeatureCollection>
  </ows:relatedFeature>
  <gml:dictionaryReference gml:id="ChemicalConcentration"
xlink:href="http://www.ned.dem.csiro.au/XMML/observables#ChemicalConcentration"/>
  <gml:dictionaryReference gml:id="ppm" xlink:href="http://www.ned.dem.csiro.au/assay/units#ppm"/>
  <gml:dictionaryReference gml:id="percent" xlink:href="http://www.ned.dem.csiro.au/assay/units#percent"/>
  <gml:dictionaryReference gml:id="As" xlink:href="http://www.ned.dem.csiro.au/assay/elements#As"/>
  <gml:dictionaryReference gml:id="Cu" xlink:href="http://www.ned.dem.csiro.au/assay/elements#Cu"/>
  <gml:dictionaryReference gml:id="Ni" xlink:href="http://www.ned.dem.csiro.au/assay/elements#Ni"/>
  <gml:dictionaryReference gml:id="d722" xlink:href="http://www.ned.dem.csiro.au/drillholes/surveys#d722"/>
</ows:MeasurementArray>
```

B.1.5 Assay Data Presented in Value Lists

An alternative representation of the assay data from the previous example, presented in compact `QuantityLists`. Effectively this is a "column" view of a table of values, compared with the `CompositeValues` in the previous example, which encode tuples or a "row" view of a table.

Note that this example includes the elements `IdentifierList` and `NumberList` from the `xmml` namespace.

