Open Geospatial Consortium Inc.

Date: 2005-10-05

Reference number of this document: **OGC 05-086**

Version: 1.0

Category: OpenGIS® Best Practices Paper

Editor: Mike Botts
University of Alabama in Huntsville

OpenGIS® Sensor Model Language (SensorML) Implementation Specification

Warning

This document is not an OGC Standard. It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard.

Recipients of this document are invited to submit, with their comments, notification of any relevant patent rights of which they are aware and to provide supporting documentation.

Document type: OpenGIS® Best Practices Paper
Document subtype:
Document stage: Approved
Document language: English

Table of Contents

i. Preface

This draft specification is one of five engineering specifications produced under OGC's Sensor Web Enablement (SWE) activity, which is being executed under OGC's Interoperability Program. The initial version was produced during OGC Web Services (OWS) 1.1 Initiative, conducted in 2001. The previous version was produced under the OGC Web Services (OWS) 1.2 Initiative, conducted June 2002 - January 2003. This version represents efforts that have been continued since these OWS 1.2 initiatives and during the OWS 3 Initiative, conducted March 2005 – October 2005. This document provides version 1.0 of the SensorML core specification.

Suggested additions, changes, and comments on this draft report are welcome and encouraged. Such suggestions may be submitted by email message or by making suggested changes in an edited copy of this document.

The changes made in this document version, relative to the previous version, are tracked by Microsoft Word, and can be viewed if desired. If you choose to submit suggested changes by editing this document, please first accept all the current changes, and then make your suggested changes with change tracking on.

ii. Document terms and definitions

This document uses the specification terms defined in Subclause 5.3 of [OGC 05-008], which is based on the ISO/IEC Directives, Part 2. Rules for the structure and drafting of International Standards. In particular, the word "shall" (not "must") is the verb form used to indicate a requirement to be strictly followed to conform to this specification

iii. Submitting Organizations

The following organizations submitted this document to the Open Geospatial Consortium, Inc:

a. University of Alabama in Huntsville
b. Iris Corporation
c. Distributed Instruments
d. CSIRO Australia
e. Galdos Systems, Inc
f. Image Matters, LLC

iv. Submission Contact Point

All questions regarding this document should be directed to the editor or the contributors:

CONTACT	COMPANY	ADDRESS	PHONE/FAX	EMAIL
Mike Botts (Editor)	University of Alabama in Huntsville	ESSC / NSSTC Huntsville, AL 35899	+01-256-961-7760	mike.botts@uah.edu
Alexandre Robin	University of Alabama in Huntsville	ESSC / NSSTC Huntsville, AL 35899	+01-256-961-7978	robin@nsstc.uah.edu
Steve Havens	IRIS Corporation	4220 Varsity Drive Suite E Ann Arbor, MI 48108	+01-734-677-3604	shavens@iriscorp.org
Simon Cox	CSIRO, Australia	PO Box 1130, Bentley WA 6102 Australia	+61-8-6436-8639	simon.cox@csiro.au
Jeff Ricker	Distributed Instruments	33 West Shore Drive Putnam Valley, NY 10579	+01-845-526-4841	ricker@distributedinstruments.com
Ron Lake	Galdos Systems, Inc.	Suite 200, 115 West Pender Street, Vancouver, B.C. V6E 2P4	+01-604-484-2750	rlake@galdosinc.com
Harry Niedzwiadek	Image Matters, LLC	214 South King St. Leesburg VA 20175 USA	+01-703-669-5510	harryn@imagem.cc

v. Revision History

Date	Release	Author	Section modified	Description
2001-07-16	001_001	meb		Original pre-OGC SensorML document
2002-04-04	v0.04	meb		First “complete” DIPR version.
2002-04-19	02-026 (v0.0.4c)	meb,han	throughout	Final OWS 1.1 review and edit; minor changes. Produced OGC 02-026. Approved as Public Discussion Paper
2003-03-25	03-005r1b (v0.8)	meb, han	throughout	Final OWS 1.2 review and edit. Produced OGC 03-005r1.
2004-11-01	04-019r5	meb	throughout	Extended much of the metadata, moved sensor models and response models to process models; split core SensorML from SensorML extensions. Extensions such as SensorModels and RadiationMOdel for remote sensors have been moved to OGC document 04-068 Approved as Recommendation Paper
2005-10-10	05-086	meb	throughout	Moved to concept of all SensorML components being modelled as process models rather than components that contain process models; “components” include transducers, sensors, process chains, and systems; updated UML,

				encodings, and documentation to reflect this; Removed basic parameters, data encodings, and position data from SensorML (sml) namespace and moved it to SWE Common (swe) namespace for easier use by SWE Submitted as Implementation Specification

Any issues in this specification are captured in the following format:

> **Issue Name**: [Issue Name goes here. (Your Initials, Date)]
>
> **Issue Description:** [Issue Description.]
>
> **Resolution:** [Insert Resolution Details and History.] (Your Initials, Date)]

vi. Recommended Changes to the OpenGIS Abstract Specification

The OpenGeospatial© Abstract Specification does not require changes to accommodate the technical contents of this document.

Foreword

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

This specification was developed under the OWS 1.1, OWS 1.2, and OWS 3 initiatives.

Additional information is available at http://vast.uah.edu/SensorML.

In addition, one can subscribe to the SensorML forum and listserver at:

http://mail.opengeospatial.org/mailman/listinfo/sensorML.

This document includes three annexes; Annexes A and B are normative, and Annex C is informative.

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

Introduction

The OpenGeospatial Consortium (OGC) Sensor Web Enablement (SWE) activity, is being executed through the OGC Web Services (OWS) initiatives (under the Interoperability Program). The SWE initiatives are establishing the interfaces and protocols that will enable a "Sensor Web" through which applications and services will be able to access sensors of all types over the Web. These initiatives have defined, prototyped and tested several foundational components needed for Sensor Web Enablement, namely:

1. **Sensor Model Language (SensorML)** – The general models and XML encodings for sensors. SensorML originated under OWS 1.1, was significantly enhanced under OWS 1.2 and is now available as a public discussion paper.
2. **Observations & Measurements (O&M)** - The general models and XML encodings for sensor observations and measurements. O&M originated under OWS 1.1 and was significantly enhanced under OWS 1.2.
3. **Sensor Observation Service (SOS)** – A service by which a client can obtain observations from one or more sensors/platforms (can be of mixed sensor/platform types). Clients can also obtain information that describes the associated sensors and platforms. This service originated under OWS 1.1 and was significantly enhanced under OWS 1.2.
4. **Sensor Planning Service (SPS)** – A service by which a client can determine collection feasibility for a desired set of collection requests for one or more mobile sensors/platforms, or a client may submit collection requests directly to these sensors/platforms. This service was defined under OWS 1.2.
5. **Web Notification Service (WNS)** – A service by which a client may conduct asynchronous dialogues (message interchanges) with one or more other services. This service is useful when many collaborating services are required to satisfy a client request, and/or when significant delays are involved is satisfying the request. This service was defined under OWS 1.2 in support of SPS operations. WNS has broad applicability in many such multi-service applications.

In addition, the following encodings and services are undergoing development as additional components to the SWE Framework:

1. **Sensor Alert Service (SAS)** – A service for advertising, subscribing to, and publishing alerts to alert listener clients. This service is being defined under the Sensor Alert Service Interoperability Experiment.
2. **TransducerML (TML)** – A model and encoding for streaming multiplexed data from a sensor system, and for describing the system and

data encoding. Developed initially by the Iris Corporation, this service is being brought into the OGC SWE framework and undergoing further development as part of the OWS 3 Initiative.

This document specifies SensorML. The other components are specified under separate documents. While SensorML serves as a component within the OGC Sensor Web Enablement framework, SensorML does not depend upon the presence of the other SWE components. It is envisioned that SensorML will be utilized both as part of, and independent of, the OGC Sensor Web Enablement framework.

Sensor Model Language: An Implementation Specification

1 Scope

This document specifies *SensorML*. *SensorML provides a model and an XML schema for defining the geometric, dynamic, and observational characteristics of sensors and sensor systems.* There are a great variety of sensor types from simple visual thermometers to complex electron microscopes and earth observing satellites.

Within the context of SensorML, sensors and sensor systems are modeled as processes. Processes are entities that take one or more inputs and through the application of well-defined methods using specific parameters, results in one or more outputs. The process model defined in SensorML can be used to describe, or serialize, a wide variety of processes, including actuators, spatial transforms, and data processes, to name a few. SensorML also supports linking between processes and thus supports the concept of process chains, which are themselves defined as processes.

With regard to Observations and Measurements, we assume three fundamental components of information:

1. There are properties of physical entities and phenomena that are capable of being measured and quantified. Within the SWE context, properties that are capable of being measured are considered as “Phenomenon”. Each of these can be referenced in an Phenonmenon dictionary. Phenomenon definitions might include, for example, properties such as temperature, count, rock type, chemical concentration, or radiation emissivity.
2. There are sensors that are capable of observing and measuring particular properties. Either by design or as a result of operational conditions, these sensors have particular response characteristics that can be used to determine the values of the measurements, as well as assess the quality of these measurements. In addition to the response characteristics, these sensor systems have properties of location and orientation that allow one to associate the measured values with a particular geospatial location at a particular time. The role of the *SensorML* is to provide characteristics required for processing, georegistering, and assessing the quality of measurements from sensor systems.

   Within this context, a SensorML documents can serve two possible roles. The first is to describe the procedure by which an existing observation was obtained. This would include the sensor measurement process, as well as any post processing of the raw observations. The second possible role is to provide processing chains with which SensorML-enabled software could derive new data from existing observations on-demand. One might consider this as a Derivable Observation, since the values don’t exist prior to execution of the processing chain.
3. Finally, there are data values that are returned by a sensor system or are derived from sensor measurements. These measurements may be accessed directly from

the sensor, or from data stores that distribute and possibly process these data into various products. The processing and georegistration of these measured values require knowledge of the properties of the sensor system. Within the context of the OGC Sensor Web Enablement framework, values returned by sensors can be provided within the Observations and Measurements schemas or other data provider types.

All three of these components are linked within the Sensor Web Enablement concepts. While these links will be discussed within this document, only the second component is within the scope of this document.

Thus the purpose of *SensorML* is to:

- provide general sensor information in support of data discovery
- support the processing and analysis of the sensor measurements
- support the geolocation of observed values (measured data)
- provide performance characteristics (e.g. accuracy, threshold, etc.)
- provide an explicit description of the process by which an observation was obtained (i.e. it's lineage)
- provide an executable process chain for deriving new data products on demand (i.e. derivable observation)
- archive fundamental properties and assumptions regarding sensor

SensorML can, but generally does not, provide a detailed description of the hardware design of a sensor. Rather it is a general schema for describing functional models of the sensor. The schema is designed such that it can be used to support the processing and geolocation of data from virtually any sensor, whether mobile or dynamic, in-situ or remotely sensed, or active or passive. This allows one to develop general, yet robust, software that can process and geolocate data from a wide variety of sensors ranging from simple to complex sensor systems.

SensorML supports both rigorous geolocation models and mathematical geolocation models. A *rigorous sensor model* is defined here as one that describes the geometry and dynamics of the instrument and provides the ability to utilize this information along with position and orientation of the platform in order to derive geolocation of the sensor data. *Mathematical sensor models* are typically derived using a rigorous model, perhaps augmented by human interaction. These general mathematical models typically hide the characteristics of the sensor and allow for geolocation of sensor data through the use of polynomial functions. Different mathematical models can be designed to define sample location within a variety of coordinate systems, including the local sensor frame, the local frame for the associated platform, or a geographic coordinate reference frame.

Within SensorML, one may chose to model a sensor platform as a System with its own Coordinate Reference System to which on-board sensor positions can be referenced. One may also chose to relate positions between various sensors while ignoring the platform reference frame, by defining any sensor position relative to an onboard GPS sensor and

an orientation (gimbal) sensor. *Either way, for the case of rigorous sensor models, we allow one to separate the description of the sensor from that of its platform. Common platforms include: ground stations, automobiles, aircraft, earth-orbiting satellites, ocean buoys, ships, and people. A deployed sensor is mounted on a static or dynamic platform (or an assembly of nested platforms).*

The detailed engineering description of the mechanical structure of the platform(s) is outside the scope of the sensor description. From our perspective, the main aspect of the platform is the applicable (possibly dynamic) coordinate system(s), which allows transformation of sensor measurements to some relevant geographic coordinate reference frame. Thus, for a rigorous model, it is only through the association of a sensor with its platform(s) that measured values can be georegistered.

2 Conformance

Conformance and Interoperability Testing for SensorML may be checked using all the relevant tests that pertain to 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 document. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this document 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.

OpenGIS® *Abstract Specification*, OGC document 99-100r1. Topic 0: Overview Topic 2 - Spatial Reference System, Version 4 Topic 7 – The Earth Imagery Case, Version 4 Topic 11 - Metadata (Same as ISO Metadata document 19115) Topic 12 - OGC Services Architecture, Version 4.1
Geographic Markup Language (GML) Implementation Specification, Version 3.1.1, 19 April 2004, OGC document 03-105r1.
Namespaces in XML. W3C Recommendation (14 January 1999). Available [Online]: <http://www.w3.org/TR/1999/REC-xml-names-19990114/>
XML Schema Part 1: Structures. W3C Recommendation (2 May 2001). Available [Online]: <http://www.w3.org/TR/xmlschema-1/>

*XML Schema Part 2: Datatype*s. W3C Recommendation (2 May 2001). Available [Online]: <http://www.w3.org/TR/xmlschema-2/>

4 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

4.1 Coordinate Reference System (CRS)

A spatial or temporal framework within which a position and/or time can be defined.

4.2 Location

The linear relationship of a point in space to some reference frame.

4.3 Location Model

A model that allows one to locate objects within one reference frame (localFrame) relative to another reference frame (referenceFrame).

4.4 Measurand

A particular property or phenomenon subject to measurement by the sensor.

4.5 Measurement

An instance of a procedure to estimate of the value of a natural phenomenon, typically involving an instrument or sensor. 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.

4.6 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 subjective assignment or some other method of estimation or assignment.

4.7 Orientation

The angular relationship of one spatial reference frame to another.

4.8 Phenomenon

An event or physical property that can be observed and measured, such as temperature, gravity, chemical concentration, orientation, number-of-individuals.

4.9 Position
The location and orientation of one reference frame to another.

4.10 Process Model
A process that takes one or more inputs, and based on parameters and methodologies, generates one or more outputs.

4.11 Process (Process Chain)
A composite process that takes one or more inputs, and based several internal processes with linkage between them, generates one or more outputs.

4.12 Reference Frame
A coordinate system by which the position (location and orientation) of an object can be referenced.

4.13 Response Model
A type of Process Model, typically associated with a Sensor that takes one or more Samples as input, and outputs one or more new Products, based on characteristic response characteristics.

4.14 Sample
A subset of the physical entity on which an observation is made.

4.15 Sensor
An entity capable of observing a phenomenon and returning an observed value. A sensor can be an instrument or a living organism (e.g. a person), but herein we concern ourselves primarily with modelling instruments, not people.

4.16 Sensor Model
In line with traditional definitions of the remote sensing community, a sensor model is a type of Location Model that allows one to georegister observations from a sensor (particularly remote sensors).

4.17 (Sensor) Platform
An entity to which can be attached sensors or other platforms. A platform has an associated local coordinate frame that can be referenced to an external coordinate reference frame and to which the frames of attached sensors and platforms can be referenced.

4.18 Transducer
An entity that receives a signal as input and outputs a modified signal as output. Includes detectors, actuators, and filters.

5 Conventions

5.1 Symbols (and abbreviated terms)

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

API	Application Program Interface
CEOS	Committee for Earth Observation Sensors
COTS	Commercial Off The Shelf
GML	Geographic Markup Language
gml:*	schema namespace for GML
ISO	International Organization for Standardization
JPL	Jet Propulsion Laboratory
ODM	Observation Dynamics Model
OGC	Open Geospatial Consortium
OWS	OGC Web Services
om:*	schema namespace for Observations and Measurement
O&M	Observations and Measurements
UML	Unified Modeling Language
SCS	Sensor Collection Service
SensorML	Sensor Model Language
sml:*	schema namespace for SensorML
SPS	Sensor Planning Service
SWE	Sensor Web Enablement
swe:*	schame namespace for SWE Common schema
WGS84	World Geodetic System 84
WNS	Web Notification Service
XML	eXtended Markup Language
xs:*	schema namespace for XMLSchema.2002
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.

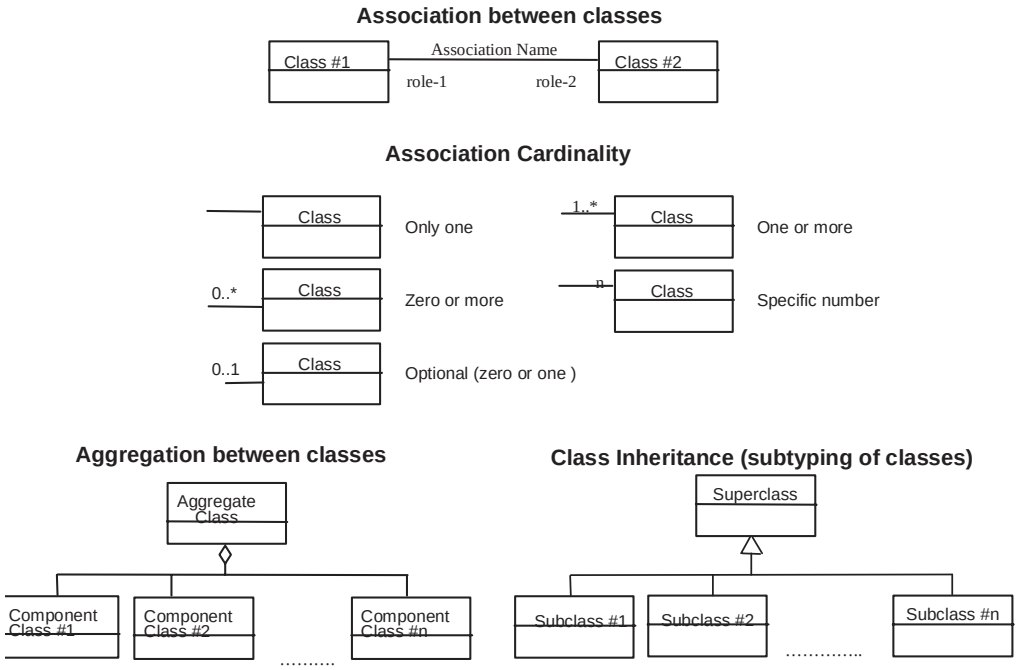


Figure 1.1 — UML notation

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 Background

6.1 Motivation

The importance of long-term monitoring of the Earth's environment and the development of improved data processing techniques, has raised awareness of the need for preserving low-level sensor data and the information required for reprocessing this data. Unfortunately, such information is often lost or difficult to find five to ten years after completion of a sensor's original mission life. The proposed SensorML is one step toward preserving part of the vital information required for geolocation and processing of sensor data for both real-time and archival observations.

Often one is unaware of private or public sensor systems that are available for a particular application. Particularly in disaster prevention or remediation, it is vital that one be able to discovery and gather observations of relevance from any appropriate sensors in the affected area. SensorML provides a standard means by which sensor and platform capabilities and properties can be published and discovered. As will be discussed in more detail below, SensorML also provides information that allows for geolocation and processing of these sensor observations without a priori knowledge of the sensor's properties.

Web-enabled sensors provide the technology to achieve rapid access to accurate environmental information from the field. Streaming sensor information in standard formats facilitates integration, analysis, and creation of various data "views" that are more meaningful to the end user and avoids the drawbacks of locating and accessing incompatible archived data. This provides a significant advantage in that it reduces the time lag between making measurements and applying those measurements in decision-making. Time savings are particularly noticeable in the management of time critical events such as emergency response, advanced warning systems, and forecasting. A second benefit is in the routine use of data for everyday decision-making. Together, these developments will advance the realization of an integrated, yet distributed, monitoring and assessment system used by government, researchers, businesses, and the public in improving decision making based on high quality, near real time data and information.

Furthermore, recent research and development activities have demonstrated several significant benefits of providing on-demand sensor geolocation within desktop or on-board tools. These include:

(1) Significant reduction of distributed data from remote sensors; large data volumes resulting from the distribution and storage of per-pixel latitudes and longitudes, as well as other pre-processed geometric relationships can be replaced with the calculation of these values on-demand;

(2) Improved capabilities for visually integrating and analytically comparing multi-sensor data;

(3) The ability to more easily correct geolocation errors from within the end-user tools and to redistribute these corrections to the user community;

(4) The ability to take advantage of several adaptive methods in computer graphics for improving interactivity within visualization tools; and

(5) Greatly improved capabilities for search and query of spatial-temporal sensor data without the need to request and perhaps store large data sets.

Traditionally, the geolocation of low-level sensor data has required writing or utilizing software specifically designed for that sensor system. The availability of a standard model language for describing platform position and rotation, as well as instrument geometry and dynamics, allows for the development of generic multi-purpose software that can provide geolocation for potentially all remotely sensed data. The availability of such software, herein referred to as an Observation Dynamics Model (ODM), in turn provides a simple, single Application Programming Interface (API) for tool developers to incorporate sensor geolocation and processing into their application software.

One intent of a standard SensorML is to allow the development of software libraries that can parse these files and calculate required look angles and timing for each sensor pixel. Other efforts are establishing standards for storage and transmission of sensor platform location and rotation in order to insure that such formats are also maintained, available, and readable by similar APIs *[CCSDS, 1999]*.

6.2 Importance to archival needs

A standard description format for sensors is important for the long-term definition of the sensor model's fundamental characteristics and assumptions for use in future reprocessing and refining of sensor data.

We are currently entering an era of Earth observation in which we have realized the importance of long-term observation of the Earth's environment. Thus, archiving the raw or low-level sensor data for future reprocessing has taken on greater importance. Equally important is that we preserve the characteristic metadata and assumptions required to reprocess the sensor data. The characteristic data include what is needed for geolocation, calibration, and radiometric processing of the remotely sensed data. Simply archiving the latitude and longitude values will not only be expensive, but will also prove to be highly inadequate. It is anticipated that further efforts within organizations such as the CEOS Data Subgroup, ISO TC211, and the OpenGeospatial Consortium, will be directed toward insuring proper standardization and archiving of other required data, such as platform position and rotation, and target or planet models. This current paper is directed specifically at the adequate description and standardization of fundamental geometric, dynamics, and measurement characteristics of the sensor.

As an example, sensor look angles have traditionally been either pre-calculated and stored within a data array structure, or are calculated as needed within software systems developed specifically for that sensor. With time, unfortunately, the actual parameter values for the geometric and dynamic characteristics of the sensor are often lost as contract reports and software become obscure, and as look angle arrays and hardwired software prove difficult to deconvolve into the characteristic sensor parameters. Once the

initial mission has been completed and the processing teams dispersed, reprocessing, correction, or refinement of the sensor data thus become very difficult, if not impossible. For example, this was the case for reprocessing of the archived Optical Line Scanner (OLS) data (Ken Knowles/University of Colorado, personal communication), as well as for the 20 year-old data from the Viking Mission to Mars (Bill Taber/JPL, personal communication).

6.3 Importance to software support

The standardization of a Sensor Model Language (SensorML) and the availability of SensorML documents for all Earth observing sensors will allow for significant opportunities for software systems to support the processing, analysis, and visual fusion of multiple sensors.