```
<?xml version="1.0" encoding="UTF-8"?>
<gml:CompositeValue gml:id="ANA-67403" xmlns:xmml="http://www.ned.dem.csiro.au/XMML"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://www.opengis.net/ows"
xsi:schemaLocation="http://www.ned.dem.csiro.au/XMML ./geoMeasurement.xsd
http://www.opengis.net/ows ../OWS_s/measurement.xsd"  observable="#ChemicalConcentration">
  <gml:valueComponent axis="Sample ID">
    <xmml:IdentifierList>SR722410 SR722411 SR722412</xmml:IdentifierList>
  </gml:valueComponent>
  <gml:valueComponent axis="Sequence Number">
    <xmml:NumberList>1 2 3</xmml:NumberList>
  </gml:valueComponent>
  <gml:valueComponent axis="As">
    <gml:QuantityList uom="#ppm">185 190 17</gml:QuantityList>
  </gml:valueComponent>
  <gml:valueComponent axis="Cu">
    <gml:QuantityList uom="#percent">0.86 0.86 missing</gml:QuantityList>
  </gml:valueComponent>
  <gml:valueComponent axis="Ni">
    <gml:QuantityList uom="#ppm">159 162 -5</gml:QuantityList>
  </gml:valueComponent>
</gml:CompositeValue>
```

Annex C
(informative)

Examples of Schemas for stronger typing

C.1 Bind Typing Parameters to Fixed Values: typedValue.xsd

In this schema we derive new XML types for Observed Values, by binding a fixed value to the observable parameter, and (optionally) extending the type with additional sub-typing axes. New named elements are declared, which may be used in instance documents.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.opengis.net/ows" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://www.opengis.net/ows"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified"
version="0.58">
  <xs:annotation>
    <xs:documentation>
typedValue.xsd

Defines a number of specific Values returned by sensors used in SensorWeb

Specific measurement types are defined by
(i) selecting the appropriate generic Value type (Quantity, Position, Category, OrderedCategory, Count)
(ii) giving it a suitable name
(iii) fixing observable to be a reference to a value from a observableType registry
(iv) adding additional sub-typing axes if required, as named parameters

The observableType registry should be capable of providing a full description including dimensions.

SJDC  2002-04-02
</xs:documentation>
  </xs:annotation>
  <!-- ====================================================================== -->
  <xs:import namespace="http://www.opengis.net/gml" schemaLocation="./value.xsd"/>
  <!-- ====================================================================== -->
  <!-- OrganismCount -->
  <xs:element name="OrganismCount" type="ows:OrganismCountType" substitutionGroup="gml:Count"/>
  <xs:complexType name="OrganismCountBaseType">
    <xs:annotation>
      <xs:documentation>Fix value of "observable"</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:restriction base="gml:CountType">
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#OrganismCount"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="OrganismCountType">
    <xs:annotation>
      <xs:documentation> add "of" to support subtyping - the dictionary chosen must make sense ... </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="ows:OrganismCountBaseType">
        <xs:attribute name="of" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!-- ===================================================================== -->
  <!-- Temperature   -->
  <xs:element name="Temperature" type="ows:TemperatureType" substitutionGroup="gml:Position"/>
  <xs:element name="TemperatureList" type="ows:TemperatureListType" substitutionGroup="gml:PositionList"/>
  <xs:complexType name="TemperatureBaseType">
    <xs:annotation>
      <xs:documentation>fix the value of observable</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:restriction base="gml:PositionType">
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#Temperature"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="TemperatureType">
    <xs:annotation>
      <xs:documentation>add "of" and "in" to support subtyping - the dictionary used must make sense ...
</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="ows:TemperatureBaseType">
        <xs:attribute name="of" type="xs:anyURI" use="optional"/>
        <xs:attribute name="in" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="TemperatureListBaseType">
    <xs:annotation>
      <xs:documentation>fix the value of observable</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:restriction base="gml:PositionListType">
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#Temperature"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="TemperatureListType">
    <xs:annotation>
      <xs:documentation>add "of" and "in" to support subtyping - the dictionary used must make sense ...
</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="ows:TemperatureListBaseType">
        <xs:attribute name="of" type="xs:anyURI" use="optional"/>
        <xs:attribute name="in" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!-- ===================================================================== -->
  <!-- Gravity   -->
  <xs:element name="Gravity" type="ows:GravityType" substitutionGroup="gml:Position"/>
  <xs:complexType name="GravityType">
    <xs:annotation>
      <xs:documentation>Restricts PositionType
                        * fixes the value of observable</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:restriction base="gml:PositionType">
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#Gravity"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
  <!-- ===================================================================== -->
  <!-- Chemistry -->
  <xs:element name="ChemicalConcentration" type="ows:ChemicalConcentrationType"
substitutionGroup="gml:Quantity"/>
```

```
  <xs:element name="DissolvedChemical" type="ows:ChemicalConcentrationType" substitutionGroup="gml:Quantity"/>
  <xs:complexType name="ChemicalConcentrationBaseType">
    <xs:annotation>
      <xs:documentation>fixes the value of observable </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:restriction base="gml:QuantityType">
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#ChemicalConcentraion"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="ChemicalConcentrationType">
    <xs:annotation>
      <xs:documentation>adds "of" to support subtyping by chemcialSpecies, "in" to indicate medium</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="ows:ChemicalConcentrationBaseType">
        <xs:attribute name="of" type="xs:anyURI" use="optional"/>
        <xs:attribute name="in" type="xs:anyURI" use="optional"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <!-- ====================================================================== -->
  <!-- Speed -->
  <xs:element name="Speed" type="ows:SpeedType" substitutionGroup="gml:Quantity"/>
  <xs:complexType name="SpeedType">
    <xs:annotation>
      <xs:documentation>Restricts QuantityType
                                    * fixes the value of observable </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:restriction base="gml:QuantityType">
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#Speed"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
  <!-- ====================================================================== -->
  <!-- Direction -->
  <xs:element name="Direction" type="ows:DirectionType" substitutionGroup="gml:Position"/>
  <xs:complexType name="DirectionType">
    <xs:annotation>
      <xs:documentation>Restricts PositionType
                                    * fixes the value of observable </xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:restriction base="gml:PositionType">
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#Direction"/>
      </xs:restriction>
    </xs:simpleContent>
  </xs:complexType>
  <!-- ====================================================================== -->
  <!-- Velocity -->
  <xs:element name="Velocity" type="ows:VelocityType" substitutionGroup="gml:_Value"/>
  <xs:complexType name="VelocityBaseType">
    <xs:annotation>
      <xs:documentation>Restricts CompositeValueType
                                    * restricts the component types
                                    * fixes the value of observable </xs:documentation>
    </xs:annotation>
    <xs:complexContent>
      <xs:restriction base="gml:CompositeValueType">
        <xs:choice>
          <xs:sequence>
            <xs:element ref="gml:_metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="magnitude" type="ows:SpeedPropertyType"/>
            <xs:element name="azimuth" type="ows:DirectionPropertyType"/>
            <xs:element name="inclination" type="ows:DirectionPropertyType" minOccurs="0"/>
          </xs:sequence>
          <xs:sequence>
            <xs:element ref="gml:_metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="velocityComponent" type="ows:SpeedPropertyType" minOccurs="2" maxOccurs="3"/>
          </xs:sequence>
        </xs:choice>
        <xs:attribute name="observable" type="xs:anyURI" use="optional"
fixed="http://www.opengis.net/ows/observableTypes#Velocity"/>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="VelocityType">
    <xs:complexContent>
      <xs:extension base="ows:VelocityBaseType">
        <xs:attribute name="of" type="xs:anyURI"/>
      </xs:extension>
    </xs:complexContent>
  </xs:complexType>
  <!-- ====================================================================== -->
  <!--
      Some other scalar types which we have not yet analysed
       and whose observable has not yet been constrained
  -->
  <xs:element name="Conductivity" type="gml:QuantityType" substitutionGroup="gml:Quantity"/>
  <xs:element name="Resistivity" type="gml:QuantityType" substitutionGroup="gml:Quantity"/>
  <xs:element name="DissolvedSolids" type="gml:QuantityType" substitutionGroup="gml:Quantity"/>
  <xs:element name="Acidity" type="gml:QuantityType" substitutionGroup="gml:Quantity"/>
  <xs:element name="Pressure" type="gml:QuantityType" substitutionGroup="gml:Quantity"/>
  <xs:element name="ElectricPotential" type="gml:PositionType" substitutionGroup="gml:Position"/>
  <xs:element name="Current" type="gml:QuantityType" substitutionGroup="gml:Quantity"/>
  <xs:element name="Density" type="gml:QuantityType" substitutionGroup="gml:Quantity"/>
  <xs:element name="Turbidity" type="gml:QuantityType" substitutionGroup="gml:Quantity"/>
  <xs:element name="Rainfall" type="gml:QuantityType" substitutionGroup="gml:Quantity"/>
  <xs:element name="SolarRadiation" type="gml:QuantityType" substitutionGroup="gml:Quantity"/>
  <!-- ====================================================================== -->
  <!-- some specific measurement property types -->
  <xs:complexType name="SpeedPropertyType">
    <xs:complexContent>
      <xs:restriction base="gml:ValueComponentType">
        <xs:sequence>
          <xs:element ref="ows:Speed"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
  <xs:complexType name="DirectionPropertyType">
    <xs:complexContent>
      <xs:restriction base="gml:ValueComponentType">
        <xs:sequence>
          <xs:element ref="ows:Direction"/>
        </xs:sequence>
      </xs:restriction>
    </xs:complexContent>
  </xs:complexType>
  <!-- ====================================================================== -->
</xs:schema>
```

Bibliography

[FOW1998] Martin Fowler, Analysis Patterns: reusable object models. Addison Wesley Longman, Menlo Park, CA. 1998.

[GML] Geography Markup Language (GML) 2.0, OGC Document Number: 01-029 http://www.opengis.net/gml/01-029/GML2.html

[ISO19103] ISO TC 211Geographic Information – Conceptual Schema Language ISO 19103 http://www.isotc211.org/pow.htm

[ISO19107] ISO TC 211Geographic Information – Spatial Schema ISO 19107 http://www.isotc211.org/pow.htm

[ISO19110] ISO TC 211Geographic Information – Feature cataloguing methodology ISO 19110 http://www.isotc211.org/pow.htm

[ISO19123] ISO TC 211 Geographic Information – Schema for coverage geometry and functions ISO 19123 http://www.isotc211.org/pow.htm

[OGCAS5] The OpenGIS Abstract Specification. Topic 5: Features. OGC Document 99-105r2. http://www.opengis.org/public/abstract/99-105r2.pdf

[SCS] Tom McCarty (ed), Sensor Collection Service IPR, OGC 02-028. http://ip.opengis.org/cgi-bin/ip/ows1view.htm

[SensorML] Mike Botts (ed), Sensor Model Language IPR, OGC 02-026. http://ip.opengis.org/cgi-bin/ip/ows1view.htm

[SUP1963] Suppes, P. & Zinnes, J.L. Basic measurement theory. P 3-76 In R. D. Luce, R. R. Bush, & E. H. Galanter (Eds.), Handbook of Mathematical Psychology, Vol. 1. New York: Wiley 1963.

[UOM2001] John Bobbitt (ed), Units of Measure and Quantity Datatypes. OGC Project Document 01-044r3 http://feature.opengis.org/members/archive/arch01/01-044r3.pdf

[VIM] International Vocabulary of Basic and General Terms in Metrology. BIPM/ISO 1993.

[XPointer] XML Pointer Language (XPointer) Version 1.0. World Wide Web Consortium. 2001. http://www.w3.org/TR/xptr/

[YOD] Yoder, J. W., Balaguer, F. and Johnson, R. From analysis to design of the observation pattern. http://www.joeyoder.com/Research/metadata/Observation/ObservationModel.pdf