Traditionally software that supported multiple sensors has been forced to deal with proprietary software designed for each individual sensor. When such software systems still exist and can be located, the software developer is often faced with trying to merge incompatible software architectures and development languages, or with rewriting the software to meet the requirements of his or her software. Even then, the addition of each new sensor system that the developer wishes to support requires the development of individual software modules specific to that sensor, often resulting in redundant code for manipulating and transforming the data.

In contrast, SensorML is based on the concept that the availability of standard sensor system descriptions allows for the development of general processing software capable of geolocating and transforming any sensor data for which such a description exists. This concept is built around the availability of description files for providing sensor-system specific information regarding platform position and rotation, instrument geometry and dynamics, target planet shape and position, and perhaps other time-tagged information, such as data dropouts, instrument modes of operation, or spacecraft clock adjustments.

This concept is dependent on the availability of generic software for parsing these files, and calculating the transformations required to geolocate, and execute processing of the sensor data. A significant advantage of the this concept for the developer and ultimately the end user, is that it provides a single source, single application programming interface, for the geolocation and processing of any sensor system, rather than requiring the developer to locate and implement proprietary software for each sensor system.

In addition, this concept allows for the development of tools that can provide on-demand geolocation and mapping. As SensorML-enabled application software becomes more common, there will be less need to store and distribute volumetrically costly latitude, longitude, altitude, and incident angles values per pixel. Furthermore, correction of sensor geolocation requires only the redistribution of much smaller description files, rather than the redistribution of large collections of reprocessed sensor data. In fact, correction or refinement of geolocation can be conducted by the end user as necessary, rather than relying strictly on the instrument team. Finally, the ability to provide spatial-temporal knowledge of the sensor's coverage, independent of the on-line presence of sensor data, allows much needed search and query capabilities for determining sensor

coverage for given a location or time, or for determining coincident sampling between two or more sensors.

In addition to the significant benefits discussed above, on-demand processing of geolocation has been shown to be as fast or faster than reading in and processing pre-calculated location values. With CPU power increasing faster than I/O rates, the improvement in the calculation of geolocation information will increase even further in the future, with the added benefit of not wasting valuable RAM capacity with storage of blocks of latitude and longitude values.

6.4 Importance to Sensor Web Enablement

The SWE architecture and design provides an important contribution for the enablement of future sensor webs. However, it is important to note that while SensorML is a key component to the SWE initiative, SensorML is not dependent on the SWE framework and can capable be used on its own or in conjunction with other sensor system architectures.

In much the same way that HTML and HTTP standards enabled the exchange of any type of information on the World Wide Web, the OGC Sensor Web Enablement (SWE) initiative is focused on developing standards to enable the discovery and exchange of sensor observations, as well as the tasking of sensor systems. The functionality that has been targeted within a sensor web includes:

- Discovery of sensors and sensor observations that meet our needs
- Determination of a sensor's capabilities and quality of measurements
- Access to sensor parameters and processes that automatically allow software to process and geolocate observations
- Retrieval of real-time or time-series observations and coverages in standard encodings
- Tasking of sensors to acquire observations of interest
- Subscription to and publishing of alerts to be issued by sensors or sensor services based upon certain criteria

SensorML is a key component for enabling autonomous and intelligent sensor webs. SensorML provides the information needed for discovery of sensors, including the sensor's capabilities, location, and taskability. It also provides the means by which real-time observations can be geolocated and processed "on-the-fly" by SensorML-aware software. SensorML describes the interface and taskable parameters by which sensor tasking services can be enabled, and allows information about the sensor to accompany alerts that are published by sensor systems. Finally, intelligent sensors can utilize SensorML descriptions during on-board processing to process and determine the location of its observations.

7 History

The concepts proposed in this paper have been successfully tested and implemented in limited studies, and have been shown to provide significant benefits to both the developer and user of sensor data systems.

The SensorML was originally conceived as a sensor description language that would aid in the processing and geolocation of multiple sensors. Being driven by the remote sensing community, the original focus was on providing properties to assist in the geolocation of individual pixels within scanners and cameras.

The concept for the standardized description files for all aspects of sensor geolocation was first proposed by the planetary science community and was implemented in the SPICE software system, that was developed and maintained by the Navigation and Ancillary Information Facility (NAIF) at NASA JPL. Written in FORTRAN and utilized by the planetary science community for nearly every mission since Galileo, SPICE has proved beneficial by reducing redundant programming for each mission, and by providing additional benefits not previous experienced before the implementation of SPICE.

In 1993, the University of Alabama in Huntsville (UAH) in cooperation with NASA JPL, implemented and tested the SPICE concepts for application within the Earth observation community. Termed the NASA EOS Interuse Experiment, this research effort focused on improving the integration of multiple sensor data by avoiding the need to distribute data as incompatible map projection grids. The results of the Interuse Experiment proved that the ODM concept for Earth observation sensors was well within our abilities, and provided perhaps even more significant benefits to the Earth observation than had been experienced within the planetary community.

With subsequent funding, the UAH VisAnalysis System Technology (VAST) team has been implementing a more lightweight ODM directed principally for the Earth observation community and developed in Java.

In April 1998, the Global Mapping Task Team (GMTT) within Committee for Earth Observation Satellites (CEOS) released the following recommendation:

> **April 1998 - Recommendation to CEOS: Interoperability of Multiple Space Agency Sensor Data from the Global Mapping Task Team – Bernried, Germany**
>
> **Definition of Problem:** There is an increasing realization by Earth observation scientists that data from space-borne sensors are not adequately nor easily georeferenced to meet their requirements. The consequence of this is that it is currently extremely difficult or impossible to combine data from different space-borne sensors or ground-based data. The first impediment is the lack of adequate, publicly available data on the spatial-temporal extents of data from space-borne sensors.
>
> **Recommendation:** We, the CEOS Global Mapping Task Team, recommend that the space agencies seriously consider the production, storage, public access, and interoperability of adequate data for describing the dynamics and geometry of the sensor system. These data might include satellite position (ephemeris), satellite rotations (attitude), sensor model (dynamics, geometry, and calibration), relevant planet models, and spacecraft clock model. This data

> should be made available in real-time. There should also be an effort to provide publicly available software to ingest the above data using a common API. Any recommendations for the most appropriate propagation model should be adequate documented or algorithms provided.

In September, 1998, Mike Botts introduced the beginnings of a "Sensor Description Format" to the CEOS GMTT and received recommendations to consider XML as a description framework. In September 1999, the initial XML-based version of the SensorML was introduced to the CEOS GMTT by Dr. Botts. In March 2000, he was awarded a contract through the NASA AIST program to complete, implement, and test the SensorML (funding was actually received in December 2000). Initial focus of the SensorML was on the geolocation and description of remote sensing instruments, with minor attention given to measurement characteristics such as radiometry [Botts, 2001].

In Fall 2000, Liping Di (a CEOS GMTT member) proposed to the ISO TC211 Committee to establish a standard sensor and data model framework. This was approved in December 2000 and Dr. Botts was asked to serve as a team member. The first meeting was scheduled for June 2001. It is expected that the SensorML will both influence and be influenced by the ISO activities. The intent is that the SensorML will be a compliant implementation of the ISO standard.

In March 2001, Dr. Botts was accepted to participate in the OpenGIS Consortium (OGC) Military Pilot Project (MPP-1) with an emphasis on introducing the concepts of the SensorML and ODM into the OGC Sensor Web Enablement (SWE) initiative.

In September 2001, the OGC IP initiated the Open Web Services (OWS) project with one of three threads focused on the initial design and testing of SWE concepts. Within that initiative, the SensorML provides a key component for describing the characteristics and capabilities of any sensor, whether the sensor is designed for in-situ or remote observation. Because of the focus on in-situ sensor within OWS 1.1, the initial phase of this activity was to drive the development of the SensorML into several new directions, including:

- Generalizing the structure and grammar to support both in-situ and remote sensors
- Providing more generalized and more robust support for describing measurement characteristics, regardless of whether the sensor measures radiation, chemical concentration, velocity, temperature, or any other physical phenomena
- Further migration toward an XML schema, with inheritance from other schema, such as OGC GML
- Providing more robust support for non-scanning sensors, such as profilers and frame cameras

From May to December 2002, OGC IP conducted a follow-up OWS (OWS 1.2) project. With regard to SWE activities, the focus was extended to remote sensors on static or dynamic platforms. With regard to SensorML, the activities related to this project included:

- General refinement of SensorML through

- Incorporation of schema components from ISO 19115
- Determine of the role of GML within SensorML
- Extension and refinement of support for multiple sensor geolocation models, including scanner/profilers, frame cameras, and rapid position coordinates (RPC).
- Support for dynamic platforms
- Refinement and extension of definitions for sensor response characteristics
- Incorporation of concepts and schema developed under the OGC coordinate systems and coordinate operations group

Additional support for SensorML refinement has been provide by the National GeoSpatial Intelligence Agency, the Joint Interoperability Test Command, and the OGC OWS 2.1 Project. A major part of efforts from March 2004 to October 2005 has been on blending SensorML with several ongoing standardization efforts, including particularly TransducerML, ISO 19130, and the U.S. Measurements and Signals Intelligence (MASINT) standardization program. These efforts have resulted in significant refinement of SensorML, as well as improved confidence in the abilities of SensorML to meet the desired objectives.

In March 2005, OGC began the OWS 3 Interoperability Project, that has allowed SensorML and the rest of the SWE components to mature and to be tested through implementation and interoperability experiments. This document is the one of the products of the OWS 3 effort ending in October 2005.

8 Design Criteria and Assumptions for SensorML

8.1 Basic definition of a sensor

Sensors are devices for the measurement of physical quantities. A sensor measurement can be modeled as a process by which an input phenomenon is observed by the sensor at some discrete moment in time. Some measure of some property of that phenomenon is then output from the sensor. The values of the measurement are dependent on the sampling and response characteristic of that sensor, as well as on the sampling and detection methodologies. Often, either through hardware processing or subsequent processing in software, the raw observations are often processes to higher-level knowledge. The SensorML model does not try to define where observation measurement and observation processing begin or end. These are both simply considered as part of the process.

There are a great variety of sensor types from simple visual thermometers to complex electron microscopes to radiometers on-board earth orbiting satellites. In some cases, sensing may be accomplished by a person rather than a device, and the result of the “measurement” may be a category rather than a numeric quantity.

Typically, sensors fall into one of two basic types. In-situ sensors measure a physical property within the area immediately surrounding the sensor, while remote sensors measure physical properties at some distance from the sensor, generally by measuring radiation reflected or emitted from an observed object. Regardless, any geometric properties described within the SensorML schema are defined within the sensor's local coordinate frame and are only related to the geospatial domain through it frame's association with the platform, mount, and their association with some geospatial reference frame. For example, to fully describe a wind profiler's wind speed and direction measurements, the height of the sensor needs to be known as that sensor could be situated on the roof of a building, mounted to a 10-meter tower, or sitting at ground-level.

8.2 Applications of SensorML

There are at least three fundamental approaches to implementing and utilizing SensorML instance documents. In the first approach, one would define a sensor system description that remains valid over a long period of time (including perhaps the future), and all time-varying parameters or other data would be considered as potential input into the system process chain. This method can support both real-time and archived observations and provides maximum flexibility for on-demand discovery and processing of observations.

The second approach is to provide a SensorML system description that is only valid over a certain period of time. For example, a data supplier might provide a satellite image derived from a scanner, and as part of the metadata for that image, would provide a SensorML system description that allows one to geolocate or further process that imagery data.

The third approach considers a SensorML instance document as instructions on how to further process existing data. In other words, a scientist could advertise and provide a particular algorithm, using a SensorML process chain definition, that can take Earth observations from a particular sensor and derive sea surface temperature or algae concentration, for example. Applying this "drag-n-drop" executable process to any segment of the observed data would allow on-demand generation of this higher-level product, as well as enable the ability to tweak parameters within the algorithm, perhaps. Additional, one could provide simply a process chain that allows geolocation of remotely sensed observation, without necessarily needing to describe the process by which the observations were made.

8.3 Sensor aggregation concepts

*** discuss Composite Design Pattern for Process and Sensor … remove arrays and groups

Transducers are typically simple devices (real or virtual) that take one input signal and generate one or more output signals. These can include simple detectors, actuators, and filters. These are typically based on simple models that map the input values to output values, based on a few fixed or adjustable parameters. There are other simple processes that might exist either in hardware or software that do basic conversion of input to output. The methodology for each of these devices is fairly well defined and can be supported in

software using a general model consisting of calibration curves and look-up-tables. In SensorML, these simpler atomic processes are defined as a *ProcessModel*.

Sensors typically consist of several such processes (or transducers) that can be linked in series or parallel resulting in a mapping from input to output that is better defined as a process chain. Other chains can be conceived that take inputs and through a series of individual processes generates new output values. In SensorML, these process chains, including sensors, are modeled as a *Process*.

For most sensors of interest to the geospatial community, there is a need to relate the sensor observations to some temporal and spatial domain in order for them to be relevant. In SensorML, a process chain or a collection of process chains that can be related in space and time are modeled as a *System*. Thus, typically, a System defines a collection of sensors that are related spatially and temporally to one another, as well as related to some geospatial domain.

8.4 Relationship of the sensor to a platform

A sensor system is typically composed of one or more sensors and a platform. Only the sensor element is viewed as being able to measure physical quantities. A platform such as an aircraft (carrying a frame camera) may be able to determine its own instantaneous orientation and position, and in such a case, these measurements would be obtained by other sensors attached to the platform. Within SensorML, platforms and sensors are treated as separate entities. Typically a sensor will be modeled as a Process or ProcessModel, whereas a platform will be modeled as a System that contains all of the sensors and defines positional and temporal relationships among them.

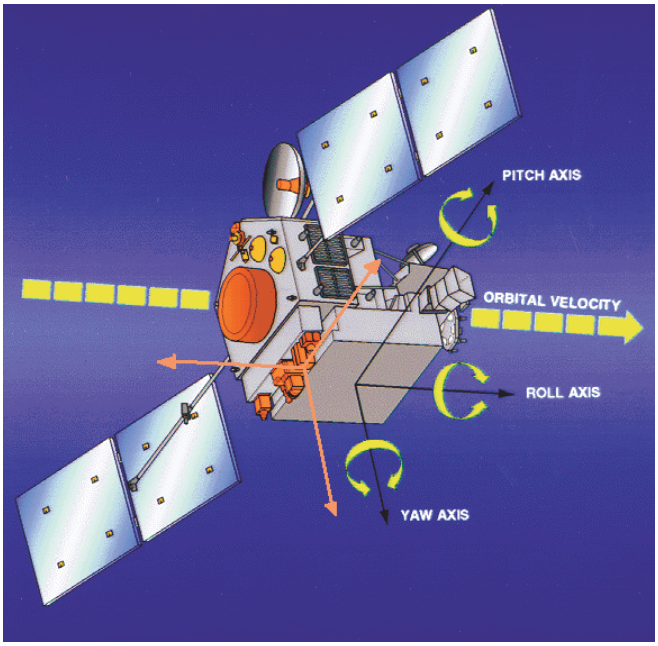


Figure 8.1. Relationship of sensor frame (pink) to the moving platform frame (black).

The geospatial position and orientation of a sensor is often derived from the platform on which it is mounted. It is therefore often beneficial to relate the sensor's coordinate frame to its coordinate frame of the platform's (e.g. through mounting angles and position) and then depend on the platform for determining geospatial positioning. All of the frames of reference can in general be dynamic or static. Figure 8.1 illustrates the relationship of a sensor's frame (in pink) that is fixed but has been translated and rotated relative to the moving spacecraft frame (in black).

Based on the type of sensor and on the characteristics of the platform, a sensor system can be classified according to Table 2.1. Based on the dynamics of the platform, a sensor system may be fixed (stationary) or mobile (dynamic). Based on the sensor characteristics, a sensor system may measure either in-situ (in place) or remotely. Thus, a remote sensing atmospheric profiler might be fixed to the ground (fixed remote) or attached to an aircraft (mobile remote). Similarly, an in-situ water quality sensor might be attached to a fixed station (fixed in-situ) or to a boat (mobile in-situ).

As previously discussed, we separate the description of the sensor from that of its platform. The main importance of the associated platform(s) is in providing the relationship of the sensor and its observations to some relevant external coordinate system (for example, a geospatial reference system).

Measures / **Mobility**	**In-Situ**	**Remote**
Fixed	Stationary O2 Probe	Doppler Radar station
Mobile	"Diving" Salinity probe	Airborne LIDAR

Table 8.1. Relationships between in-situ and remote sensors and dynamic and fixed platforms.

8.5 Coordinate reference systems

All geometric and temporal characteristics of a sensor system must be related to a specified coordinate reference system (CRS). Within the SensorML, definitions for sample geometry, look angle, and collection geometry are often described relative to the sensor's CRS. In such cases, it is only through the sensor's relationship to its mount and platform(s), that the sensor and its measurements can be related to an external CRS, such as geographic latitude and longitude.

This is accomplished by CRSes and describing their relationships to one another. The relationship between CRSes can be accomplished either by describing a transform process between the coordinate reference systems or by defining the state of the object relative to a CRS. For instance, an individual sample's geometry (e.g. shape and size) is defined in the localized coordinates of that sample. Its relationship to a sensor's frame may be specified through a collection geometry definition. The sensor's CRS may, in turn, be related to its platform's CRS through its mounting angles and position. Finally,

the platform's CRS is related to a geospatial CRS by defining its position and orientation within that CRS. The successive transformation of each of these coordinate frames into its parent CRS provides the information necessary to georegister the sensor's measurements. Its is also possible, using particular sensor models within SensorML, to relate the sample geometry and position directly to a geospatial CRS.

For a remote sensor, it is necessary to determine the intersection of a pixel's look ray and the surface of the sensor's target (e.g. the Earth's ellipsoid). Typically the look angle and the sensors target are transformed into a common spatial reference frame, such as the Earth-Centered Earth-Fixed (ECEF) or Earth Centered Inertial (ECI) reference system. For in-situ sensors, the process is typically much easier.

The CRS concept will also be applied to temporal domain when applicable. One local time frame that is useful for defining the geometry and dynamics of scanners, is seconds past the start of a scan (scan start time). Also, for some sensor systems, time is recorded relative to a local clock or the start of the mission. In such cases, time frames and their transforms to "Earth time" will be defined in SensorML.

8.6 Measurement / observation concepts

A sensor is designed to *measure* a particular property within a given *sample* space. When these measurements are taken, they result in an *observation* that may be immediately utilized or stored. In its lowest level, this observation is typically a proxy measurement of some property other than the desired physical property, itself. For example, an observation may be the height of mercury in a thermometer or the voltage across a circuit. In order for these observations to be related to a more useful physical property, a new observation must be derived using known sensor calibration functions and perhaps other processing algorithms.

SensorML allows one to describe whatever level of observations the creator of the document wishes to expose. For example, one might specify that the sensor measures raw voltages and then provide calibration descriptions that would allow conversion to other physical quantities. Alternatively, or in addition to, the sensor description might specify that the sensor measures temperature, and then expose the calibration used to derive those temperature values, or not.

A SensorML document will describe what physical properties are measured by the sensor, as well as information concerning the properties and quality of these measurements. In addition, a SensorML document may provide or link to the values of these measurements using one or more data provider types.

The definition of the data provider schemas is outside of the scope of this document. However, it is important that observations be capable of being associated with the appropriate sensor and with the appropriate measurement description is associated with that sensor. One such encoding is the Observations and Measurements schema.

8.7 Sensor Response Characteristics

The response characteristics of a sensor determine how the sensor will react to a particular stimulus (i.e. Phenomenon) and how it will operate under given environmental

conditions. Within the sensor response characteristics will be specifications for sensitivity (e.g. threshold, dynamic range, capacity, band width, etc.), accuracy and precision, and behavior under certain environmental conditions.

A very large number of sensor response characteristics can be deinfed using a general response model (e.g. the transducer model defined in Annex C). However, there may be specific sensors that require addition of different parameters to fully describe their response behavior. Where possible, these should be derived from the general model.

8.8 Sample and collection geometry concepts

As discussed above, a sensor measures some property within a spatially and temporally defined sample. In the case of an in-situ sensor, this sample includes some spatial volume in the immediate vicinity of the sensor. This volume may be infinitesimally small or it may be unknown or unimportant. For remote sensors, the sample involves some volume or surface area located away from the immediate vicinity of the sensor.

The geometry of a sample may be specified relative to any coordinate system. However, particularly for a remote sensor, the geometric descriptions in SensorML are typically defined relative to the sensor's local coordinate frames and not a geospatial coordinate frame. As discussed before, this allows the same sensor model to be "attached" to any stationary or dynamic platform without a need to significantly change the SensorML description. In such a case, an individual sample's geometry, such as perhaps its size, shape, or point-spread function, is described relative to a local sample coordinate frame.

This sample frame can be related to the sensor's frame by either a simple transformation or in the case of collection of samples, by a more complex transformation involving arrays or scan patterns. Possible transformations for sample collections include unstructured grids, regular arrays, scanners, frame cameras, and mathematical functions.

9 SensorML Conceptual Models

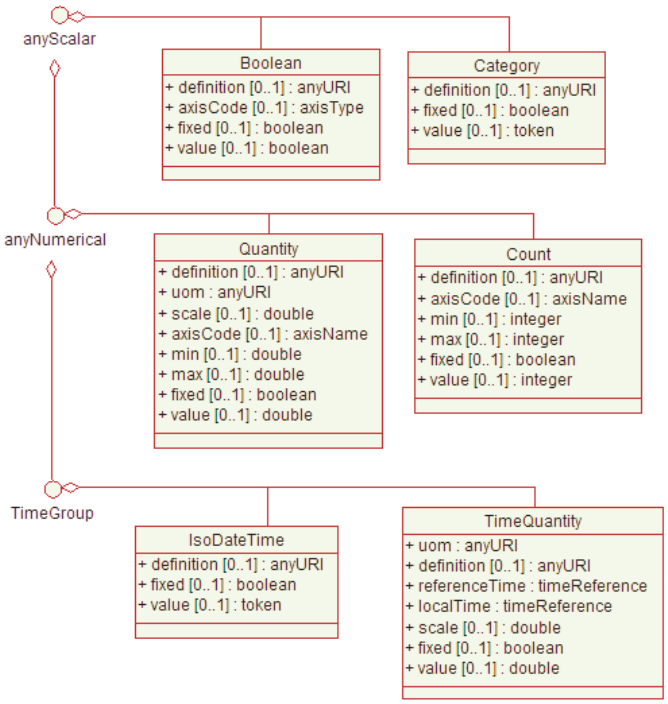
In SensorML, all components are modeled as processes. This includes components normally viewed as hardware, including transducers, sensors, sensor systems and platforms. All components are modeled as processes that take input, and that through the application of an algorithm defined by a method and parameter values, generates output. All such components can therefore participate in process chains which are themselves processes with inputs, outputs, and parameters.

In a sense, SensorML can be viewed as a process model language with an emphasis on application to sensor data. SensorML does not try to replace other existing technologies through which distributing processing might be exnabled, but instead provides a language through which specific process models and process chains can be serialized and executed via SensorML-enabled software.

9.1 SWE Common Components

9.1.1 Simple Data Types

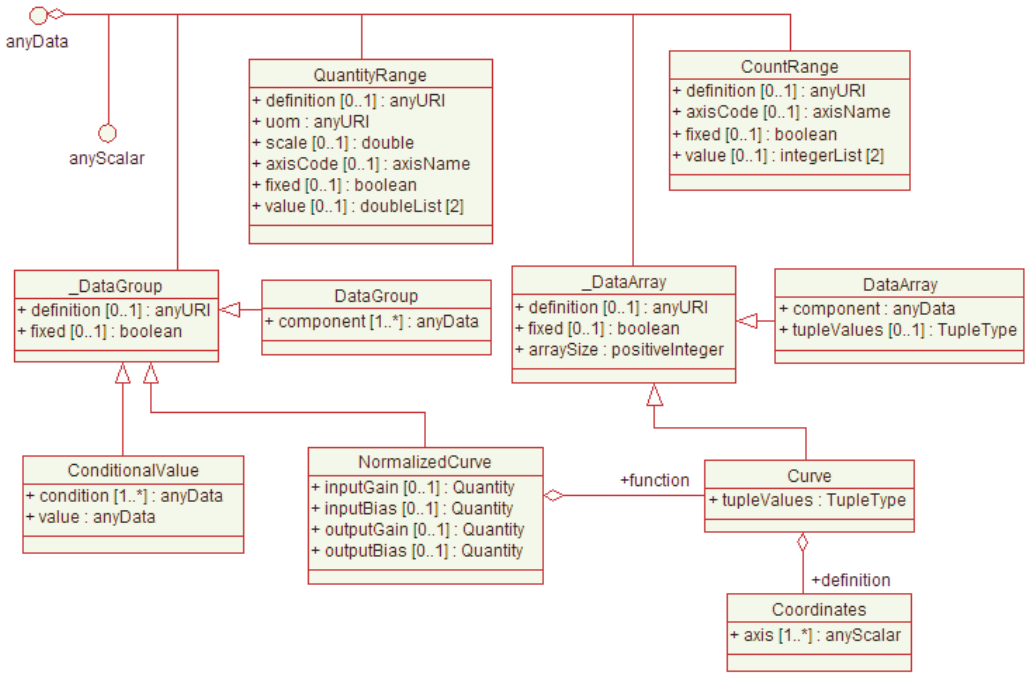
This document defines several basic value types and data encodings that will exist in the Sensor Web Enablement Common namespace (swe). These include definitions that are expected to be shared among all SWE encodings and services. Of primary importance are the definitions of fundamental data values, originally defined in previous Observations and Measurements. These include scalar values that we chose to define as *Quantity*, *Count*, *Boolean*, *Category*, and Time. Within SensorML, these serve as the basis for specifying all inputs, outputs, and parameters within a Process.



Quantity represents a data type that takes decimal as its value and represents a real value along some defined axis. The *definition* attribute identifies the context of the value and takes *anyURI* as its value (e.g. urn:ogc:def:phenomenon:1.0:temperature). A *Quantity* provides the units of measure through the *uom* attribute, as well as an optional *scale* factor. The *axisCode* attribute allows one to relate the *Quantity* to the axis of a defined reference frame. For *Quantity* values that are variable (i.e. *fixed*="false"), one may provide minimum and maximum allowed values through the *min* and *max* attributes, respectively.

Count represents a data type that takes an integer as its value and includes many of the same attributes as *Quantity*, including *definition*, *axisCode*, *min*, *max*, and *fixed*. *Boolean* takes true or false as its value, and takes *definition*, *axisCode*, and *fixed* as attributes. A *Category* takes a token (i.e. simple string) as its value and represents a data value that is a subset of some larger grouping of values as defined in the *definition* attribute (e.g. "dog" may be the value of a category defined by "mammal").

Time is supported is treated as a special type of numerical, and can be supported using an *IsoDateTime* which takes as its value, a date and time in the ISO 1806 standard format (e.g. 2005-10-08T16:40:03.45Z or 2005-10-08T16:40:03.45Z/2005-10-08T18:35:20.34Z for a time range). Using the *TimeQuantity*, one may also provide time as units beyond an epoch, such as seconds past 1970-01-01T00:00:0.0Z (i.e. Julian time), or seconds past a mission start time.



9.1.2 Aggregate Data Types

These basic data types can be grouped within any of several aggregate objects. *QuantityRange* takes two decimal values separated by white space, which represent the minimum and maximum values, in that order. Similarly, *CountRange* takes two integers.

DataGroup (derived from abstract *_DataGroup*) defines some logical collection of data values of any type. Like individual data values, the *DataGroup* itself can contain *definition* and *fixed* attributes. For example, a particular DataGroup might define a position (with a definition, perhaps of urn:ogc:def:phenomenon:position) and elements including time, x, y, z, deltaX, deltaY, deltaZ). The elements of a *DataGroup* are defined using the *component* property which takes *anyData* as its value.

A *ConditionalValue* is a special type of *_DataGroup* whose *value* is based on one or more *condition* properties. For example, an atmospheric pressure measured at 25 degrees Celsius.

A *DataArray* (derived from the abstract *_DataArray*) defines an array of values of the type specified in *component.* Since *component* is of the type *anyData*, *DataArray* may be used to define a simple array of *anyScalar* value (e.g. temperature). However, it may also be used to define an array of a *DataGroup*, in which case the value would be expected to be an array of like tuples (e.g. 200 occurrences of the position *DataGroup* described above). Similarly, the *DataArray* allows for the nesting of arrays, since the component of a *DataArray* may itself be another *DataArray*. Examples that should make these concepts clearer will be provided in the encoding sections to follow. In addition to the *definition* and *fixed* attributes, *DataArray* takes an *arraySize* which specifies the number of elements in the array. The value of the *DataArray* is provided by the *tupleValues* property, to be discussed in more detail in later sections.

A *Curve* is a special type of *_DataArray* which provides an ordered set of coordinate values along one or more axes, as defined in the *Coordinates* element. Like *DataArray*, the curves coordinate values are provided by the *tupleValue* property. When used inside of a process, the *Curve* provides a look-up-table by which one can map *input* values to *output* values.

A *NormalizedCurve* takes a *Curve* as the value of its *function* property, but adds *inputGain*, *inputBias*, *outputGain*, and *outputBias* as additional properties. These properties alter the mapping of input values to output values according to the equation, based on the following algorithm:

normalizedInput = input * inputGain + inputBias

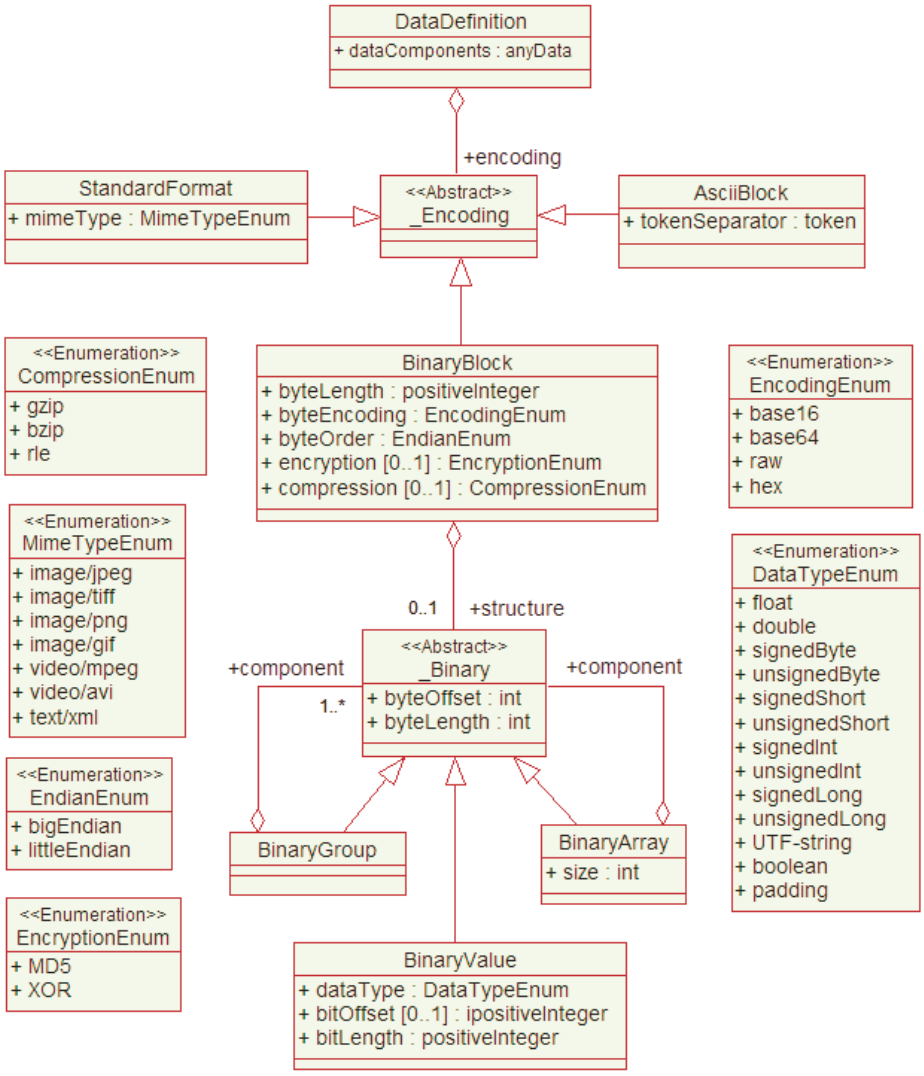
obtain normalizedOutput from the Curve Look-up-table based on normalizedInput

output = normalizedOutput * outputGain + outputBias

Using a normalized curve allows one to reuse a general curve form, while allowing the curve values to be altered within a process chain by changing bias and gain values. A normalizedCurve should not have more than two Coordinates.

9.1.3 Data Definition

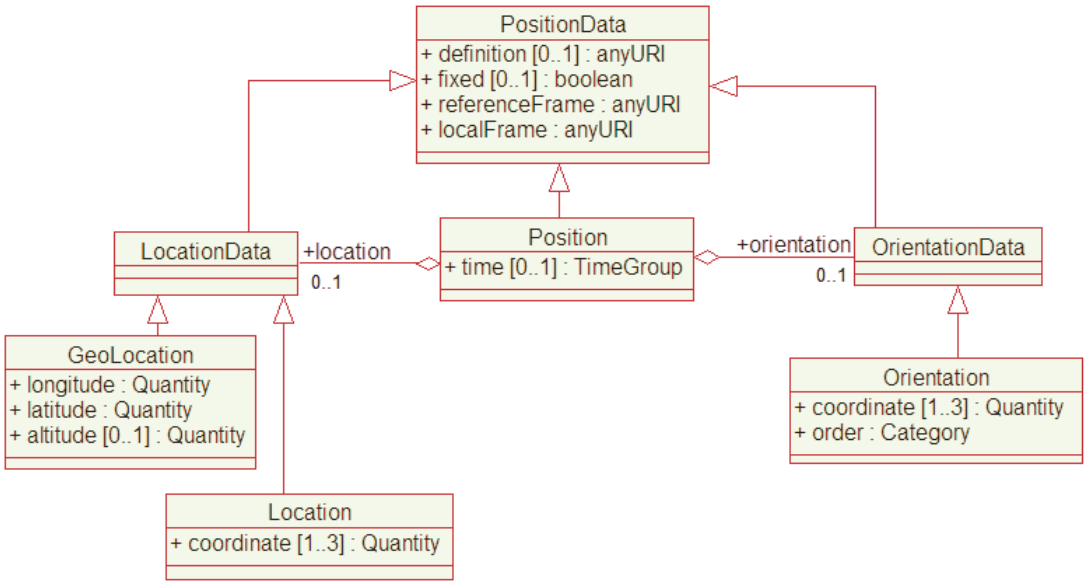
The SWE framework utilizes common data structure and encodings to achieve a higher level of compatibility and reuse of software between the various encodings and services. By providing a schema to explicitly describe the data components expected and the encoding of the values, the SWE Common *DataDefinition* also allows much flexibility for providing large amounts of data in a compact form. *DataDefinition* includes two major properties: *dataComponents* which uses *anyData* types to describe the nature and the meaning of data elements within a tuple, and *encoding* which describes whether the data values within the tuple are encoded as an ASCII block, a binary block, or as a simple, standard MIME-type encoding (e.g. jpeg image). The use of the *DataDefinition*, with clarifying examples, will be discussed more fully in Section 10.



9.1.4 Position Data

Positional information is vital to the processing and utilization of sensor observations. Positional information includes not only location, but also orientation. For remote sensors or for any sensor mounted on a dynamic platform, positional information is also temporal in nature. Thus, positional information can also include time, as well as various dynamic state properties such as velocity and acceleration.

In SensorML, positional information is considered to be the same as any other observation data. Positional data be output by some processes and can serve as input or parameters of others. Time and position may also be mixed with other data within a tuple or cluster. Thus, in SensorML, *PositionData* is derived from *DataGroup* and uses SWE Common *anyData* type for *coordinate* definitions.



Position has no meaning unless it is specified relative to some spatial-temporal coordinate frame. In addition to specifying position relative to some geodetic coordinate frame, it is important to be able to specify locations relative to engineering coordinate frames. In SensorML, positional data is used to specify the position of a local coordinate frame to a external reference coordinate frame. These are specified by the *localFrame* and *referenceFrame* attributes, respectively.

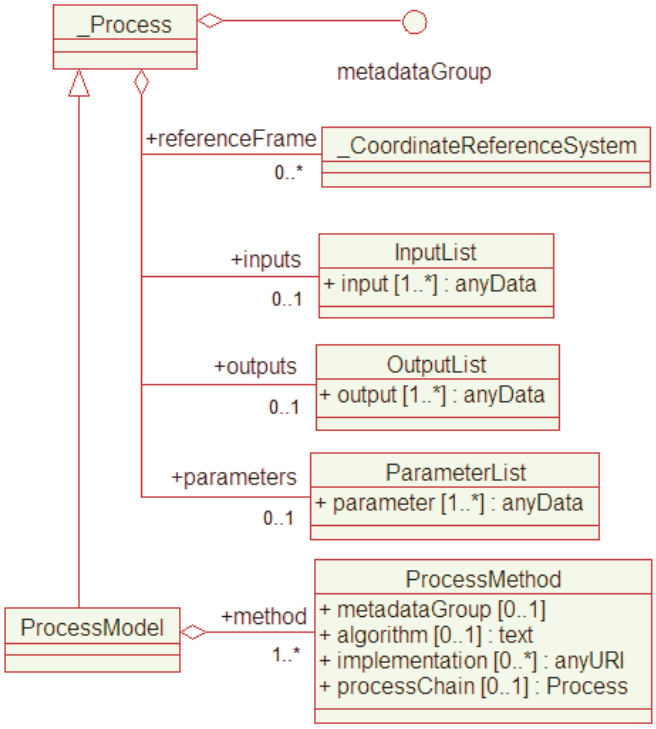
LocationData provides the coordinates of the origin of the *localFrame* relative to the *referenceFrame*, while *OrientationData* provide the angular relationships between the axes of the *localFrame* and those of the *referenceFrame*. Further discussion of *PositionData*, along with examples, will be provided in Section 10.

9.2 Base Process Model

Process models and process chains in SensorML are intended as serializations of executable components. Thus, an instance of a process in SensorML should describe the inputs and outputs expected, as well as the parameters and methodology required to create output values from input values. Process models and chains are expected to be linkable, and these links are expected to be described within a process chain or system.

9.2.1 Abstract Base: _Process

_Process is the base for all processes in SensorML. All processes include *inputs*, *outputs*, and *parameters*. These three properties all take lists that include members of *anyData* type. Therefore, all descriptions of *inputs*, *outputs*, and *parameters* within SensorML processes utilize the SWE Common data types, described earlier, including *Quantity*, *Count*, *Boolean*, *Category*, *_DataGroup*, and *_DataArray*. The use of these common data types throughout the SWE framework of encodings and services, supports linking between SWE components as well as providing a high degree of consistency.



The properties *inputs* and *outputs* obviously represent "ports" where outside processes exchange data with this process. However, in SensorML, any *parameter* elements that do not have the attribute *fixed* equal to true, can also be considered as a potential input into the process. In other words, the value of any variable *parameter* within a process can be set or affected by outside processes or data sources.

In addition to *inputs*, *outputs*, and *parameters*, all processes in SensorML can provide several properties that, for the sake of clarity, have been collected within a *metadataGroup*. The properties within the *metadataGroup* will be described in detail in Section 9.5. These data, while important for resource discovery, for qualification of results, and for assistance to humans, are not considered essential to the execution of the process within a process chain. Thus, all data or information required for actual execution of the process should be included within the *inputs, outputs*, and *parameters* properties. It is possible, and encouraged that those parameters desired for resource discovery be linked to or copied within the appropriate metadata section.

In addition to describing purely mathematical processes, SensorML is expected to describe processes that have some relationship to space and time. These might include transducers (detectors and actuators), sensor systems, samplers, and sensor platforms, for example. For much of the processing of sensor data, it is important that temporal and spatial reference frame be described and that the relationships between various reference frames be able to be explicitly defined. The *referenceFrame* property is optional and should thus not be included for processes where temporal or spatial state is meaningless.

For this reason, *_Process* includes a *referenceFrame* property that expects a description of any spatial or temporal coordinate reference system that might be used for interrelating internal components within the process, as well as reference frames that might allow this component to be related to other components within a process chain or system. SensorML will utilize the Coordinate Reference System (CRS) models defined for GML.

9.2.2 ProcessModel and method definition

The base concrete model for processes in SensorML is a *ProcessModel*. *ProcessModel* derives all properties from the abstract base class *_Process*, and adds a description of the process methodology through the *method* property. *ProcessModel* is used to define more or less atomic processes that are expected to be used within more complex process chains.

The *method* property provides recommended or suggested methodology by which one transforms input values to appropriate output values, based on the provided parameter values. The desire is that SensorML instances will define process chains that are self contained and executable without requiring apriori knowledge of complex processes. For this goal to be realized, SensorML-enabled software must understand how to execute the individual process models within a chain and, of course, support the flow of data between these process models according to the link defined within the process instance. The support for enabling software that can execute an individual process model is provided by the *method* property.

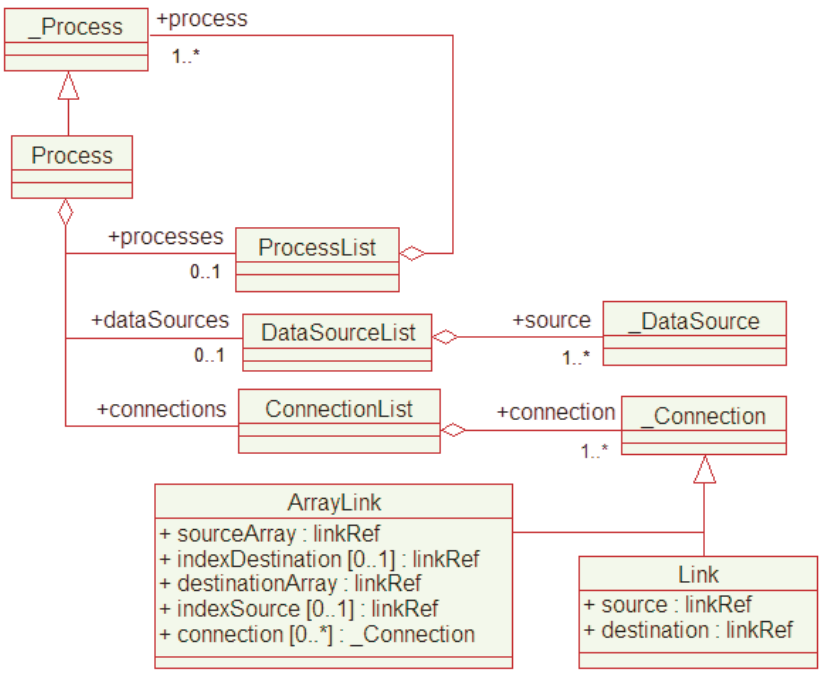
SensorML supports three fundamental means for defining the method. The first means, using the *algorithm* property, is by providing simply a textual description of the algorithm in such a way as to allow a competent programmer with enough knowledge to encode the algorithm in software. Relying totally on this method to support drag-n-drop execution of process chains will require any SensorML-enabled software to provide or have access to a library of modules that includes perhaps code for each possible process model.

The second means, utilizing a URI link provided in the *implementation* property, has the potential to support execution of unknown process models, possibly at runtime. The *implementation* URI should provide a link to either (1) an executable module of code (e.g. a Java or C# class file) that can be imported as part of the software, preferably at runtime, (2) a MathML instance of the algorithm, or (3) a remote process or service that is capable of executing the process. We expect that a follow-up document will more fully define possible API's and implementation specifications for these *implementation* options. The third means of specifying the method of a Process is to use a *processChain*.

It is expected that the *ProcessMethod* for a given *ProcessModel* will be defined within an online registry or other online resource. It is possible for a *ProcessMethod* to provide any and all of these different means of specifying the method, and may include multiple *implementation* references.

9.3 Composite Process (Process Chain)

In SensorML a process chain "snippet" is defined using the *Process* element. *Process* is based on a Composite Design and thus consists of a collection of other *_Process* elements while it is itself derived from *_Process*. In short, a *Process* has all the properties of a *_Process*, including *inputs*, *outputs*, *parameters*, and *metadataGroup*, but adds *processes*, data *sources*, and *connections* properties.



Since *Process* is itself of type *_Process*, a process chain in SensorML can take other process chains as components in addition to more atomic *ProcessModel* components. The *inputs* and *outputs* of a *Process*, in essence, define the beginning and ending data

components of the process chain, respectively. Only data components exposed through the *Process inputs*, *outputs*, or *parameters* properties are available for linking with outside processes.

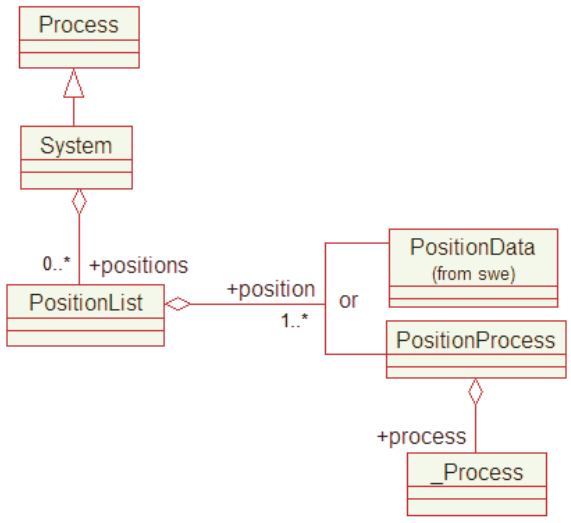
In addition to containing processes, a process chain can include *dataSources* that provide input and parameter values to processes in the chain. These *dataSources* take, as their value, a derived *_DataSource* which is expected to include a SWE Common *DataDefinition* defined earlier. The *DataDefinition* provides a *dataComponents* definition that is consistent with process *inputs*, *outputs*, and *parameters* and thus utilize the same *connections* specifications used between processes. Encodings for derived *_DataDefinition* types will be discussed in Section 10.

In *Process*, the process chain is defined by first describing all of the *processes* and *dataSources* in the chain and then describing the appropriate *connections* between components. The most basic *connection* is a *Link* which merely provides references to the *source* and *destination* data components. The *source* is usually a data component within a data source or within a process *OutputList*. It can consist of a *anyScalar* data type or a specific aggregate type, defined as a *DataGroup* or *DataArray*. The *destination* data component is typically an *input* or *parameter* of a process. The specifics regarding how a *linkRef* is defined will be described in the encoding Section 10.

An additional connection, *ArrayLink*, allows one to specify *source* or *destination* references pointing either to a complete *DataArray* or to a particular element within a *DataArray*.

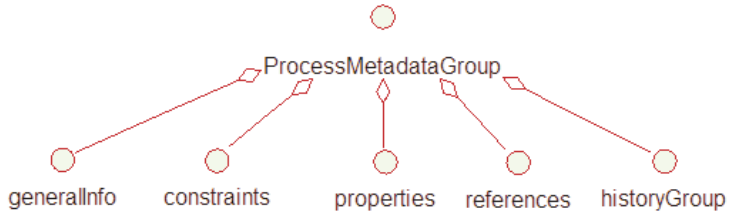
9.4 System

System is derived from *Process*, and thus inherits all of the properties of the process chain. *System* provides connectivity of a process chain to the real-world by including position information. Positions can be provided using *PositionData* or through a process defined as either a *ProcessModel* or *Process*.



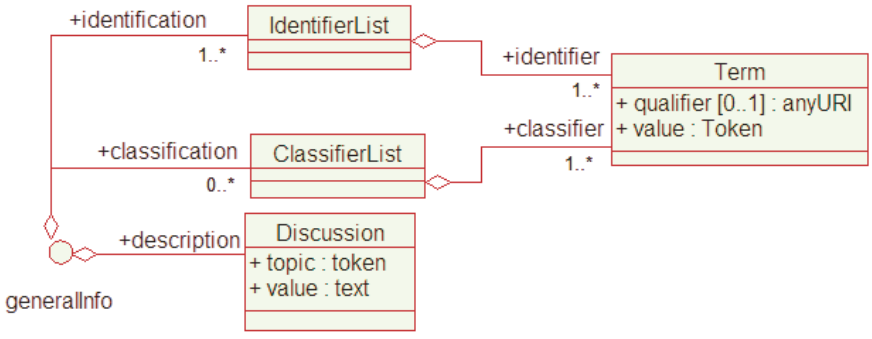
9.5 Process Metadata Group

Every element in SensorML derived from *_Process* by extension inherits the *metadataGoup* which includes five optional groups of metadata. As discussed previously, these metadata are provided primarily to support discovery of resources, qualification of process results, and assistance to humans. These metadata include identifiers, classifiers, constraints, capabilities, properties, contacts, documentation sources, and history. Each of these groups will be discussed in detail below.



9.5.1 General Information (Identification and Classification)

The *generalInfo* group includes three properties, *identification*, *classification*, and *description*. The *identification* property take a *IdentifierList* which includes one or more *identifier* elements. An *identifier* takes a *Term* as its value. The *Term* has a *qualifier* attribute that specifies in this case the type of identifier. For example, an identifier with a qualifier of "urn:ogc:def:identication:tailNumber" might take "N291PV" as its value. Other possible qualifiers for identifiers might include, perhaps, shortName, longName, acronym, serialNumber, or partNumber.



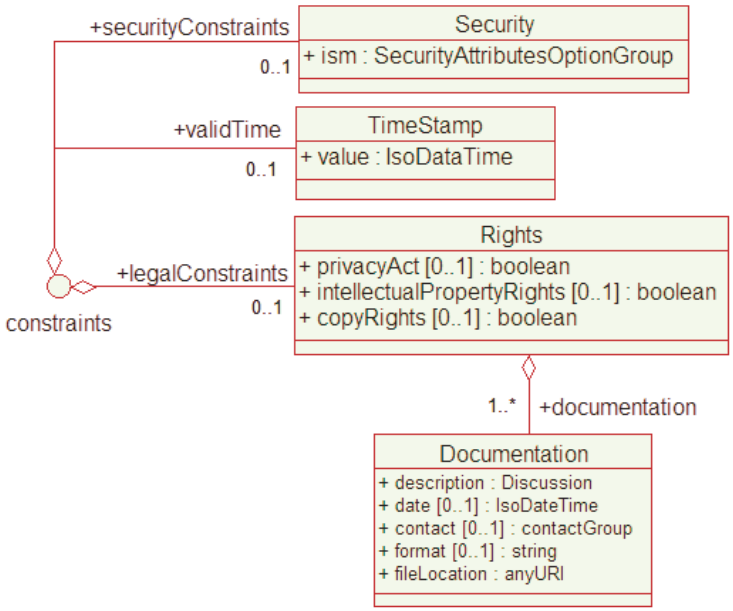
Similarly, *classification* provides a list of possible classifiers that might aid in the discovery of processes, sensors, or sensor systems. Qualifiers for a classifier *Term* might include, for instance, sensorType, observable, processType, or mission. The generalInfo group also provides a description property which allows human readable description of the process.

9.5.2 Constraints

A SensorML resource description can be constrainted by three properties: national and international *securityConstraints*, *validTime*, and *legalConstraints*. The model for specification of security constraints is based on the Security Banner Marking model of the Intelligence Community Information Security Marking (IC ISM) Standard.

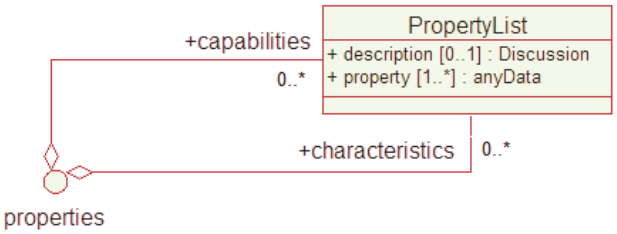
The *validTime* property indicates the time instance or time range over which this process description is valid. Time constraints are important for processes in which parameter values or operation modes may change with time.

The *legalConstraints* property specifies whether Privacy Act, Intellectual Property Rights, or copyrights apply to the content of the process description or its use. In addition to the boolean attributes, *privacyAct*, *intellectualPropertyRights*, and *copyrights*, *legalConstraints* take *documentation* as its value, thereby providing for more explicit description of the legal constraints.



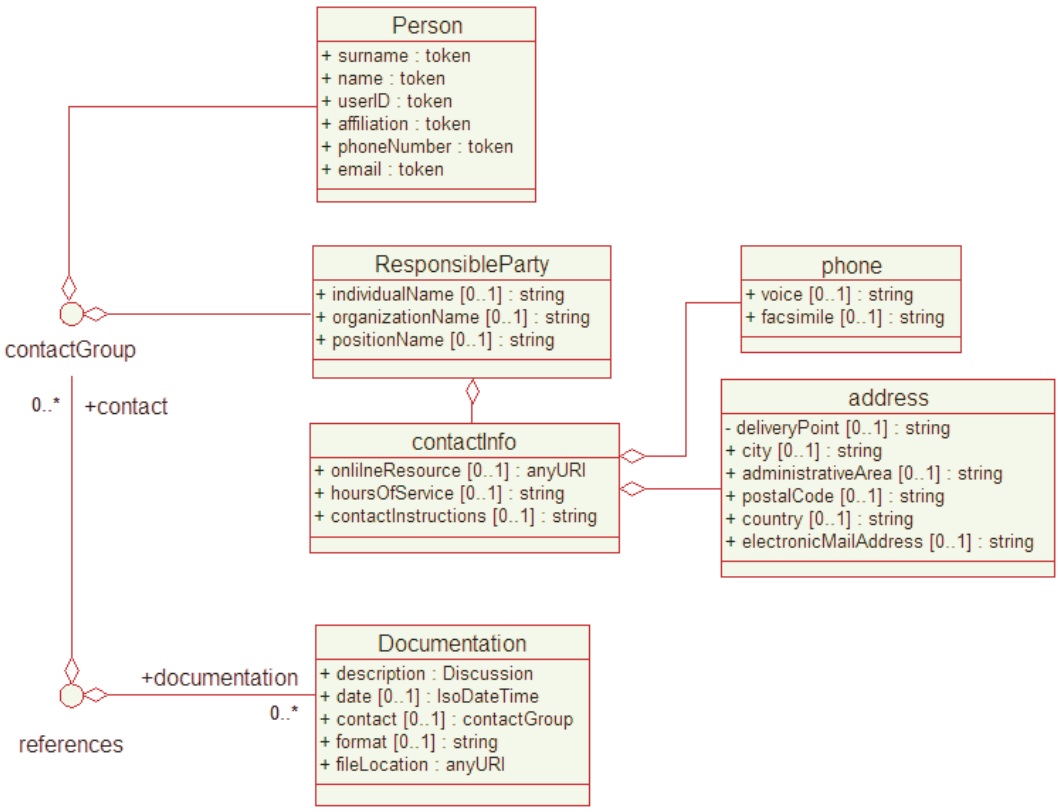
9.5.3 Properties (Capabilities and Characteristics)

Any SensorML resource (e.g. a process, sensor, sensor system) may possess various *characteristics* or *capabilities* that are useful for its discovery. For example, a remote sensor might measure radiation within a certain band (e.g. IR) and at a certain ground resolution. A process might have certain quality constraints. Any resource may have certain limits (e.g. operational and survivable limits), based on physical or mathematical conditions. The *characteristics* and *capabilities* properties take a *PropertyList* type as their value, which allows for grouping of various properties using SWE Common *anyData* types.



9.5.4 References (Contacts and Documentation)

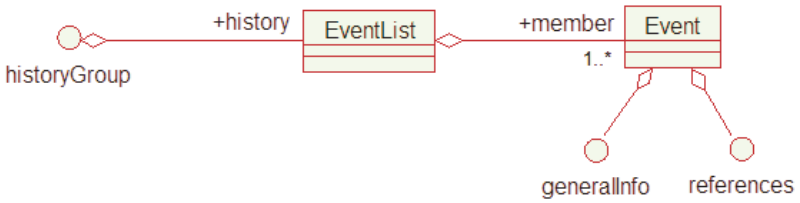
The *references* group provides *contact* and *documentation* properties that are useful for human consideration. The contact property takes two possible values for contact information: *Person*, which is based on the IC Department of Defense Discovery Metadata Specification (DDMS), and *ResponsibleParty*, which is based on ISO 19115. The documentation property provides a description and URI to an online resource (e.g. specification documentation, peer-reviewed algorithm literature, etc).



9.5.5 History

Within SensorML, history of a resource can be provided through a collections of *Event* objects. These are provided within an *EventList* that serves as the value of the *history*

property. Events might for instance, specify calibration or maintenance history of a sensor, or changes to an algorithm within a process.



9.6 SensorML as Applied to Sensors

The choice to model sensors and sensor systems as a process is in recognition that it is very difficult at times to decide where sample collection and detection end and where processing begins. It is also logical that any transducer, whether a detector, actuator, or filter, merely acts as a process that converts on input into another output. For a detector, the input is typically some physical phenomenon (e.g. temperature, position, electrical current, etc.) and the output is either some other physical phenomenon (e.g. voltage) or some digital number (DN) representing a quantification of the input phenomenon. Likewise, an actuator takes some physical phenomenon or digital signal as input and outputs a phenomenon (e.g. motion, sound, temperature, etc.) as reaction.

The SensorML model therefore also recognizes that observed data can be the result of a measurement, a simulation, or data processing chain, and thus makes no distinction between these data nor the description of the processes that created them.

9.6.1 Sensor Response and Geolocation

It is further recognized that most sensor observations consist of at least two processes: a sampling process and a detection process. The sampling process determines the subset of the surrounding environment on which a measurement is taken. For simple in situ sensors, the sampling may be based solely on the proximity of the environment element to the detector. For a microphone and for remote sensors of radiation, the sampling process may consist of a combination of focusing devices (e.g. audio cones and baffles or optical lenses and mirrors) and on the properties of the transmitting media. Within the remote sensing community, the models for describing the collection or sampling process are typically referred to as “Sensor Models” or “Geolocation Models” and are crucial to determining the location of the observation within a geospatial domain. These Sensor Models can either be rigorous models that define the relative spatial-temporal transforms between sensor system components, or they can be functional models that define the mapping of observation elements to geospatial location based, perhaps, on polynomial equations.

The detection process describes the response of the sensor to the stimulating phenomenon once the sample has reached the sensor’s detector. This process has typically been referred to as the “Response Model”, with calibration curves being a fundamental part of

mapping the detectors input to output. Whether one chooses to treat the sampling and detection processes as separate processes or as one, is an implementation choice.

9.6.2 Observations and Data Encoding

In SensorML, the data components that result from measurements or processing from a sensor, sensor system, simulation, or process chain are specified as part of the *outputs* property. However, the *outputs* property of a *_Process*, only defines the data components of the sensor or process, but not the data encoding. Thus, a sensor system might describe that it outputs time, latitude, longitude, altitude, temperature, and pressure, and perhaps even what units of measure these are in, but it does not define that these values are encoded as base64, or some binary structure, for example.

Thus, *_Process* does not specify the encoding of the data from a sensor or the encoding of data that should be used as it is passed through a process chain. SensorML does define a *Data* model that provides both *dataComponents* (using SWE Common *anyData*) and encoding (using SWE Common *_Encoding*). This can be used to provide input and parameter values within a process chain, but the encoding description of a sensors output in reserved for observation models, such as those provided by O&M and TML.

9.6.3 Sensor Response Model

Any given sensor will include one or more detectors which convert input phenomenon to an output phenomenon or digital values. A simple detector can typically be defined as a *ProcessModel* in which its *parameters* define the response characteristics of the detector. The response model, and thus the parameters, may be as simple as a linear steady state response (i.e. calibration) curve, or may include a host of parameters such as non-linear steady state response, latency time, integration time, accuracy, impulse response, and frequency response.

This document defines a recommended (i.e. non-normative) encoding for a transducer that should be suitable as a *ProcessModel* for a large number of detectors, actuators, and filters. It uses the parameters listed above and provides support for an array of detectors whose parameter values can vary by detector index. It is to the advantage of the sensor community, that a general response model be developed that is common for a wide range of sensors. This would thus serve not only as a common response model for most transducer devices, but would also serve as a common base that one can extend to meet specific needs.

9.6.4 Sensor Models

The term “sensor model” is commonly used within the remote sensing community to refer to a model used for geolocating observations measured by a sensor located some distance from the source of the phenomenon (i.e. a remote sensor). Typically, a remote sensor detects radiation that is either reflected or transmitted by an object. The sensors that detect this radiation can utilize a complex system of lenses, filters, rotating mirrors, and various detectors, resulting in potentially complex models for determining the location of individual observations.

Remote sensors typically fall under a few categories, including frame cameras (including video cameras), profilers, and scanners (including line, conic, pushbroom, swishbroom, and volumetric scanners). In addition, several sensor models exist that utilize polynomial relationships between the location of the observation on an image, for example, and its location on a planetary surface. Such models include but are not limited to Tie-Point models, Rational Polynomial Coefficients, and Replacement Sensor Model.

The recommended (non-normative) transducer model defined in this document includes a general sensor model for specifying internal geometry of a sensor, through *samplePosition* and *ambiguityShape* properties. Other sensor models, such as those listed above, may be derived from _Process. As with response models, it is advantageous to the sensor community that the number of sensor models be kept to a minimum in order to assist in the development and maintenance of SensorML-enabled software capable of geolocating remotely sensed observations on-demand.

10 SensorML XML Schema Encoding

10.1 Encoding Principles

The SensorML models presented in the previous section, are encoded as XML Schema documents. It is envisioned that XML instance documents will be created for each process and system based upon the framework and rules provided by the XML Schema.

While it may be desirable to derive application schema for basic process models, it is neither expected nor desired that application schemas be developed for each system or system type, unless those schemas are derived by restriction in order to remove the inclusion of certain properties or to fix the values of particular properties.

10.1.1 XML Encoding Conventions

The *http://www.opengis.net/sensorML* namespace is used for fundamental core SensorML elements that can be applicable for a wide range of sensors. As they are developed, elements that will probably be useful only to a select group of sensor and platforms, or only within particular sensor communities, will utilize the *http://www.opengis.net/sml-x* namespace.

The SensorML schema also utilizes components from four external XML schemas:

Geographic Markup Language (GML), version 3.1.1, *http://www.opengis.net/gml*

SWE Common, *http://www.opengis.net/swe*

Observations & Measurements, *http://www.opengis.net/om*

Intelligence Community Information Security Marking, v2, *urn:us:gov:ic:ism:v2*

The “rules” used to encode the SensorML models into an XML Schema are similar to those used in GML. SensorML uses the concept that there are Objects and that these Objects have properties that take Objects or basis types (e.g. string, decimal, etc) as their values. While this convention occasionally results in deeper nesting of elements, it provides explicit association between Objects and is comparable to conventions utilized in the Resource Description Framework and the Semantic Web. SensorML follows the same lexical conventions used in GML:

- objects are instantiated as XML elements with a conceptually meaningful name in UpperCamelCase
- properties are instantiated as XML elements whose name is in lowerCamelCase
- abstract elements have an underscore prepended to their _Name
- the names of XML types are mainly in UpperCamelCase ending in the word “Type”

As in GML, most properties and Objects in the UML diagrams are encoded as elements within the XML Schema. Only a few properties, such as id, xlink components, type, definition, or units of measure (uom), are encoded as attributes within an element. Most

properties that are encoded as attributes with SensorML support some form of URI links to other Objects, dictionary entries, or online references.

10.1.2 ID, URI, and Linkable Properties

Like GML, SensorML makes extensive use of XLink components to support hypertext referencing in XML. This allows one to reuse Objects that are either internal or external to the instance document. This is supported by extensive use of the *id* attribute (taking an *xs:ID* as its value) within Objects, and utilizing the SWE attribute group, *swe:AssociationGroup*, within property elements.

In properties that support XLink components, one can usually choose define that property value inline, as in:

```
<contact>
    < ResponsibleParty id="JoeBlow">
    ....
    </ResponsibleParty>
</contact>
```

Or one can reference an object within the same document:

```
<contact xlink:href="#JoeBlow"/>
```

or an object within an external document:

```
<contact xlink:href="http://www.myCom.com/personel.xml#JoeBlow"/>
```

10.2 SWE Common Data

SWE Common data types is are used throughout SensorML and throughout other SWE encodings and web services. Within SensorML, the *AnyData* group serves as a value for *InputList*, *OutputList*, *ParameterList*, and *PropertyList*. The schema for simple data types is parameter.xsd within the *http://www.opengis:net/swe* namespace.

10.2.1 Simple Data (hard-typing and soft-typing)

In SensorML, both soft-typing and hard-typing of data parameters is supported as indicated in the soft-typed and hard-typed examples:

```
<!—soft-typed -->

 <component name=”focalLength”>
     <Quantity id=”fov” uom="urn:ogc:def:units:ogc:1.0:mm" > 0.1</ Quantity>
 </component>

<!—hard-typed -->

 <focalLength>
     <Quantity id=”fov” uom="urn:ogc:def:units:ogc:1.0:mm" > 0.1</ Quantity>
 </focalLength>
```

The main difference between the two examples is that the first example takes a *xs:qname* within the *name* attribute, while the second example has the *qname* as the element name. The recommended behavior of a parser would be to utilize the *qname* within the *name* attribute if it exist and to use the element name otherwise. It is highly recommended that the hard-typing occur at the property level and not at the object level for simple data types, such as *Quantity*, *Count*, *Category*, or *Boolean*.

All elements of the *AnyScalar* group accept *xs:emptyType*, as well as a *xs:simpleType*, as their value. For example, a *Quantity* can accept either an *xs:double* or nothing as its value:

```
<Quantity id="elevationAngle" fixed="false" uom="#degrees" min="10" max="45"> 22.0 </Quantity>

<Quantity id="elevationAngle" fixed="false" uom="#degrees" min="10" max="45"/>
```

Empty components will typically be targets of links between one or more processes or data sources. Minimum and maximum allowed values for a *Quantity* or *Count* can be defined using the *min* and *max* attributes, respectively. *AnyData* components with the *fixed* attribute set to *true* cannot be altered through links.

10.2.2 Data Aggregates

Most of the properties that utilize *AnyData* within SensorML will group data components in one or more *DataGroup* objects. The main reason for supporting hard-typing within SensorML is to allow one to define reusable data groups using XMLSchema and to explicitly specify which components are expected. *Note: The need for hard-typing in this case is a result of limitations on how one must support rules in XML Schema; XML schematron rules are more flexible and could allow one to specify specific groups while using soft-typing.*

An example of a soft-typed data group:

```
<!—soft-typed DataGroup-->

 <output name="status">
     <swe:DataGroup>
         <swe:component name="hFOM">
             <swe:Time definition="urn:ogc:data:ogc1.0.3:gps:HFOM" uom="urn:ogc:unit:meter"/>
         </swe:component>
         <swe:component name="vVFOM">
             <swe:Quantity definition="urn:ogc:data: ogc:.0.3:gps:VFOM" uom="urn:ogc:unit:meter"/>
         </swe:component>
         <swe:component name="hDOP">
             <swe:Quantity definition="urn:ogc:data:ogc:1.0.3:gps:HDOP" uom="urn:ogc:unit:meter"/>
         </swe:component>
         <swe:component name="vDOP">
             <swe:Quantity definition="urn:ogc:data:ogc:1.0.3:gps:VDOP" uom="urn:ogc:unit:meter"/>
         </swe:component>
         <swe:component name="numSys">
             <swe:Count definition="urn:ogc:data:ogc:1.0.3:gps:NUMSYS"/>
         </swe:component>
         <swe:component name="navMode">
             <swe:Count definition="urn:ogc:data:ogc:1.0.3:gps:NAVMODE"/>
         </swe:component>
     </swe:DataGroup>
```

```
</output>
```

A hard-typed data group will be derived from *_GroupBaseType* by extension and will add appropriate hard-typed data component properties. An example instance from such a data group would appear as:

```
<!—hard-typed DataGroup-->

<output name="status">
    <GpsStatus>
        <hFOM>
            <swe:Time definition="urn:ogc:def:data:ogc:1.0.30:gps:HFOM" uom="urn:ogc:unit:meter"/>
        </hFOM>
        < vFOM>
            <swe:Quantity definition="urn:ogc:data:ogc:1.0.30:gps:VFOM" uom="urn:ogc:unit:meter"/>
        </vFOM>
        <hDOP>
            <swe:Quantity definition="urn:ogc:data:ogc:1.0.30:gps:HDOP" uom="urn:ogc:unit:meter"/>
        </hDOP>
        <vDOP>
            <swe:Quantity definition="urn:ogc:data:ogc:1.0.30:gps:VDOP" uom="urn:ogc:unit:meter"/>
        </vDOP>
        <numSys>
            <swe:Count definition="urn:ogc:data:ogc:1.0.30:gps:NUMSYS"/>
        </ numSys >
        <navMode>
            <swe:Count definition="urn:ogc:data:ogc:1.0.30:gps:navMode"/>
        </navMode >
    </GpsStatus>
</output>
```

One may also support data arrays using *AnyData* group, as in the following example which defines the HRG scan line for the SPOT 5 satellite sensor. This example specifies that there an 3000 member array of pixels for a single scan line and that each element (or pixel) in that array consist of a cluster of 4 components (or bands).

```
<sml:output name="HRG  scanLine">
    <swe:DataArray arraySize="3000">
        <swe:component name="pixel">
            <swe:DataGroup>
                <swe:component name="xs1">
                    <swe:Quantity definition="urn:ogc:data:ogc:1.0.3:DN"/>
                </swe:component>
                <swe:component name="xs2">
                    <swe:Quantity definition="urn:ogc:data:ogc:1.0.3:DN"/>
                </swe:component>
                <swe:component name="xs3">
                    <swe:Quantity definition="urn:ogc:data:ogc:1.0.3:DN"/>
                </swe:component>
                <swe:component name="swir">
                    <swe:Quantity definition="urn:ogc:data:ogc:1.0.3:DN"/>
                </swe:component>
            </swe:DataGroup>
        </swe:component>
    </swe:DataArray>
</sml:output>
```

A *DataGroup* or *DataArray* with no data values for the components, as in the above example, may be used to define linkable *inputs*, *outputs*, or *parameters* in a process, or may be used in the *dataComponent* property of a *DataDefinition* as in this example of navigation data from the NASA Ames AIRDAS sensor system:

```
<swe:Data>
    <swe:definition>
        <swe:DataDefinition>
            <swe:dataComponents>
                <swe:DataGroup>
                    <swe:component name="time">
                        <swe:Time definition="urn:ogc:def:phenomenon:ogc:1.0.3:time"
                            referenceTime="1970-01-01T00:00:00Z" uom="urn:ogc:unit:second"/>
                    </swe:component>
                    <swe:component name="latitude">
                        <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:angle"
                            uom="urn:ogc:def:unit:ogc:1.0:degree"/>
                    </swe:component>
                    <swe:component name="longitude">
                        <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:angle"
                            uom="urn:ogc:def:unit:ogc:1.0:degree"/>
                    </swe:component>
                    <swe:component name="altitude">
                        <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:distance"
                            uom="urn:ogc:def:unit:ogc:1.0:foot"/>
                    </swe:component>
                    <swe:component name="trueHeading">
                        <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:angle"
                            uom="urn:ogc:def:unit:ogc:1.0:degree"/>
                    </swe:component>
                    <swe:component name="pitch">
                        <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:angle"
                            uom="urn:ogc:def:unit:ogc:1.0:degree"/>
                    </swe:component>
                    <swe:component name="roll">
                        <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:angle"
                            uom="urn:ogc:def:unit:ogc:1.0:degree"/>
                    </swe:component>
                </swe:DataGroup>
            </swe:dataComponents>
            <swe:encoding>
                <swe:AsciiBlock decimalSeparator="." tokenSeparator="," tupleSeparator=" "/>
            </swe:encoding>
        </swe:DataDefinition>
    </swe:definition>
    <swe:value>
        1.068088738124E9,32.6018,-116.6153,13790,0.0,7.218017578125,-0.999755859375
        1.068088738124E9,32.6018,-116.6153,13790,0.0,7.174072265625,-1.0986328125
        1.068088738124E9,32.6018,-116.6153,13790,0.0,7.1685791015625,-1.16455078125
        1.068088738124E9,32.6018,-116.6153,13790,0.0,7.152099609375,-1.20849609375
        1.068088738124E9,32.6018,-116.6153,13790,0.0,7.14111328125,-1.219482421875
        1.068088738124E9,32.6018,-116.6153,13790,0.0,7.174072265625,-1.241455078125
        1.068088739124E9,32.6017,-116.6161,13790,0.0,7.174072265625,-1.2469482421875
        1.068088739124E9,32.6017,-116.6161,13790,0.0,7.2235107421875,-1.2469482421875
        1.068088739124E9,32.6017,-116.6161,13790,0.0,7.2235107421875,-1.23046875
        1.068088739124E9,32.6017,-116.6161,13790,0.0,7.2454833984375,-1.1920166015625
        1.068088739124E9,32.6017,-116.6161,13790,0.0,7.283935546875,-1.131591796875
        1.068088739124E9,32.6017,-116.6161,13790,0.0,7.305908203125,-1.087646484375
        1.068088739124E9,32.6017,-116.6161,13790,0.0,7.327880859375,-1.0382080078125
        1.068088739124E9,32.6017,-116.6161,13790,0.0,7.327880859375,-0.955810546875
        1.068088739124E9,32.6017,-116.6161,13790,0.0,7.3333740234375,-0.90087890625
    <swe:value>
</swe:Data>
```

Similarly, the scan line data from the AIRDAS scanner can be encoded in base64:

```
<swe:Data>
    <swe:definition>
        <swe:DataDefinition>
            <swe:dataComponents>
                <swe:DataGroup>
                    <swe:component name="julianTime">
                        <swe:Time definition="urn:ogc:def:phenomenon:ogc:1.0.3:time"
                            referenceTime="1970-01-01T00:00:00Z" uom="second"/>
                    </swe:component>
                    <swe:component name="scanLine">
                        <swe:DataArray>
                            <swe:size>
                                <swe:Count>720</swe:Count>
                            </swe:size>
                            <swe:component name="radiance">
                                <swe:DataGroup>
                                    <swe:component name="band1">
                                        <swe:Quantity definition="urn:ogc:def:data:ogc:1.0.3:DN"/>
                                    </swe:component>
                                    <swe:component name="band2">
                                        <swe:Quantity definition="urn:ogc:def:data:ogc:1.0.3:DN"/>
                                    </swe:component>
                                    <swe:component name="band3">
                                        <swe:Quantity definition="urn:ogc:def:data:ogc:1.0.3:DN"/>
                                    </swe:component>
                                    <swe:component name="band4">
                                        <swe:Quantity definition="urn:ogc:def:data:ogc:1.0.3:DN"/>
                                    </swe:component>
                                </swe:DataGroup>
                            </swe:component>
                        </swe:DataArray>
                    </swe:component>
                </swe:DataGroup>
            </swe:dataComponents>
            <swe:encoding>
                <swe:BinaryBlock>
                    <swe:byteEncoding>base64</swe:byteEncoding>
                    <swe:byteOrder>bigEndian</swe:byteOrder>
                    <swe:structure>
                        <swe:BinaryArray arraySize="720">
                            <swe:BinaryValue bitLength="16" dataType="unsignedInteger"/>
                        </swe:BinaryArray>
                    </swe:structure>
                </swe:BinaryBlock>
            </swe:encoding>
        </swe:DataDefinition>
    </swe:definition>
    <swe:value>
    3BTYRdhb3OLXctMQzkzKrNMJ3YzVSM0szJ3K+NGh0afP8tSv05jUc84gzn3Xfc56zSHLzMlgyyrK
    nsrgyGvHxM1c0qnnZukb5lj5HOug8sXvRufY8tv1E+2W3+Lwb/3PCBPu4OIa7lvgBu85B/H+AOfA
    8EXtg+NZ8HnmUfA8/8fxnupj5CDu0vN76FLvovKo7QYBEgGS/3wJjwlnCO7/L/50/nf2GvDu6P3u
    9+xt5QHh8Oa168Hu4/LH9YL7D/P326rp6e+I7koFufzS7cX0yQxxAK7qpPB98SrvkgikBUT97A+q
    8DHlxuCF2r7qQNpF2SbirNnp09DTNNGUzgXN6dCE0+3T39Ggz1LNbsv6x0/Jd8j4xCbE/8WByvbR
    ac8o0gLSQ9EWzf/QXNYm1nvXM8crxRnAvtb335DTBefF6ifsA+Nt4rrgC9mr3iDaCdo75ADlAOZ4
    5kvYGdcb2S7WGc7pzjfL5tYc4nPpXOQjzdjKfMVIw9vEmMTmzBzPhc54zPnQp9HJ1ljXnNVn0oXT
    odVpyjbJ69BhzWrDSL9nwOm/8MGbwPPBQ8JawuPCg8NuxBbFm8PuwhjJGcoyyTvEasBHxLfID8hH
    zBLU9NUl1pHVidYd35nhDOgh7LDtXfa699bxheHo4o/gbuAp52Xfp/iB/e/0C+mc5JLjcOXz/BXt
    Ct/e3Nbfuuq47UjquPcFHLQJZ+72/pr2S/Jz6SAZ0EayFl787eHP03DesdMW3VnmvuNL7XfnLej9
    5ZrjV+fj4SvgPdr73ILiPPEe/Xv3gfas8E/lyOpU7rEL4A4OE4MJRNcC2JXqQfKE6Cvwa/Bw8gLo
    NuU297rkYeMX3LHaENii1mPdsdXU15rQ0Mse0UvQNM6IzBvQPNWi0+HKeMiDzGfMPdj02ibUxtcj
    … truncated for brevity ….
    </swe:value>
</swe:Data>
```

10.2.3 Curves and Tuples

Curve is derived from *_GroupBaseType* and is part of the *AnyData* group. It allows one to define one or more axes for a curve (taking *AnyScalar* for each *axis* value) and to provide a series of coordinate values using the *tupleValues* properties. An example of a calibration curve mapping temperature to resistance is given below.

```
<swe:Curve arraySize="21">
    <swe:definition>
        <swe:Coordinates>
            <swe:axis name="temperature">
                <swe:Quantity definition="urn:ogc:phenomenon:ogc:1.0.3:temperature"
                    uom="urn:ogc:unit:celsius"/>
            </swe:axis>
            <swe:axis name="resistance">
                <swe:Quantity id="RES" definition="urn:ogc:phenomenon:ogc:1.0.3:resistance"
                    uom="urn:ogc:unit:ohm" scale="1000"/>
            </swe:axis>
        </swe:Coordinates>
    </swe:definition>
    <swe:tupleValues>
        -40,328.4 -35,237.7 -30,173.9 -25,128.5 -20,95.89 -15,72.23 -10,54.89 -5,42.07 0,32.51 5,25.31
        10,19.86 15,15.69 20,12.49 25,10 30,8.06 35,6.536 40,5.331 45,4.373 50,3.606 55,2.989 60,2.49
    </swe:tupleValues>
</swe:Curve>
```

The *definition* property defines two *Quantity* axes (i.e. temperature and resistance). The *tupleValues* property allows one to provide an array of the coordinate values using an efficient data block. Within the *tupleValues* block, are 21 pairs of temperature and resistance values, with white space separating tuples (i.e. the value pair in this case) and commas separating the tokens within a tuple. *Curve* can support axes of different types, for example, to map a *Count* to a *Category* or perhaps an array index (*Count*) to a *Quantity* value.

A *NormalizedCurve* is also defined in the SWE common parameters.xsd schema, which supports allows one to provide bias and gain values for either or both of the input and output axes of the curve.

10.3 Base Process Model

The main purpose of the *ProcessModel* is to provide an atomic executable process description. While *ProcessModel* provides metadata that is useful for discovery and assistance to humans, the properties that are critical for supporting execution of the *ProcessModel* within SensorML-enabled software are the *inputs*, *outputs*, *parameters*, and *method* properties.

The following example is for a Davis 7817 Thermometer used in a weather station. The *inputs* and *outputs* are simple, each taking single *Quantity* component for the component definition … real temperature for input and resistance for output. There are two parameters for the *ProcessModel*: *steadyStateResponse* and *accuracy*. Both parameters use a *NormalizedCurve* temperature to resistance (for the *steadyStateResponse*) and temperature to absolute accuracy (for the *accuracy* response). The *method* for this *ProcessModel* is a general transducer method that has been defined elsewhere and is referenced here using the *xlink:href* association.

```
<ProcessModel xmlns="http://www.opengis.net/sensorML" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xmlns:swe="http://www.opengis.net/swe" xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/sensorML ../base/sensorML.xsd" id="Davis_7817">

… metadata omitted for brevity …

    <!--~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <!--Sensor Coordinate Frame-->
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <referenceFrame>
        <gml:EngineeringCRS id="SENSOR_FRAME">
            <gml:srsName>Sensor Frame</gml:srsName>
            <gml:usesCS xlink:href="urn:ogc:def:crs:ogc:1.0:xyzFrame"/>
            <gml:usesDatum>
                <gml:Datum>
                    <gml:datumName>Sensor Datum</gml:datumName>
                    <gml:anchorPoint>origin is at the tip of the thermometer;
                        x and Y are orthogonal to z but undetermined
                        z is along the long axis of symmetry of the thermometer
                    </gml:anchorPoint>
                </gml:Datum>
            </gml:usesDatum>
        </gml:EngineeringCRS >
    </referenceFrame>
    <!--~~~~~~~~~~~~~~~~-->
    <!--Sensor Inputs-->
    <!--~~~~~~~~~~~~~~~~-->
    <inputs>
        <InputList>
            <input name="temperature">
                <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:temperature"/>
            </input>
        </InputList>
    </inputs>
    <!--~~~~~~~~~~~~~~~~-->
    <!--Sensor Outputs-->
    <!--~~~~~~~~~~~~~~~~-->
    <outputs>
        <OutputList>
            <output name="measuredTemperature">
                <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:resistance"/>
            </output>
        </OutputList>
    </outputs>
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <!--  Temperature Response  -->
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <parameters>
        <ParameterList>
            <steadyStateResponse>
                <NormalizedCurve fixed="true">
                    <function>
                        <swe:Curve arraySize="21">
                            <swe:definition>
                                <swe:Coordinates>
                                    <swe:axis name="temperature">
                                        <swe:Quantity id="TEMP1"
                                            definition="urn:ogc:def:phenomenon:ogc:1.0.3:temperature"
                                            uom="urn:ogc:unit:celsius"/>
                                    </swe:axis>
                                    <swe:axis name="resistance">
                                        <swe:Quantity id="RES"
                                            definition="urn:ogc:def:phenomenon:ogc:1.0.3:resistance"
                                            uom="urn:ogc:unit:ohm" scale="1e3"/>
                                    </swe:axis>
                                </swe:Coordinates>
                            </swe:definition>
```

```
                        <swe:tupleValues>-40,328.4 -35,237.7 -30,173.9 -25,128.5 -20,95.89 -15,72.23
                            -10,54.89 -5,42.07 0,32.51 5,25.31 10,19.86 15,15.69 20,12.49 25,10 30,8.06
                            35,6.536 40,5.331 45,4.373 50,3.606 55,2.989 60,2.49
                        </swe:tupleValues>
                    </swe:Curve>
                </function>
            </NormalizedCurve>
        </steadyStateResponse>
        <accuracy>
            <NormalizedCurve fixed="true">
                <function>
                    <swe:Curve arraySize="6">
                        <swe:definition>
                            <swe:Coordinates>
                                <swe:axis name="temperature">
                                    <swe:Quantity id="TEMP2"
                                        definition="urn:ogc:def:phenomenon:ogc:1.0.3:temperature"
                                        uom="urn:ogc:def:unit:ogc:1.0.3:celsius"/>
                                </swe:axis>
                                <swe:axis name="absoluteError">
                                    <swe:Quantity id="ERR"
                                        definition="urn:ogc:data:quantity:absoluteAccuracy"
                                        uom="urn:ogc:def:unit:ogc:1.0.3:celsius"/>
                                </swe:axis>
                            </swe:Coordinates>
                        </swe:definition>
                        <swe:tupleValues>-40,0.0 -20,0.1 -10,0.2 0,0.3 10,0.4 20,0.5</swe:tupleValues>
                    </swe:Curve>
                </function>
            </NormalizedCurve>
        </accuracy>
    </ParameterList>
  </parameters>
  <method xlink:href="urn:ogc:def:process:ogc:1.0.3:transducer"/>
</ProcessModel>
```

10.4 Composite Process (Process Chain)

Whereas *ProcessModel* defines an atomic process that can be executed using algorithms or code defined by the *method* property, a *Process* defines a process chain where the chain itself defines the execution methodology. *Process* inherits all the properties of the abstract *_ProcessType*, and thus includes *metadataGroup*, *referenceFrame*, *inputs*, *outputs*, and *parameters*.

The steps for defining a *Process* are:

- Define the metadata, *referenceFrame* (if needed), *inputs*, *outputs*, and *parameters* for the process chain itself. This exposes those data components that are accessible and linkable by components external to this process chain
- Define, or reference through *xlink:href*, all of the *processes* that are used inside of this *Process*
- Define all *connections* between process *inputs*, *outputs*, and *parameters* throughout the chain, including the *inputs*, *outputs*, and *parameters* of the *Process* itself

The following example defines a weather station that in this case, measures three phenomenon (temperature, wind speed, and wind direction), outputs the measured values for these phenomenon, as well as a calculated wind chill factor and a maximum temperature alert. The first part of this example defines the metadata (omitted), *inputs*, *outputs*, and *parameters* of the Process (i.e. process chain) itself.

```
<Process xmlns="http://www.opengis.net/sensorML" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:swe="http://www.opengis.net/swe" xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/sensorML ../../../base/Process.xsd" id="Weather_Station">

… metadata omitted for brevity
        <!--~~~~~~~~~~~~~~~~-->
        <!--Process Inputs-->
        <!--~~~~~~~~~~~~~~~~-->
        <inputs>
            <InputList>
                <input name="physicalPhenomena">
                    <swe:DataGroup>
                        <swe:component name="atmosphericTemperature">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:temperature"/>
                        </swe:component>
                        <swe:component name="wind">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:wind"/>
                        </swe:component>
                    </swe:DataGroup>
                </input>
            </InputList>
        </inputs>
        <!--~~~~~~~~~~~~~~~~-->
        <!--Process Outputs-->
        <!--~~~~~~~~~~~~~~~~-->
        <outputs>
            <OutputList>
                <!--~~~~~~~~~~~~~~~~~~~~-->
                <!—weather output cluster -->
                <!--~~~~~~~~~~~~~~~~~~~~-->
                <output name="weatherMeasurements">
                    <swe:DataGroup id="outputDataGroup">
                        <swe:component name="measuredTemperature">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:temperature"
                                uom="urn:ogc:unit:celsius"/>
                        </swe:component>
                        <swe:component name="measuredWindSpeed">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:windSpeed"
                                uom="urn:ogc:unit:metersPerSecond"/>
                        </swe:component>
                        <swe:component name="measuredWindDirection">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:windDirection"
                                uom="urn:ogc:unit:degree"/>
                        </swe:component>
                        <swe:component name="windChill">
                            <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:windChill"
                                uom="urn:ogc:unit:celsius"/>
                        </swe:component>
                    </swe:DataGroup>
                </output>
                <!--~~~~~~~~~~~~~~~~~~~~~~~-->
                <!—temperature alert cluster  -->
                <!--~~~~~~~~~~~~~~~~~~~~~~~-->
                <output name="maximumTemperatureAlert">
                    <swe:DataGroup>
                        <swe:component name="eventTime">
                            <swe:IsoDateTime/>
```

```
                    </swe:component>
                    <swe:component name="measuredValue">
                        <swe:Quantity definition="urn:ogc:def:alert:threshold"/>
                    </swe:component>
                </swe:DataGroup>
            </output>
        </OutputList>
    </outputs>
    <!--~~~~~~~~~~~~~~~~-->
    <!--Process Parameters-->
    <!--~~~~~~~~~~~~~~~~-->
    <parameters>
        <ParameterList>
            <parameter name="maximumTemperatureThreshold">
                <swe:DataGroup>
                    <swe:component name="comparisonCriteria">
                        <swe:Category definition="urn:ogc:def:data:ogc:1.0.3:relationship"
                            fixed="true">greaterThan</swe:Category>
                    </swe:component>
                    <swe:component name="maxThreshold">
                        <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:temperature"/>
                    </swe:component>
                </swe:DataGroup>
            </parameter>
        </ParameterList>
    </parameters>
```

The second part of the example defines the processes that are a part of this process chain. These include three Transducers (thermometer and two components of the anemometer). In the full example, each of these transducer descriptions could include useful metadata, and would also include the response characteristics as part of the parameters, as shown in an earlier example.

In addition to the three transducers, the ProcessList includes two additional processes: one for calculating windchill based on the temperature and wind speed, and one for comparing temperature values against a maximum threshold value.

```
    <!--~~~~~~~~~~~~~~-->
    <!—processes     -->
    <!--~~~~~~~~~~~~~~-->
    <processes>
        <ProcessList>
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <!-- Description of Temperature Transducer -->
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <process name="thermometer">
                <Transducer id="Davis_7817">
                    … details omitted for brevity ……..
                    <inputs>
                        <InputList>
                            <input name="temperature">
                                <swe:Quantity definition="urn:ogc:phenomenon:temperature"/>
                            </input>
                        </InputList>
                    </inputs>
                    <outputs>
                        <OutputList>
                            <output name="measuredTemperature">
                                <swe:Quantity definition="urn:ogc:phenomenon:temperature"/>
                            </output>
                        </OutputList>
```

```
            </outputs>
            <parameters>
                <ParameterList>
                    <steadyStateResponse>
                        … details omitted for brevity …
                    </steadyStateResponse>
                </ParameterList>
            </parameters>
            <method xlink:href="urn:ogc:process:transducer"/>
        </Transducer>
    </process>
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <!-- Description of Wind Speed Transducer -->
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <process name="windSpeedTransducer">
        <Transducer id="windSpd_7911">
        … details omitted for brevity …
            <inputs>
                <InputList>
                    <input name="windSpeed">
                        <swe:Quantity definition="urn:ogc:phenomenon:speed"/>
                    </input>
                </InputList>
            </inputs>
            <outputs>
                <OutputList>
                    <output name="measuredWindSpeed">
                        <swe:Quantity definition="urn:ogc:phenomenon:speed"/>
                    </output>
                </OutputList>
            </outputs>
            <parameters>
                <ParameterList>
                    <steadyStateResponse>
                        … details omitted for brevity …
                    </steadyStateResponse>
                </ParameterList>
            </parameters>
            <method xlink:href="urn:ogc:process:transducer"/>
        </Transducer>
    </process>
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <!-- Description of Wind Direction Transducer -->
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <process name="windDirectionTransducer">
        <Transducer id="windDir_7911">
            … details omitted for brevity …
            <inputs>
                <InputList>
                    <input name="windDirection">
                        <swe:Quantity definition="urn:ogc:phenomenon:direction"/>
                    </input>
                </InputList>
            </inputs>
            <outputs>
                <OutputList>
                    <output name="measuredWindDirection">
                        <swe:Quantity definition="urn:ogc:phenomenon:direction"/>
                    </output>
                </OutputList>
            </outputs>
            <parameters>
                <ParameterList>
                    <steadyStateResponse>
                        … details omitted for brevity …
                    </steadyStateResponse>
                </ParameterList>
```

```
            </parameters>
            <method xlink:href="urn:ogc:process:transducer"/>
        </Transducer>
    </process>
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <!--Threshold Comparison Process  -->
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <process name="maxTempCompare">
        <Process id="ValueCompareProcess">
            <inputs>
                <InputList>
                    <input name="value1">
                        <swe:Quantity/>
                    </input>
                    <input name="value2">
                        <swe:Quantity/>
                    </input>
                </InputList>
            </inputs>
            <outputs>
                <OutputList>
                    <output name="criteriaMet">
                        <swe:Quantity/>
                    </output>
                    <output name="criteriaFailed">
                        <swe:Quantity/>
                    </output>
                </OutputList>
            </outputs>
            <parameters>
                <ParameterList>
                    <parameter name="criteria">
                        <swe:DataGroup>
                            <swe:component name="comparisonCriteria">
                                <swe:Category definition="urn:ogc:relationship"/>
                            </swe:component>
                            <swe:component name="maxThreshold">
                                <swe:Quantity/>
                            </swe:component>
                        </swe:DataGroup>
                    </parameter>
                </ParameterList>
            </parameters>
            <method xlink:href="urn:ogc:def:process:ogc:1.0.3:valueComparison"/>
        </Process>
    </process>
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <!—Wind Chill Process  -->
    <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
    <process name="windChill">
        <Process id="WindChillProcess">
            <inputs>
                <InputList>
                    <input name="temperature">
                        <swe:Quantity uom="urn:ogc:unit:celsius"/>
                    </input>
                    <input name="windSpeed">
                        <swe:Quantity uom="urn:ogc:unit:metersPerSecond"/>
                    </input>
                </InputList>
            </inputs>
            <outputs>
                <OutputList>
                    <output name="windChill">
                        <swe:Quantity uom="urn:ogc:unit:celsius"/>
                    </output>
                </OutputList>
```

```
                    </outputs>
                    <method xlink:href="urn:ogc:process:windChillCalculation"/>
                </Process>
            </process>
        </ProcessList>
    </processes>
```

The last part of the *Process* example defines the *connections* between *inputs*, *outputs*, and *parameters*. The *connection* property uses a *Link* object to reference the *source* and *destination* of a connector.

To reference a particular data component inside of a *Process*, SensorML defines a syntax using property *qnames* as path components starting with at the base of the *Process* instance. Thus all path references will begin with either *inputs*, *outputs*, *parameters*, or *processes,* indicating those properties of the *Process.* A path only uses property names (lowerCamelCase elements) and not Object names. If the property has a value assigned to the *name* attribute, then that will be used in pace of the element name.

For example, a reference to the first input component of the Process would be "*inputs/physicalPhenomenon/temperature*" while the reference to the input of the thermometer would be "*processes/thermometer/inputs/temperature*".

```
    <connections>
        <ConnectionList>
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <!-- process inputs to transducer inputs -->
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <connection name="inputToThermometer">
                <Link>
                    <source ref="inputs/physicalPhenomena/temperature"/>
                    <destination ref="processes/thermometer/inputs/temperature"/>
                </Link>
            </connection>
            <connection name="inputToWindSpeed">
                <Link>
                    <source ref="inputs/physicalPhenomena/wind"/>
                    <destination ref="processes/windSpeedTransducer/inputs/windSpeed"/>
                </Link>
            </connection>
            <connection name="inputToWindDirection">
                <Link>
                    <source ref="inputs/physicalPhenomena/wind"/>
                    <destination ref="processes/windDirectionTransducer/inputs/windDirection"/>
                </Link>
            </connection>
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <!-- transducer outputs to process outputs -->
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <connection name="outputToTemperature">
                <Link>
                    <source ref="processes/thermometer/outputs/measuredTemperature"/>
                    <destination ref="outputs/weatherMeasurement/measuredTemperature"/>
                </Link>
            </connection>
            <connection name="outputToWindSpeed">
                <Link>
                    <source ref="processes/windSpeedTransducer/outputs/measuredWindSpeed"/>
                    <destination ref="outputs/measuredWindSpeed"/>
                </Link>
            </connection>
```

```xml
            <connection name="outputToWindDirection">
               <Link>
                  <source ref="processes/windDirectionTransducer/outputs/measuredWindDirection"/>
                  <destination ref="outputs/measuredWindDirection"/>
               </Link>
            </connection>
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <!-- wind chill inputs and outputs -->
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <connection name="temperatureToWindChill">
               <Link>
                  <source ref="processes/thermometer/outputs/measuredTemperature"/>
                  <destination ref="processes/windChill/inputs/temperature"/>
               </Link>
            </connection>
            <connection name="windSpeedToWindChill">
               <Link>
                  <source ref="processes/windSpeedTransducer/outputs/measuredWindSpeed"/>
                  <destination ref="processes/windChill/inputs/windSpeed"/>
               </Link>
            </connection>
            <connection name="windChillToOutput">
               <Link>
                  <source ref="processes/windChill/outputs/windChill"/>
                  <destination ref="outputs/windChill"/>
               </Link>
            </connection>
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <!-- maxTemp threshold inputs, outputs, and parameters -->
            <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
            <connection name="temperatureToMax">
               <Link>
                  <source ref="processes/thermometer/outputs/measuredTemperature"/>
                  <destination ref="processes/maxTempCompare/inputs/value1"/>
               </Link>
            </connection>
            <connection name="maxTempToAlert">
               <Link>
                  <source ref="processes/maxTempCompare/outputs/criteriaMet"/>
                  <destination ref="outputs/maximumTemperatureAlert/measuredValue"/>
               </Link>
            </connection>
            <connection name="maxTempThreshold">
               <Link>
                  <source ref="parameters/maximumTemperatureThreshold"/>
                  <destination ref="processes/maxTempCompare/parameters/criteria"/>
               </Link>
            </connection>
         </ConnectionList>
      </connections>
```

While *connection* references inside of a *Process* description may be challenging for humans to follow, they are simple for XML-aware software to parse. This syntax should also be easily implemented in future application software that will allow creation of SensorML instances using graphical interfaces.

10.5 System

A *System* is a process chain (derived from *ProcessType*) that includes positional information (spatial and temporal), allowing one to relate a process and its components to the real world. It thus inherits all the properties of the *Process* (as illustrated above), but

also includes an additional *positions* property, which takes a collection of *position* definitions within its *PositionList*. The *position* property takes either a *_PositionData* object that explicitly defines a position, or a *Process* from which a position can be derived (perhaps as a function of time). Such a process should include a *_PositionData* data group as one of its *outputs*.

A *_PositionData* object defines the position (location and orientation) of a local frame (identified by the *localFrame* attribute) to an external reference frame (defined by the *referenceFrame* attribute). The example below provides a location of a weather station to a longitude, latitude, and altitude based on a WGS84 Datum. The example then provides the location of the anemometer's *referenceFrame* relative to the weather station *referenceFrame*.

The *localFrame* and *referenceFrame* attributes both use a URI to reference the appropriate *referenceFrame* definitions. In the example, the station and anemometer reference frames would have both been defined within the appropriate process description, and would have been assigned *id* attribute values of "STATION_FRAME" and "ANEMOMETER_FRAME", respectively.

```
<positions>
    <PositionList>
        <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
        <!-- Position of Station in Lat, Lon, Alt -->
        <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
        <position>
            <swe:Position localFrame="#STATION_FRAME"
                        referenceFrame="urn:ogc:def:crs:EPSG:1.0:4329">
                <swe:location>
                    <swe:Location definition="urn:ogc:phenomenon:location">
                        <swe:coordinate name="x">
                            <swe:Quantity uom="urn:ogc:unit:degree">34.72450</swe:Quantity>
                        </swe:coordinate>
                        <swe:coordinate name="y">
                            <swe:Quantity uom="urn:ogc:unit:degree">-86.94533</swe:Quantity>
                        </swe:coordinate>
                        <swe:coordinate name="z">
                            <swe:Quantity uom="urn:ogc:unit:meter">20.1169</swe:Quantity>
                        </swe:coordinate>
                    </swe:Location>
                </swe:location>
            </swe:Position>
        </position>
        <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
        <!-- Position of Anemometer relative to Station -->
        <!--~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~-->
        <position>
            <swe:Position localFrame="#ANEMOMETER_FRAME"
                        referenceFrame="#STATION_FRAME">
                <swe:location>
                    <swe:Location definition="urn:ogc:phenomenon:location">
                        <swe:coordinate name="x">
                            <swe:Quantity uom="urn:ogc:unit:degree">0</swe:Quantity>
                        </swe:coordinate>
                        <swe:coordinate name="y">
                            <swe:Quantity uom="urn:ogc:unit:degree">0</swe:Quantity>
                        </swe:coordinate>
                        <swe:coordinate name="z">
                            <swe:Quantity uom="urn:ogc:unit:meter">3.0</swe:Quantity>
                        </swe:coordinate>
```

```
                    </swe:Location>
                </swe:location>
            </swe:Position>
        </position>
    </PositionList>
</positions>
</System>
```

10.6 Metadata Group

10.6.1 Identification and Classification

The *identifier* and *classifier* properties provide both human and software readable names, types, and applications of resources. Both take a *Term* which has an optional *qualifier* attribute that further refines the meaning of the token value.

A *qualifier* value for an *identifier* might include, for example, a shortName for use perhaps in menus and data trees, a longName, missionID, or modelNumber. Examples of *qualifier* attribute values for c*lassifier* might include sensorType, application, or phenomenon. The *identification* and *classification* properties provide relevant information that can be mined to support asset discovery and cataloging.

For a Davis thermometer, an example instance “snippet” that includes identifiers and classifiers might appear as:

```
<identification>
    <IdentifierList>
        <identifier name="longName">
            <Term>Davis 7817 External Temperature Sensor</Term>
        </identifier>
        <identifier name="shortName">
            <Term>Davis 7817 Thermometer</Term>
        </identifier>
        <identifier name="modelNumber">
            <Term qualifier="urn:ogc:def:ogc:identifier:modelNumber">7817</Term>
        </identifier>
        <identifier name="manufacturer">
            <Term qualifier="urn:ogc:def:ogc:identifier:manufacturer">Davis Instruments</Term>
        </identifier>
    </IdentifierList>
</identification>
<classification>
    <ClassifierList>
        <classifier name="intendedApplication">
            <Term qualifier="urn: ogc:def:ogc:v1.0:classifier:application">weather</Term>
        </classifier>
        <classifier name="sensorType">
            <Term qualifier="urn: ogc:def:ogc:v1.0:classifier:sensorType">thermometer</Term>
        </classifier>
        <classifier name="phenomenon">
            <Term qualifier="urn:ogc:def:ogc:v1.0:phenomenon">temperature</Term>
        </classifier>
    </ClassifierList>
</classification>
```

10.6.2 Characteristics and Capabilities

The *characteristics* and *capabilities* properties provide information that is useful for discovery and for assistance to humans. Both take a *PropertyList*, which includes one or more *property* components which take *AnyData* as their value. Often this will be a DataGroup, either soft-typed or hard-typed. A few common capabilities and characteristics DataGroups have been derived in SensorML (commonProperties.xsd), including MeasurementCharacteristics, PhysicalProperties, and Limits.

The following example illustrates the use of these three hard-typed property groups as well as a soft-typed group for interface characteristics.

```
<capabilities>
    <PropertyList>
        <property name="measurementProperties">
            <MeasurementCapabilities>
                <measureResolution>
                    <swe:Quantity definition="urn:ogc:def:phenomenonogc:1.0.3:temperature"
                        uom="urn:ogc:def:unit:ogc:1.0:degreeCelsius"> 0.1 </swe:Quantity>
                </measureResolution>
                <dynamicRange>
                    <swe:QuantityRange definition="urn:ogc:def:phenomenonogc:1.0.3:temperature "
                        uom=" urn:ogc:def:unit:ogc:1.0:degreeCelsius "> -45 60 </swe:QuantityRange>
                </dynamicRange>
                <accuracy>
                    <swe:QuantityRange definition="urn:ogc:def:data:ogc:1.0.3:accuracy"
                        uom="urn:ogc:def:unit:ogc:1.0:percent"> -0.5 0.5 </swe:QuantityRange>
                </accuracy>
            </MeasurementCapabilities>
        </property>
        <property name="survivableRange">
            <Limits definition="urn:ogc:def:property:ogc:1.0.3:survivableLimits">
                <limit name="windSpeedLimits">
                    <swe:QuantityRange definition="urn:ogc:def:phenomenon:ogc:1.0.2:windSpeed"
                        uom="urn:ogc:unit:metersPerSecond">0 175</swe:QuantityRange>
                </limit>
            </Limits>
        </property>
    </PropertyList>
</capabilities>
<characteristics>
    <PropertyList>
        <property name="physicalProperties">
            <PhysicalProperties>
                <mass>
                    <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:mass"
                        uom="urn:ogc:def:unit:ogc:1.0.3:gram"> 128 </swe:Quantity>
                </mass>
                <length>
                    <swe:Quantity definition="urn:ogc:phenomenon:length"
                        uom="urn:ogc:def:unit:ogc:1.0.3:mm">  32 </swe:Quantity>
                </length>
                <diameter>
                    <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:diameter"
                        uom="urn:ogc:unit:mm"> 6.5 </swe:Quantity>
                </diameter>
            </PhysicalProperties>
        </property>
        <property name="interface">
            <swe:DataGroup definition="urn:ogc:property:electricalInterface">
                <swe:component name="cableLength">
                    <swe:Quantity definition="urn:ogc:def:phenomenon:ogc:1.0.3:length"
                        uom="urn:ogc:def:unit:ogc:1.0.3:mm"> 7.6 </swe:Quantity>
                </swe:component>
```

```
                    <swe:component name="connectorType">
                        <swe:Category definition="urn:ogc:data:category:connector:electrical">
                            RJ-11 </swe:Category>
                    </swe:component>
                    <swe:component name="cableType">
                        <swe:Category definition="urn:ogc:data:category:cable:electrical">
                            26-AWG:4-conductor </swe:Category>
                    </swe:component>
                </swe:DataGroup>
            </property>
        </PropertyList>
    </characteristics>
```

11 Future Directions and Remaining Issues

It is anticipated that there will be both minor refinements and additions to the existing schema in future versions and as part of SensorML extensions. Furthermore, efforts are underway to implement software capable of parsing SensorML, mining information for catalogs, processing and geolocating observation data based on SensorML instances, displaying SensorML instances in a human-friendly viewer, and creating SensorML instances using graphical interfaces. These will provide lessons learned that will be used to refine the schemas of SensorML.

Below is a limited list of recognized desires and needs in future releases.

1. Establish authoritative glossaries and schema repositories for process and sensor-related terms.
2. Begin development of common ProcessModel definitions complete with method descriptions and software code.
3. Finish development standard sensor models for geolocation of observations from remote sensors. Support both rigorous and polynomial models. Consider sensor models from ISO 19130 project.
4. Investigate the application of MathML for SensorML method descriptions and execution.
5. Investigate avenues and implement solutions for increased synergy between SensorML and GML. This will probably involve some modifications to both SensorML and GML.
6. Provide mapping between SensorML and IEEE P1451.
7. Finish blending efforts between TransducerML and SensorML.
8. Investigate and implement the use of SensorML process descriptions within industry-standard distributed processing protocols (e.g. BPEL).

Annex A. XML Schemas for SensorML (normative)

sweImports.xsd

SensorML imports several schema from SWE Common.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Alexandre Robin (University of Alabama in Huntsville) -->
<xs:schema targetNamespace="http://www.opengis.net/swe" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:include schemaLocation="../../../sweCommon/1.0.30/parameters.xsd"/>
    <xs:include schemaLocation="../../../sweCommon/1.0.30/positionData.xsd"/>
    <xs:include schemaLocation="../../../sweCommon/1.0.30/data.xsd"/>
</xs:schema>
```

base.xsd.

The base.xsd schema provides the basic definitions and abstract elements used by several SensorML schema. It also defines the metadata groups and components.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Alexandre Robin (University of Alabama in Huntsville) -->
<xs:schema targetNamespace="http://www.opengis.net/sensorML" xmlns:ism="urn:us:gov:ic:ism:v2"
xmlns:sml="http://www.opengis.net/sensorML" xmlns:swe="http://www.opengis.net/swe"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>Base class definitions for core SensorML</xs:documentation>
    </xs:annotation>
    <!--========================-->
    <!--  Includes and Imports  -->
    <!--========================-->
    <!-- import swe common schema -->
    <xs:import namespace="http://www.opengis.net/swe" schemaLocation="./sweImports.xsd"/>
    <!-- import US Intelligence Community schema for security speciifications-->
    <xs:import namespace="urn:us:gov:ic:ism:v2" schemaLocation="../../../ic/2.0/IC-ISM-v2.xsd"/>
    <!--==================-->
    <!--  Global Groups  -->
    <!--==================-->
    <xs:group name="constraints">
        <xs:sequence>
            <xs:element name="securityConstraint" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sml:Security"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element ref="sml:validTime" minOccurs="0"/>
            <xs:element name="legalConstraint" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sml:Rights"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
```

```
</xs:group>
<xs:group name="generalInfo">
    <xs:sequence>
        <xs:element ref="sml:identification" maxOccurs="unbounded"/>
        <xs:element ref="sml:classification" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:group name="references">
    <xs:sequence>
        <xs:element ref="sml:contact" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="sml:documentation" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
</xs:group>
<xs:group name="history">
    <xs:sequence>
        <xs:element name="history">
            <xs:complexType>
                <xs:sequence minOccurs="0">
                    <xs:element ref="sml:EventList"/>
                </xs:sequence>
                <xs:attributeGroup ref="swe:AssociationAttributes"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:group>
<xs:group name="properties">
    <xs:sequence>
        <xs:element ref="sml:capabilities" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element ref="sml:characteristics" minOccurs="0" maxOccurs="unbounded"/>
        <!--<xs:element ref="sml:taskableProperties" minOccurs="0" maxOccurs="unbounded"/>-->
    </xs:sequence>
</xs:group>
<xs:group name="contact">
    <xs:sequence>
        <xs:element name="contact" maxOccurs="unbounded">
            <xs:complexType>
                <xs:choice>
                    <xs:element ref="sml:Person"/>
                    <xs:element ref="sml:ResponsibleParty"/>
                </xs:choice>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:group>
<!--==============================-->
<!--  Global Concrete Properties  -->
<!--==============================-->
<xs:element name="description">
    <xs:annotation>
        <xs:documentation>Textual description of the object</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence minOccurs="0">
            <xs:element name="Discussion" type="swe:TextType"/>
        </xs:sequence>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
    </xs:complexType>
</xs:element>
<xs:element name="identification">
    <xs:annotation>
        <xs:documentation>Mean of providing various identity and alias values, with types such as
        "longName", "abbreviation", "modelNumber", "serialNumber", whose terms can be defined
        in a dictionary</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
```

```
            <xs:element name="IdentifierList">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="identifier" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="sml:Term"/>
                                </xs:sequence>
                                <xs:attribute name="name" type="swe:qnameSimpleType" use="optional"/>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="classification">
    <xs:annotation>
        <xs:documentation>Mean of specifying classification values with types such as "sensorType",
        "intendedApplication", etc., whose terms can be defined in a dictionary</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:element name="ClassifierList">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="classifier" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="sml:Term"/>
                                </xs:sequence>
                                <xs:attribute name="name" type="swe:qnameSimpleType" use="optional"/>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="contact">
    <xs:annotation>
        <xs:documentation>Relevant contacts for that object</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:choice minOccurs="0">
            <xs:element ref="sml:ContactList"/>
            <xs:group ref="sml:contact"/>
        </xs:choice>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
        <xs:attribute name="role" type="xs:anyURI" use="required"/>
    </xs:complexType>
</xs:element>
<xs:element name="documentation">
    <xs:annotation>
        <xs:documentation>Relevant documentation for that object</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:choice minOccurs="0">
            <xs:element ref="sml:DocumentList"/>
            <xs:element ref="sml:Document"/>
        </xs:choice>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
        <xs:attribute name="role" type="xs:anyURI" use="required"/>
    </xs:complexType>
</xs:element>
```

```
<xs:element name="capabilities">
    <xs:annotation>
        <xs:documentation>Capability list for quick discovery</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence minOccurs="0">
            <xs:element ref="sml:PropertyList"/>
        </xs:sequence>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
    </xs:complexType>
</xs:element>
<xs:element name="characteristics">
    <xs:annotation>
        <xs:documentation>Characteristic list for quick discovery</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence minOccurs="0">
            <xs:element ref="sml:PropertyList"/>
        </xs:sequence>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
    </xs:complexType>
</xs:element>
<!--============================-->
<!--  Global Concrete Objects  -->
<!--============================-->
<xs:element name="PropertyList">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="sml:description" minOccurs="0"/>
            <xs:element name="property" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:group ref="swe:AnyData" minOccurs="0"/>
                    <xs:attribute name="name" type="swe:qnameSimpleType" use="optional"/>
                    <xs:attributeGroup ref="swe:AssociationAttributes"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:complexType>
</xs:element>
<xs:element name="Document">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="sml:description"/>
            <xs:element name="date" type="swe:IsoDateTimeType" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Date of creation</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element ref="sml:contact" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Person who created the document</xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="format" type="swe:CategoryType" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Specifies the fornat of the file pointed to by location
                    </xs:documentation>
                </xs:annotation>
            </xs:element>
            <xs:element name="fileLocation">
                <xs:annotation>
                    <xs:documentation>Points to the actual document corresponding to that version
                     </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:attributeGroup ref="swe:AssociationAttributes"/>
```

```
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="optional"/>
            <xs:attribute name="version" type="xs:token" use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="DocumentList">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:description" minOccurs="0"/>
                <xs:element name="member" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence minOccurs="0">
                            <xs:element ref="sml:Document"/>
                        </xs:sequence>
                        <xs:attribute name="name" type="swe:qnameSimpleType" use="required"/>
                        <xs:attributeGroup ref="swe:AssociationAttributes"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Event">
        <xs:complexType>
            <xs:sequence>
                <xs:group ref="sml:generalInfo"/>
                <xs:group ref="sml:references"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="EventList">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="member" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:Event"/>
                        </xs:sequence>
                        <xs:attribute name="name" type="swe:qnameSimpleType" use="required"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="optional"/>
        </xs:complexType>
    </xs:element>
    <!--============================-->
    <!-- Contact Info  -->
    <!--============================-->
    <xs:element name="Person">
        <xs:annotation>
            <xs:documentation>based on IC:DMMS</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="surname" type="xs:token"/>
                <xs:element name="name" type="xs:token"/>
                <xs:element name="userID" type="xs:token"/>
                <xs:element name="affiliation" type="xs:token"/>
                <xs:element name="phoneNumber" type="xs:token"/>
                <xs:element name="email" type="xs:token"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="ContactList">
```

```
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:description" minOccurs="0"/>
                <xs:element name="member" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:group ref="sml:contact"/>
                        <xs:attributeGroup ref="swe:AssociationAttributes"/>
                        <xs:attribute name="role" type="xs:anyURI" use="optional"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="ResponsibleParty" type="sml:ResponsiblePartyType">
        <xs:annotation>
            <xs:documentation>based on ISO 19115</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="ResponsiblePartyType">
        <xs:sequence>
            <xs:element name="individualName" type="xs:string" minOccurs="0"/>
            <xs:element name="organizationName" type="xs:string" minOccurs="0"/>
            <xs:element name="positionName" type="xs:string" minOccurs="0"/>
            <xs:element ref="sml:contactInfo" minOccurs="0"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:complexType>
    <xs:element name="contactInfo">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="phone" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="voice" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
                            <xs:element name="facsimile" type="xs:string" minOccurs="0"
                                maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="address" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="deliveryPoint" type="xs:string" minOccurs="0"
                                maxOccurs="unbounded"/>
                            <xs:element name="city" type="xs:string" minOccurs="0"/>
                            <xs:element name="administrativeArea" type="xs:string" minOccurs="0"/>
                            <xs:element name="postalCode" type="xs:string" minOccurs="0"/>
                            <xs:element name="country" type="xs:string" minOccurs="0"/>
                            <xs:element name="electronicMailAddress" type="xs:string" minOccurs="0"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="onlineResource" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:attribute name="reference" type="xs:anyURI" use="required"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="hoursOfService" type="xs:string" minOccurs="0"/>
                <xs:element name="contactInstructions" type="xs:string" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <!--============================-->
    <!-- Constraints  -->
    <!--============================-->
    <xs:element name="Security">
```

```
        <xs:annotation>
            <xs:documentation>based on IC:ISM definition</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:attributeGroup ref="ism:SecurityAttributesOptionGroup"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="validTime">
        <xs:annotation>
            <xs:documentation>Time validity constraint of description</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:choice>
                <xs:sequence>
                    <xs:element name="StartTime" type="swe:IsoDateTimeType"/>
                    <xs:element name="EndTime" type="swe:IsoDateTimeType"/>
                </xs:sequence>
                <xs:element name="TimeStamp" type="swe:IsoDateTimeType"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
    <xs:element name="Rights">
        <xs:annotation>
            <xs:documentation>based on IC:DDMS definition</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:documentation"/>
            </xs:sequence>
            <xs:attribute name="privacyAct" type="xs:boolean" use="optional"/>
            <xs:attribute name="intellectualPropertyRights" type="xs:boolean" use="optional"/>
            <xs:attribute name="copyRights" type="xs:boolean" use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Term">
        <xs:annotation>
            <xs:documentation>A simple token identifying a term (single spaces allowed)</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:token">
                    <xs:attribute name="qualifier" type="xs:anyURI"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:attributeGroup name="securityAttributes">
        <xs:attribute name="classification" type="xs:token" use="required"/>
        <xs:attribute name="ownerProducer" type="xs:token" use="required"/>
        <xs:attribute name="classificationReason" type="xs:token" use="optional"/>
        <xs:attribute name="classifiedBy" type="xs:token" use="optional"/>
        <xs:attribute name="declassDate" type="xs:date" use="optional"/>
        <xs:attribute name="SCIcontrols" type="xs:token" use="required"/>
        <xs:attribute name="disseminationControls" type="xs:token" use="optional"/>
        <xs:attribute name="FGIsourceOpen" type="xs:token" use="optional"/>
        <xs:attribute name="releaableTo" type="xs:token" use="required"/>
    </xs:attributeGroup>
</xs:schema>
```

commonProperties.xsd.

The commonProperties.xsd schema provides definitions for hard-typed data groups useful for capabilities and characteristics properties.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Alexandre Robin (University of Alabama in Huntsville) -->
<xs:schema targetNamespace="http://www.opengis.net/sensorML" xmlns:ism="urn:us:gov:ic:ism:v2"
xmlns:sml="http://www.opengis.net/sensorML" xmlns:swe="http://www.opengis.net/swe"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>Defines common properties that can be used in Process capabilities or characteristics
        </xs:documentation>
    </xs:annotation>
    <!--========================-->
    <!--  Includes and Imports  -->
    <!--========================-->
    <!-- import swe common schema -->
    <xs:import namespace="http://www.opengis.net/swe" schemaLocation="./sweImports.xsd"/>
    <!--=================-->
    <!--  Global property groups-->
    <!--=================-->
    <xs:element name="Limits" substitutionGroup="swe:_DataGroup">
        <xs:annotation>
            <xs:documentation>Defines environmental limits, such as operational limits or survivable limits (should
                be specified in "definition" attribute for the "Limit" element); the limiting phenomenon (e.g.
                temperature, altitude, etc.) should be specified in the "definition" attribute of the Quantity, Count,
                Category scalar); values of limits can be Quantity or Count ranges (e.g. min-max altitude) or
                Category values (e.g. clear sky)</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="swe:_GroupBaseType">
                    <xs:sequence>
                        <xs:element name="limit" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:choice>
                                    <xs:element ref="swe:QuantityRange"/>
                                    <xs:element ref="swe:CountRange"/>
                                    <xs:element ref="swe:Category"/>
                                    <xs:element ref="swe:ConditionalValue"/>
                                </xs:choice>
                                <xs:attribute name="name" type="swe:qnameSimpleType" use="required"/>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="PhysicalProperties" substitutionGroup="swe:_DataGroup">
        <xs:annotation>
            <xs:documentation>Defines physical characteristics such as dimensions, mass, and material
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="swe:_GroupBaseType">
                    <xs:sequence>
                        <xs:element name="mass" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="swe:Quantity"/>
                                </xs:sequence>
```

```
                    </xs:complexType>
                </xs:element>
                <xs:element name="length" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:Quantity"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="width" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:Quantity"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="height" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:Quantity"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="diameter" minOccurs="0" maxOccurs="2">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:Quantity"/>
                        </xs:sequence>
                        <xs:attribute name="name" type="swe:qnameSimpleType" use="optional"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="material" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:Category"/>
                        </xs:sequence>
                        <xs:attribute name="name" type="swe:qnameSimpleType" use="optional"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="generalShape" use="optional">
                <xs:simpleType>
                    <xs:restriction base="xs:string">
                        <xs:enumeration value="box"/>
                        <xs:enumeration value="sphere"/>
                        <xs:enumeration value="cylinder"/>
                        <xs:enumeration value="ellipse"/>
                        <xs:enumeration value="sheet"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="MeasurementCapabilities" substitutionGroup="swe:_DataGroup">
    <xs:annotation>
        <xs:documentation>Properties useful for discovery of measurement process capabilities
        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="swe:_GroupBaseType">
                <xs:sequence>
                    <xs:element name="measureResolution" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>the smallest meaningful change of a value capable of being
```

```
                output by a  process; can use min-max attributes of Quantity to provide
                reasonable bounds
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:choice>
                <xs:element ref="swe:Quantity"/>
                <xs:element ref="swe:ConditionalValue"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
    <xs:element name="spatialResolution" minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
            <xs:documentation>the mean spatial resolution within some spatial reference
                frame (specified in SpatialDimensions element); can use min-max attributes of
                Quantity to provide reasonable bounds</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="SpatialDimensions">
                    <xs:complexType>
                        <xs:complexContent>
                            <xs:extension base="swe:_GroupBaseType">
                                <xs:sequence>
                                    <xs:element name="condition" minOccurs="0">
                                        <xs:complexType>
                                            <xs:group ref="swe:AnyScalar"/>
                                            <xs:attribute name="name"
                                                type="swe:qnameSimpleType"
                                                use="optional"/>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element name="meanDistance" maxOccurs="3">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element ref="swe:Quantity"/>
                                            </xs:sequence>
                                            <xs:attribute name="name"
                                                type="swe:qnameSimpleType"
                                                use="optional"/>
                                        </xs:complexType>
                                    </xs:element>
                                </xs:sequence>
                                <xs:attribute name="referenceFrame" type="xs:anyURI"
                                    use="optional"/>
                            </xs:extension>
                        </xs:complexContent>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="name" type="swe:qnameSimpleType" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="temporalResolution" minOccurs="0">
        <xs:annotation>
            <xs:documentation>the mean frequency or period of sampling; can use min-max
                attributes of Quantity to provide reasonable bounds</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:choice>
                <xs:element ref="swe:Quantity"/>
                <xs:element ref="swe:ConditionalValue"/>
            </xs:choice>
        </xs:complexType>
    </xs:element>
    <xs:element name="dynamicRange" minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
```

```
                        <xs:documentation>the range of values over which a process can meaningfully
                            measure </xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:choice>
                            <xs:element ref="swe:QuantityRange"/>
                            <xs:element ref="swe:ConditionalValue"/>
                        </xs:choice>
                    </xs:complexType>
                </xs:element>
                <xs:element name="frequencyRange" minOccurs="0" maxOccurs="unbounded">
                    <xs:annotation>
                        <xs:documentation>range of frequencies over which a measure can be taken (e.g.
                            spectral or electromagnetic frequency)</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:choice>
                            <xs:element ref="swe:QuantityRange"/>
                            <xs:element ref="swe:ConditionalValue"/>
                        </xs:choice>
                    </xs:complexType>
                </xs:element>
                <xs:element name="lookAngle" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:choice>
                            <xs:element ref="swe:QuantityRange"/>
                            <xs:element ref="swe:ConditionalValue"/>
                        </xs:choice>
                    </xs:complexType>
                </xs:element>
                <xs:element name="lookRange" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:choice>
                            <xs:element ref="swe:QuantityRange"/>
                            <xs:element ref="swe:ConditionalValue"/>
                        </xs:choice>
                    </xs:complexType>
                </xs:element>
                <xs:element name="accuracy" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:QuantityRange"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</xs:element>
<!--could add properties such as electrical interface-->
</xs:schema>
```

process.xsd.

The process.xsd schema provides definitions for base process types, as well as concrete definitions for Process and ProcessModel.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Alexandre Robin (University of Alabama in Huntsville) -->
<xs:schema targetNamespace="http://www.opengis.net/sensorML"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:sml="http://www.opengis.net/sensorML"
xmlns:swe="http://www.opengis.net/swe" xmlns:om="http://www.opengis.net/om" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>Defines Basic Process Elements and Types for SensorML</xs:documentation>
    </xs:annotation>
    <!--
        NOTE: we indent to support switch between process using special ProcessModels rather than a special
            Link because it provides more flexibility (but is more verbose)
    -->
    <!--========================-->
    <!--  Includes and Imports  -->
    <!--========================-->
    <xs:include schemaLocation="./base.xsd"/>
    <xs:include schemaLocation="./commonProperties.xsd"/>
    <xs:include schemaLocation="./coordinateSystem.xsd"/>
    <xs:import namespace="http://www.opengis.net/swe" schemaLocation="./sweImports.xsd"/>
    <xs:import namespace="http://www.opengis.net/om" schemaLocation="./omImports.xsd"/>
    <!--========================-->
    <!--  Substitution groups  -->
    <!--========================-->
    <xs:element name="_Process" type="sml:_AbstractProcessType" abstract="true">
        <xs:annotation>
            <xs:documentation>base substitution group for all processes</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!--==================-->
    <!--  Global objects  -->
    <!--==================-->
    <xs:element name="ProcessModel" type="sml:ProcessModelType" substitutionGroup="sml:_Process">
        <xs:annotation>
            <xs:documentation>simple atomic process</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Process" type="sml:ProcessType" substitutionGroup="sml:_Process">
        <xs:annotation>
            <xs:documentation>Process formed by chaining sub-processes (i.e. process chain)
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Link">
        <xs:annotation>
            <xs:documentation>Link object used to make connections between processes</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="source">
                    <xs:complexType>
                        <xs:attribute name="ref" type="sml:linkRef" use="required"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="destination">
                    <xs:complexType>
                        <xs:attribute name="ref" type="sml:linkRef" use="required"/>
```

```
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="type" type="xs:anyURI" use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="ArrayLink">
        <xs:annotation>
            <xs:documentation>Special Link to handle accessing array elements sequentially</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:sequence minOccurs="0">
                    <xs:element name="sourceArray">
                        <xs:complexType>
                            <xs:attribute name="ref" type="sml:linkRef"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="indexDestination" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:attribute name="ref" type="sml:linkRef"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:sequence minOccurs="0">
                    <xs:element name="destinationArray">
                        <xs:complexType>
                            <xs:attribute name="ref" type="sml:linkRef"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="indexSource" minOccurs="0">
                        <xs:complexType>
                            <xs:attribute name="ref" type="sml:linkRef"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:element ref="sml:connection" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:complexType>
    </xs:element>
    <xs:element name="Dangle">
        <xs:annotation>
            <xs:documentation>Special Link that allows a probe to access values that are not true outputs of a
                process; one connects a probe via the Dangle ID</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="source">
                    <xs:complexType>
                        <xs:attribute name="ref" type="xs:anyURI" use="required"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="required"/>
            <xs:attribute name="name" type="swe:nameSimpleType" use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Data">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="definition">
                    <xs:complexType>
                        <xs:sequence minOccurs="0">
                            <xs:element ref="swe:DataDefinition"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="swe:AssociationAttributes"/>
                    </xs:complexType>
```

```
            </xs:element>
            <xs:element name="value" type="swe:DataValueType"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:complexType>
</xs:element>
<!--==================-->
<!--  Complex Types  -->
<!--==================-->
<xs:complexType name="_AbstractSMLType" abstract="true">
    <xs:annotation>
        <xs:documentation>Main Abstract SensorML Object</xs:documentation>
    </xs:annotation>
    <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>
<xs:complexType name="_AbstractListType">
    <xs:sequence>
        <xs:any namespace="##any" processContents="strict" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>
<xs:complexType name="_AbstractProcessType">
    <xs:complexContent>
        <xs:extension base="sml:_AbstractSMLType">
            <xs:sequence>
                <xs:group ref="sml:metadataGroup"/>
                <xs:element ref="sml:referenceFrame" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="inputs" type="sml:inputsType" minOccurs="0"/>
                <xs:element name="outputs" type="sml:outputsType" minOccurs="0"/>
                <xs:element name="parameters" type="sml:parametersType" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="_ProcessModelType">
    <xs:complexContent>
        <xs:extension base="sml:_AbstractProcessType">
            <xs:sequence>
                <xs:element name="method" type="sml:methodType"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="_ProcessType" abstract="true">
    <xs:annotation>
        <xs:documentation>Base element from which all processes should derive</xs:documentation>
        <xs:documentation>Base element from which all processes should derive</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:extension base="sml:_AbstractProcessType">
            <xs:sequence>
                <xs:element ref="sml:processes" minOccurs="0"/>
                <xs:element ref="sml:dataSources" minOccurs="0"/>
                <xs:element ref="sml:connections" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="ProcessModelType">
    <xs:annotation>
        <xs:documentation>Complex Type for Simple Process</xs:documentation>
        <xs:documentation>Complex Type for Simple Process</xs:documentation>
    </xs:annotation>
    <xs:complexContent>
        <xs:restriction base="sml:_ProcessModelType">
            <xs:sequence>
                <xs:group ref="sml:metadataGroup"/>
```

```
                    <xs:element ref="sml:referenceFrame" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:inputs" minOccurs="0"/>
                    <xs:element ref="sml:outputs" minOccurs="0"/>
                    <xs:element ref="sml:parameters" minOccurs="0"/>
                    <xs:element ref="sml:method"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ProcessType">
        <xs:annotation>
            <xs:documentation>Complex Type for Process Chains</xs:documentation>
            <xs:documentation>Complex Type for Process Chains</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:restriction base="sml:_ProcessType">
                <xs:sequence>
                    <xs:group ref="sml:metadataGroup"/>
                    <xs:element ref="sml:referenceFrame" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:inputs" minOccurs="0"/>
                    <xs:element ref="sml:outputs" minOccurs="0"/>
                    <xs:element ref="sml:parameters" minOccurs="0"/>
                    <xs:element ref="sml:processes" minOccurs="0"/>
                    <xs:element ref="sml:dataSources" minOccurs="0"/>
                    <xs:element ref="sml:connections" minOccurs="0"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ProcessMethodType">
        <xs:annotation>
            <xs:documentation>Method Definition Type (Should it be a dictionary entry?)</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element name="algorithm" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>Textual description of the algorithm</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:simpleContent>
                        <xs:extension base="xs:string">
                            <xs:attributeGroup ref="swe:AssociationAttributes"/>
                        </xs:extension>
                    </xs:simpleContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="implementation" minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                    <xs:documentation>Points to the reference implementation of this process in the specified
                        programming language</xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:attribute name="uri" type="xs:anyURI" use="required"/>
                    <xs:attribute name="language" type="xs:anyURI" use="required"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="processChain" minOccurs="0">
                <xs:annotation>
                    <xs:documentation>method itself can be a process chain defined by a System
                    </xs:documentation>
                </xs:annotation>
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sml:Process"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="swe:AssociationAttributes"/>
                </xs:complexType>
```

```
            </xs:element>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:complexType>
    <xs:complexType name="methodType">
        <xs:sequence minOccurs="0">
            <xs:element name="ProcessMethod" type="sml:ProcessMethodType">
                <xs:annotation>
                    <xs:documentation>Method describing the process (text or MathML)</xs:documentation>
                </xs:annotation>
            </xs:element>
        </xs:sequence>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
    </xs:complexType>
    <!--================-->
    <!--  Simple Types  -->
    <!--================-->
    <xs:simpleType name="linkRef">
        <xs:restriction base="xs:token"/>
    </xs:simpleType>
    <!--======================-->
    <!--  Global properties  -->
    <!--======================-->
    <xs:element name="inputs">
        <xs:annotation>
            <xs:documentation>list of input signals</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="sml:inputsType">
                    <xs:sequence minOccurs="0">
                        <xs:element name="InputList">
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:restriction base="sml:_AbstractListType">
                                        <xs:sequence>
                                            <xs:element name="input" maxOccurs="unbounded">
                                                <xs:complexType>
                                                    <xs:group ref="swe:AnyData" minOccurs="0"/>
                                                    <xs:attribute name="name"
                                                        type="swe:qnameSimpleType" use="required"/>
                                                    <xs:attributeGroup ref="swe:AssociationAttributes"/>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:restriction>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="outputs">
        <xs:annotation>
            <xs:documentation>list of output signals</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="sml:outputsType">
                    <xs:sequence minOccurs="0">
                        <xs:element name="OutputList">
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:restriction base="sml:_AbstractListType">
                                        <xs:sequence>
```

```
                                        <xs:element name="output" maxOccurs="unbounded">
                                            <xs:complexType>
                                                <xs:group ref="swe:AnyData" minOccurs="0"/>
                                                <xs:attribute name="name"
                                                    type="swe:qnameSimpleType" use="required"/>
                                                <xs:attributeGroup ref="swe:AssociationAttributes"/>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:restriction>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="parameters">
    <xs:annotation>
        <xs:documentation>list of parameters</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:restriction base="sml:parametersType">
                <xs:sequence minOccurs="0">
                    <xs:element name="ParameterList">
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:restriction base="sml:_AbstractListType">
                                    <xs:sequence>
                                        <xs:element name="parameter" maxOccurs="unbounded">
                                            <xs:complexType>
                                                <xs:group ref="swe:AnyData" minOccurs="0"/>
                                                <xs:attribute name="name"
                                                    type="swe:qnameSimpleType" use="required"/>
                                                <xs:attributeGroup ref="swe:AssociationAttributes"/>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:restriction>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="method" type="sml:methodType">
    <xs:annotation>
        <xs:documentation>process method container</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="processes">
    <xs:annotation>
        <xs:documentation>collection of subprocesses that can be chained using connections
        </xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:restriction base="sml:processesType">
                <xs:sequence minOccurs="0">
                    <xs:element name="ProcessList">
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:restriction base="sml:_AbstractListType">
```

```
                                        <xs:sequence>
                                            <xs:element name="process" maxOccurs="unbounded">
                                                <xs:complexType>
                                                    <xs:sequence minOccurs="0">
                                                        <xs:element ref="sml:_Process"/>
                                                    </xs:sequence>
                                                    <xs:attribute name="name"
                                                        type="swe:qnameSimpleType" use="required"/>
                                                    <xs:attributeGroup ref="swe:AssociationAttributes"/>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:restriction>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="dataSources">
        <xs:annotation>
            <xs:documentation>collection of data sources that can be linked to input or parameters of processes
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="sml:dataSourcesType">
                    <xs:sequence minOccurs="0">
                        <xs:element name="DataSourceList">
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:restriction base="sml:_AbstractListType">
                                        <xs:sequence>
                                            <xs:element name="source" maxOccurs="unbounded">
                                                <xs:annotation>
                                                    <xs:documentation>Specify or reference a common
                                                        observation as source of data
                                                    </xs:documentation>
                                                </xs:annotation>
                                                <xs:complexType>
                                                    <xs:choice minOccurs="0">
                                                        <xs:element ref="om:CommonObservation"/>
                                                        <xs:element ref="sml:Data"/>
                                                    </xs:choice>
                                                    <xs:attribute name="name"
                                                        type="swe:qnameSimpleType" use="required"/>
                                                    <xs:attributeGroup ref="swe:AssociationAttributes"/>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:restriction>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="connections">
        <xs:annotation>
            <xs:documentation>provides links between processes or between data sources and processes
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
```

```
            <xs:complexContent>
                <xs:restriction base="sml:connectionsType">
                    <xs:sequence minOccurs="0">
                        <xs:element name="ConnectionList">
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:restriction base="sml:_AbstractListType">
                                        <xs:sequence>
                                            <xs:element ref="sml:connection" maxOccurs="unbounded"/>
                                        </xs:sequence>
                                    </xs:restriction>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="connection">
        <xs:annotation>
            <xs:documentation>Specify a connection between two I/O signals or data components
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:choice>
                <xs:element ref="sml:Link"/>
                <xs:element ref="sml:ArrayLink"/>
                <xs:element ref="sml:Dangle"/>
            </xs:choice>
            <xs:attribute name="name" type="swe:qnameSimpleType" use="optional"/>
        </xs:complexType>
    </xs:element>
    <!--==================-->
    <!--  Global groups  -->
    <!--==================-->
    <xs:group name="metadataGroup">
        <xs:annotation>
            <xs:documentation>Group containing all metadata information</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:group ref="sml:generalInfo" minOccurs="0"/>
            <xs:group ref="sml:constraints" minOccurs="0"/>
            <xs:group ref="sml:properties" minOccurs="0"/>
            <xs:group ref="sml:references" minOccurs="0"/>
            <xs:group ref="sml:history" minOccurs="0"/>
        </xs:sequence>
    </xs:group>
</xs:schema>
```

system.xsd.

The system.xsd schema defines a System derived from Process, with additional positional information.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Michael E Botts (University of Alabama in Huntsville) -->
<xs:schema targetNamespace="http://www.opengis.net/sensorML" xmlns:swe="http://www.opengis.net/swe"
xmlns:sml="http://www.opengis.net/sensorML" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>System object for core SensorML</xs:documentation>
    </xs:annotation>
    <!--========================-->
    <!--  Includes and Imports  -->
    <!--========================-->
    <xs:include schemaLocation="./process.xsd"/>
    <xs:include schemaLocation="./coordinateSystem.xsd"/>
    <xs:import namespace="http://www.opengis.net/swe" schemaLocation="./sweImports.xsd"/>
    <!--====================-->
    <!--  Global Elements  -->
    <!--====================-->
    <xs:element name="System" type="sml:SystemType" substitutionGroup="sml:_Process">
        <xs:annotation>
            <xs:documentation>System containing components</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!--==================-->
    <!--  Complex Types  -->
    <!--==================-->
    <xs:complexType name="SystemType">
        <xs:complexContent>
            <xs:extension base="sml:ProcessType">
                <xs:sequence>
                    <xs:element ref="sml:positions" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <!--==================-->
    <!--  Complex Types  -->
    <!--==================-->
    <xs:element name="positions">
        <xs:complexType>
            <xs:sequence minOccurs="0">
                <xs:element name="PositionList">
                    <xs:complexType>
                        <xs:complexContent>
                            <xs:restriction base="sml:_AbstractListType">
                                <xs:sequence>
                                    <xs:element name="position" maxOccurs="unbounded">
                                        <xs:complexType>
                                            <xs:choice minOccurs="0">
                                                <xs:element name="PositionProcess">
                                                    <xs:complexType>
                                                        <xs:sequence>
                                                            <xs:element name="process">
                                                                <xs:complexType>
                                                                    <xs:sequence minOccurs="0">
                                                                        <xs:element ref="sml:_Process"/>
                                                                    </xs:sequence>
                                                                    <xs:attributeGroup
ref="swe:AssociationAttributes"/>
```

```
                                                    </xs:complexType>
                                                </xs:element>
                                            </xs:sequence>
                                            <xs:attribute name="id" type="xs:ID" use="optional"/>
                                            <xs:attribute name="referenceFrame" type="xs:anyURI"
                                                use="required"/>
                                            <xs:attribute name="localFrame" type="xs:anyURI"
                                                use="required"/>
                                        </xs:complexType>
                                    </xs:element>
                                    <xs:element ref="swe:PositionData"/>
                                </xs:choice>
                                <xs:attribute name="name" type="swe:qnameSimpleType"/>
                                <xs:attributeGroup ref="swe:AssociationAttributes"/>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
</xs:sequence>
<xs:attributeGroup ref="swe:AssociationAttributes"/>
</xs:complexType>
</xs:element>
</xs:schema>
```

coordinateSystem.xsd.

The coordinateSystem.xsd schema defines the referenceFrame property and uses the GML 3.0.1 CoordinateReferenceSystem.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Michael E Botts (University of Alabama in Huntsville) -->
<xs:schema targetNamespace="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:swe="http://www.opengis.net/swe" xmlns:xlink="http://www.w3.org/1999/xlink"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <!--========================-->
    <!--  Includes and Imports  -->
    <!--========================-->
    <xs:import namespace="http://www.opengis.net/gml"
            schemaLocation="http://schemas.opengis.net/gml/3.1.1/base/coordinateReferenceSystems.xsd"/>
    <xs:import namespace="http://www.opengis.net/swe"
            schemaLocation="../../../sweCommon/1.0.30/parameters.xsd"/>
    <!--========================-->
    <!--  referenceFrame uses the Coordinate Reference System as defined by GML  -->
    <!--========================-->
    <xs:element name="referenceFrame">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="gml:EngineeringCRS"/>
            </xs:sequence>
            <xs:attribute name="name" type="swe:nameSimpleType" use="optional"/>
            <xs:attributeGroup ref="swe:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

omImports.xsd.

SensorML uses CommonObservation from the Observations and Measurements schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Alexandre Robin (University of Alabama in Huntsville) -->
<xs:schema targetNamespace="http://www.opengis.net/om" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:include schemaLocation="../../../om/omLite/commonObservation.xsd"/>
</xs:schema>
```

Annex B: XML Schemas for SWE Common (normative)

parameters.xsd.

The parameters.xsd schema defines simple data types and data aggregates used throughout SensorML and the Sensor Web Enablement initiative.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) --
>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Alexandre Robin (University of Alabama in Huntsville)
-->
<xs:schema targetNamespace="http://www.opengis.net/swe" xmlns:swe="http://www.opengis.net/swe"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>Data components for core SensorML
        Note that this schema defines a number of components that mirror types and elements already defined in
        GML, in the schema documents basicTypes.xsd, measures.xsd and temporal.xsd</xs:documentation>
    </xs:annotation>
    <!--========================-->
    <!--  Includes and Imports  -->
    <!--========================-->
    <xs:import namespace="http://www.w3.org/1999/xlink"
        schemaLocation="http://schemas.opengis.net/gml/3.1.0/xlink/xlinks.xsd"/>
    <!--====================-->
    <!--  Attribute Groups  -->
    <!--====================-->
    <xs:attributeGroup name="AssociationAttributes">
        <xs:attribute ref="xlink:href" use="optional"/>
    </xs:attributeGroup>
    <!--============================-->
    <!--  Main Substitution Groups  -->
    <!--============================-->
    <xs:element name="_DataGroup" type="swe:_GroupBaseType" abstract="true"/>
    <xs:element name="_DataArray" type="swe:_ArrayBaseType" abstract="true"/>
    <!--==========================-->
    <!--  Simple Data Components  -->
    <!--==========================-->
    <xs:element name="Boolean" type="swe:BooleanType">
        <xs:annotation>
            <xs:documentation>True or False, 0 or 1</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Count" type="swe:CountType">
        <xs:annotation>
            <xs:documentation>Integer number used for a counting value </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="CountRange" type="swe:CountRangeType">
        <xs:annotation>
            <xs:documentation>Integer pair used for specifying a count range</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Quantity" type="swe:QuantityType">
        <xs:annotation>
            <xs:documentation>Decimal number with optional unit and bounds</xs:documentation>
        </xs:annotation>
    </xs:element>
```

```
<xs:element name="QuantityRange" type="swe:QuantityRangeType">
    <xs:annotation>
        <xs:documentation>Decimal pair for specifying a quantity range</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="Category" type="swe:CategoryType">
    <xs:annotation>
        <xs:documentation>A simple token identifying a term or category (single spaces allowed);
            definition attribute should provide dictionary entry useful for interpretation of the value
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="Discussion" type="swe:TextType">
    <xs:annotation>
        <xs:documentation>A descriptive text with an optional topic</xs:documentation>
    </xs:annotation>
</xs:element>
<!--============================-->
<!--  Time Elements  -->
<!--============================-->
<xs:element name="IsoDateTime" type="swe:IsoDateTimeType">
    <xs:annotation>
        <xs:documentation>Time instance given in ISO 8601 format (e.g. 2004-04-18T12:03:04.6Z)
            or currentTime</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="TimeQuantity" type="swe:TimeQuantityType">
    <xs:annotation>
        <xs:documentation>Time relative to a time origin</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="Time" type="swe:TimeType">
    <xs:annotation>
        <xs:documentation>Time relative to a time origin</xs:documentation>
    </xs:annotation>
</xs:element>
<!--============================-->
<!--  Complex Data Components  -->
<!--============================-->
<xs:element name="DataGroup" substitutionGroup="swe:_DataGroup">
    <xs:annotation>
        <xs:documentation>Group of other data components</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="swe:_GroupBaseType">
                <xs:sequence minOccurs="0">
                    <xs:element name="component" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:group ref="swe:AnyData" minOccurs="0"/>
                            <xs:attribute name="name" type="swe:qnameSimpleType" use="required"/>
                            <xs:attributeGroup ref="swe:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="DataArray" substitutionGroup="swe:_DataArray">
    <xs:annotation>
        <xs:documentation>Array of other data components with a size</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="swe:_ArrayBaseType">
                <xs:sequence minOccurs="0">
```

```
                        <xs:element name="component">
                            <xs:complexType>
                                <xs:group ref="swe:AnyData" minOccurs="0"/>
                                <xs:attribute name="name" type="swe:qnameSimpleType" use="optional"/>
                                <xs:attributeGroup ref="swe:AssociationAttributes"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element ref="swe:tupleValues" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="tupleValues">
        <xs:annotation>
            <xs:documentation>List of space separated tuples</xs:documentation>
        </xs:annotation>
        <xs:simpleType>
            <xs:list itemType="swe:tupleType"/>
        </xs:simpleType>
    </xs:element>
    <xs:element name="Curve" substitutionGroup="swe:_DataArray">
        <xs:annotation>
            <xs:documentation>Curve describing variations of a parameter vs. another quantity
             </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="swe:CurveType"/>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="ConditionalValue" type="swe:ConditionalValueType"
        substitutionGroup="swe:_DataGroup"/>
    <xs:element name="NormalizedCurve" type="swe:NormalizedCurveType"
        substitutionGroup="swe:_DataGroup"/>
    <xs:complexType name="CurveType">
        <xs:complexContent>
            <xs:extension base="swe:_ArrayBaseType">
                <xs:sequence>
                    <xs:element name="definition">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element name="Coordinates">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:extension base="swe:_GroupBaseType">
                                                <xs:sequence>
                                                    <xs:element name="axis" maxOccurs="unbounded">
                                                        <xs:complexType>
                                                            <xs:group ref="swe:AnyScalar"/>
                                                            <xs:attribute name="name"
                                                                type="swe:qnameSimpleType" use="required"/>
                                                        </xs:complexType>
                                                    </xs:element>
                                                </xs:sequence>
                                            </xs:extension>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                    <xs:element ref="swe:tupleValues"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
```

```
</xs:complexType>
<xs:complexType name="NormalizedCurveType">
    <xs:complexContent>
        <xs:extension base="swe:_GroupBaseType">
            <xs:sequence>
                <xs:element name="inputGain" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:Quantity"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="inputBias" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:Quantity"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="outputGain" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:Quantity"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="outputBias" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:Quantity"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="function">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="swe:Curve"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="ConditionalValueType">
    <xs:complexContent>
        <xs:extension base="swe:_GroupBaseType">
            <xs:sequence>
                <xs:element name="condition" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:choice minOccurs="0">
                            <xs:element ref="swe:Discussion"/>
                            <xs:group ref="swe:AnyData"/>
                        </xs:choice>
                        <xs:attribute name="name" type="swe:qnameSimpleType" use="required"/>
                        <xs:attributeGroup ref="swe:AssociationAttributes"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="value">
                    <xs:complexType>
                        <xs:group ref="swe:AnyData" minOccurs="0"/>
                        <xs:attributeGroup ref="swe:AssociationAttributes"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
```

```
</xs:complexType>
<!--================-->
<!--  Simple Types  -->
<!--================-->
<!-- EmptyType allows Quantity, Count, etc to be nillable (i.e. have no value) -->
<!-- Consider using XML Schema's nillable="true" -->
<xs:simpleType name="emptyType">
    <xs:restriction base="xs:string">
        <xs:length value="0"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="booleanOrEmpty">
    <xs:union memberTypes="xs:boolean swe:emptyType"/>
</xs:simpleType>
<xs:simpleType name="countOrEmpty">
    <xs:union memberTypes="xs:integer swe:emptyType"/>
</xs:simpleType>
<xs:simpleType name="decimalOrEmpty">
    <xs:union memberTypes="xs:double swe:emptyType"/>
    <!--Need double to support scientific notation (ex: 10e3)</xs:documentation-->
</xs:simpleType>
<xs:simpleType name="iso8601">
    <xs:union memberTypes="xs:date xs:time xs:dateTime swe:timeString swe:emptyType"/>
    <!-- Implemented in GML using gml:TimePositionUnion or gml:TImePositionType -->
</xs:simpleType>
<xs:simpleType name="timeSimpleType">
    <xs:union memberTypes="xs:date xs:time xs:dateTime swe:timeString xs:double swe:emptyType"/>
</xs:simpleType>
<xs:simpleType name="timeString">
    <xs:restriction base="xs:string">
        <xs:enumeration value="currentTime"/>
        <xs:enumeration value="unknown"/>
    </xs:restriction>
    <!-- Implemented in GML using indeterminatePosition attribute on gml:TImePositionType  -->
</xs:simpleType>
<xs:simpleType name="uomType">
    <xs:restriction base="xs:anyURI"/>
</xs:simpleType>
<!-- consider using GML 3.2's UomIdentifier -->
<!-- this gives an (unambiguous) choice of the UCUM symbol or a URI -->
<xs:simpleType name="uomIdentifier">
    <xs:union memberTypes="swe:uomSymbol swe:uomURI"/>
</xs:simpleType>
<xs:simpleType name="uomSymbol">
    <xs:restriction base="xs:string">
        <xs:pattern value="[^: \n\r\t]+"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="uomURI">
    <xs:restriction base="xs:anyURI">
        <xs:pattern value="([a-zA-Z][a-zA-Z0-9\-\+\.]*:|\.\./|\./|#).*"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="definitionType">
    <xs:restriction base="xs:anyURI"/>
</xs:simpleType>
<xs:simpleType name="nameSimpleType">
    <xs:restriction base="xs:token">
        <xs:maxLength value="30"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="qnameSimpleType">
    <xs:restriction base="xs:QName"/>
</xs:simpleType>
<xs:simpleType name="axisCodeSimpleType">
    <xs:restriction base="xs:string">
        <xs:enumeration value="X"/>
```

```
            <xs:enumeration value="Y"/>
            <xs:enumeration value="Z"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="timeReferenceSimpleType">
        <xs:union memberTypes="xs:date xs:time xs:dateTime xs:anyURI"/>
    </xs:simpleType>
    <xs:simpleType name="tupleType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="decimalList">
        <xs:list itemType="xs:double"/>
        <!-- this is a double, because decimal does not allow scientific notation -->
    </xs:simpleType>
    <xs:simpleType name="decimalPair">
        <xs:restriction base="swe:decimalList">
            <xs:length value="2"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="countList">
        <xs:list itemType="xs:integer"/>
    </xs:simpleType>
    <xs:simpleType name="countPair">
        <xs:restriction base="swe:countList">
            <xs:length value="2"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="arraySizeSimpleType">
        <xs:union memberTypes="xs:positiveInteger xs:IDREF"/>
    </xs:simpleType>
    <!--================-->
    <!--  Scalar Types with additional metadata included as XML attributes  -->
    <!--================-->
    <xs:complexType name="BooleanType">
        <xs:simpleContent>
            <xs:extension base="swe:booleanOrEmpty">
                <xs:attribute name="id" type="xs:ID" use="optional"/>
                <xs:attribute name="definition" type="swe:definitionType" use="optional"/>
                <xs:attribute name="axisCode" type="swe:axisCodeSimpleType" use="optional"/>
                <xs:attribute name="fixed" type="xs:boolean" use="optional" default="false"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="CountType">
        <xs:simpleContent>
            <xs:extension base="swe:countOrEmpty">
                <xs:attribute name="id" type="xs:ID" use="optional"/>
                <xs:attribute name="definition" type="swe:definitionType" use="optional"/>
                <xs:attribute name="axisCode" type="swe:axisCodeSimpleType" use="optional"/>
                <xs:attribute name="min" type="xs:integer" use="optional"/>
                <xs:attribute name="max" type="xs:integer" use="optional"/>
                <xs:attribute name="fixed" type="xs:boolean" use="optional" default="false"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="CountRangeType">
        <xs:simpleContent>
            <xs:extension base="swe:countPair">
                <xs:attribute name="id" type="xs:ID" use="optional"/>
                <xs:attribute name="definition" type="swe:definitionType" use="optional"/>
                <xs:attribute name="axisCode" type="swe:axisCodeSimpleType" use="optional"/>
                <xs:attribute name="fixed" type="xs:boolean" use="optional" default="false"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="QuantityType">
        <xs:simpleContent>
```

```
            <xs:extension base="swe:decimalOrEmpty">
                <xs:attribute name="id" type="xs:ID" use="optional"/>
                <xs:attribute name="definition" type="swe:definitionType" use="optional"/>
                <xs:attribute name="uom" type="swe:uomType" use="optional"/>
                <xs:attribute name="scale" type="xs:double" use="optional"/>
                <xs:attribute name="axisCode" type="swe:axisCodeSimpleType" use="optional"/>
                <xs:attribute name="min" type="xs:double" use="optional"/>
                <xs:attribute name="max" type="xs:double" use="optional"/>
                <xs:attribute name="fixed" type="xs:boolean" use="optional" default="false"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="QuantityRangeType">
        <xs:simpleContent>
            <xs:extension base="swe:decimalPair">
                <xs:attribute name="id" type="xs:ID" use="optional"/>
                <xs:attribute name="definition" type="swe:definitionType" use="optional"/>
                <xs:attribute name="uom" type="swe:uomType" use="optional"/>
                <xs:attribute name="scale" type="xs:double" use="optional"/>
                <xs:attribute name="axisCode" type="swe:axisCodeSimpleType" use="optional"/>
                <xs:attribute name="fixed" type="xs:boolean" use="optional" default="false"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="TextType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attribute name="id" type="xs:ID" use="optional"/>
                <xs:attribute name="topic" type="xs:token" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="CategoryType">
        <xs:simpleContent>
            <xs:extension base="xs:token">
                <xs:attribute name="id" type="xs:ID" use="optional"/>
                <xs:attribute name="definition" type="swe:definitionType" use="optional"/>
                <xs:attribute name="fixed" type="xs:boolean" use="optional" default="false"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <!--============================-->
    <!--  Time Types  -->
    <!--============================-->
    <xs:complexType name="IsoDateTimeType">
        <xs:simpleContent>
            <xs:extension base="swe:iso8601">
                <xs:attribute name="id" type="xs:ID" use="optional"/>
                <xs:attribute name="definition" type="swe:definitionType" use="optional"
fixed="urn:ogc:data:time:iso8601"/>
                <xs:attribute name="fixed" type="xs:boolean" use="optional" default="false"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="TimeQuantityType">
        <xs:simpleContent>
            <xs:extension base="swe:QuantityType">
                <xs:attribute name="referenceTime" type="swe:timeReferenceSimpleType" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="TimeType">
        <xs:simpleContent>
            <xs:extension base="swe:timeSimpleType">
                <xs:attribute name="id" type="xs:ID" use="optional"/>
                <xs:attribute name="definition" type="swe:definitionType" use="optional"/>
                <xs:attribute name="referenceTime" type="swe:timeReferenceSimpleType" use="optional"/>
```

```
                <xs:attribute name="localTime" type="swe:timeReferenceSimpleType" use="optional"/>
                <xs:attribute name="uom" type="swe:uomType" use="optional"/>
                <xs:attribute name="scale" type="xs:double" use="optional"/>
                <xs:attribute name="fixed" type="xs:boolean" use="optional" default="false"/>
            </xs:extension>
        </xs:simpleContent>
        <!-- Implemented in GML as gml:TimeType -->
    </xs:complexType>
    <!--==========================-->
    <!--  Abstract Complex Types  -->
    <!--==========================-->
    <xs:complexType name="_GroupBaseType" abstract="true">
        <xs:attribute name="id" type="xs:ID" use="optional"/>
        <xs:attribute name="definition" type="swe:definitionType" use="optional"/>
        <xs:attribute name="fixed" type="xs:boolean" use="optional" default="false"/>
    </xs:complexType>
    <xs:complexType name="_ArrayBaseType" abstract="true">
        <xs:complexContent>
            <xs:extension base="swe:_GroupBaseType">
                <xs:attribute name="arraySize" type="swe:arraySizeSimpleType" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <!--==================-->
    <!--  Common Groups  -->
    <!--==================-->
    <xs:group name="AnyScalar">
        <xs:choice>
            <xs:element ref="swe:Boolean"/>
            <xs:group ref="swe:AnyNumerical"/>
            <xs:element ref="swe:Category"/>
        </xs:choice>
    </xs:group>
    <xs:group name="AnyNumerical">
        <xs:choice>
            <xs:element ref="swe:Count"/>
            <xs:element ref="swe:Quantity"/>
            <xs:group ref="swe:Time"/>
        </xs:choice>
    </xs:group>
    <xs:group name="AnyData">
        <xs:choice>
            <xs:group ref="swe:AnyScalar"/>
            <xs:element ref="swe:QuantityRange"/>
            <xs:element ref="swe:CountRange"/>
            <xs:element ref="swe:_DataGroup"/>
            <xs:element ref="swe:_DataArray"/>
        </xs:choice>
    </xs:group>
    <xs:group name="Time">
        <xs:choice>
            <xs:element ref="swe:IsoDateTime"/>
            <xs:element ref="swe:TimeQuantity"/>
            <xs:element ref="swe:Time"/>
        </xs:choice>
    </xs:group>
    <xs:group name="Curves">
        <xs:choice>
            <xs:element ref="swe:Curve"/>
            <xs:element ref="swe:NormalizedCurve"/>
        </xs:choice>
    </xs:group>
</xs:schema>
```

data.xsd.

The data.xsd schema defines data encodings used in SensorML and throughout the Sensor Web Enablement initiative.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) --
>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Alexandre Robin (University of Alabama in Huntsville)
-->
<xs:schema targetNamespace="http://www.opengis.net/swe" xmlns:swe="http://www.opengis.net/swe"
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>Defines basic ResponseType definition and commonly used sensor
characteristics</xs:documentation>
    </xs:annotation>
    <!--========================-->
    <!--  Includes and Imports  -->
    <!--========================-->
    <xs:import namespace="http://www.w3.org/1999/xlink"
schemaLocation="http://schemas.opengis.net/gml/3.1.0/xlink/xlinks.xsd"/>
    <xs:include schemaLocation="./parameters.xsd"/>
    <!-- ============================
    Substitution groups
   ============================-->
    <xs:element name="_Encoding" type="swe:_EncodingType" abstract="true"/>
    <xs:element name="_MultiplexEncoding" type="swe:_MultiplexEncodingType" abstract="true"/>
    <xs:element name="  Binary" type="swe:  BinaryType" abstract="true"/>
    <xs:element name="_Data" type="swe:_DataType" abstract="true"/>
    <!-- ============================
    Concrete elements
   ============================-->
    <!-- BEGIN change schema patterns to match GML use of global types - SJDC 2005-08-07-->
    <!-- Does not change model but enables re-use in other contexts -->
    <xs:complexType name="DataDefinitionType">
        <xs:sequence>
            <xs:element name="dataComponents" type="swe:DataComponentsPropertyType"/>
            <xs:element name="encoding" type="swe:EncodingPropertyType"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:complexType>
    <!-- ..... -->
    <xs:element name="DataDefinition" type="swe:DataDefinitionType"/>
    <!-- ..... -->
    <xs:complexType name="DataDefinitionPropertyType">
        <xs:sequence minOccurs="0">
            <xs:element ref="swe:DataDefinition"/>
        </xs:sequence>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
    </xs:complexType>
    <xs:complexType name="EncodingPropertyType">
        <xs:sequence minOccurs="0">
            <xs:element ref="swe:_Encoding"/>
        </xs:sequence>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
    </xs:complexType>
    <xs:complexType name="MultiplexEncodingPropertyType">
        <xs:sequence minOccurs="0">
            <xs:element ref="swe:_MultiplexEncoding"/>
```

```
        </xs:sequence>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
    </xs:complexType>
    <xs:complexType name="DataComponentsPropertyType">
        <xs:sequence minOccurs="0">
            <xs:group ref="swe:AnyData"/>
        </xs:sequence>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
    </xs:complexType>
    <!-- ============ -->
    <xs:complexType name="DataValueType">
        <xs:annotation>
            <xs:documentation>supports data values as a string, as with tuple and base64 blocks; can also allow
                an external link to out-of-band data; can also support multiplex data that takes XML elements, such
                 as TML</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:extension base="xs:anyType">
                <xs:attribute name="externalLink" type="xs:anyURI" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SimpleDataPropertyType">
        <xs:annotation>
            <xs:documentation>Supports a DataGroup/DataArray with inline data values</xs:documentation>
        </xs:annotation>
        <xs:group ref="swe:AnyData" minOccurs="0"/>
        <xs:attributeGroup ref="swe:AssociationAttributes"/>
    </xs:complexType>
    <!-- END change schema patterns to match GML use of global types -->
    <!-- ============ -->
    <xs:element name="BinaryValue" substitutionGroup="swe:_Binary">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="swe:_BinaryType">
                    <xs:attribute name="dataType" use="required">
                        <xs:simpleType>
                            <xs:restriction base="xs:token">
                                <xs:enumeration value="urn:ogc:data:float"/>
                                <xs:enumeration value="urn:ogc:data:double"/>
                                <xs:enumeration value="urn:ogc:data:boolean"/>
                                <xs:enumeration value="urn:ogc:data:signedByte"/>
                                <xs:enumeration value="urn:ogc:data:unsignedByte"/>
                                <xs:enumeration value="urn:ogc:data:signedShort"/>
                                <xs:enumeration value="urn:ogc:data:unsignedShort"/>
                                <xs:enumeration value="urn:ogc:data:signedInt"/>
                                <xs:enumeration value="urn:ogc:data:unsignedInt"/>
                                <xs:enumeration value="urn:ogc:data:signedLong"/>
                                <xs:enumeration value="urn:ogc:data:unsignedLong"/>
                                <xs:enumeration value="urn:ogc:data:UTF-string"/>
                                <xs:enumeration value="urn:ogc:data:padding"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                    <xs:attribute name="bitOffset" type="xs:integer" use="optional" default="0"/>
                    <xs:attribute name="bitLength" type="xs:positiveInteger" use="required"/>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="BinaryGroup" substitutionGroup="swe:_Binary">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="swe:_BinaryType">
                    <xs:sequence>
                        <xs:element name="component" maxOccurs="unbounded">
                            <xs:complexType>
```

```
                        <xs:sequence>
                            <xs:element ref="swe:_Binary"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="BinaryArray" substitutionGroup="swe:_Binary">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="swe:_BinaryType">
                <xs:sequence>
                    <xs:element name="component">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="swe:_Binary"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="arraySize" type="swe:arraySizeSimpleType" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="StandardFormat" substitutionGroup="swe:_Encoding">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="swe:_EncodingType">
                <xs:attribute name="mimeType" type="xs:token" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="XMLTuple" substitutionGroup="swe:_Encoding">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="swe:_EncodingType">
                <xs:attribute name="decimalSeparator" type="xs:string" use="required"/>
                <xs:attribute name="tokenSeparator" type="xs:string" use="required"/>
                <xs:attribute name="tupleSeparator" type="xs:string" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="BinaryBlock" substitutionGroup="swe:_Encoding">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="swe:_EncodingType">
                <xs:sequence>
                    <xs:element name="byteLength" type="xs:positiveInteger"/>
                    <xs:element name="byteEncoding">
                        <xs:simpleType>
                            <xs:restriction base="xs:anyURI">
                                <xs:enumeration value="urn:ogc:data:raw"/>
                                <xs:enumeration value="urn:ogc:data:hex"/>
                                <xs:enumeration value="urn:ogc:data:base64"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="byteOrder">
                        <xs:simpleType>
                            <xs:restriction base="xs:anyURI">
                                <xs:enumeration value="urn:ogc:data:bigEndian"/>
```

```
                                <xs:enumeration value="urn:ogc:data:littleEndian"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="encryption" minOccurs="0">
                        <xs:simpleType>
                            <xs:restriction base="xs:anyURI">
                                <xs:enumeration value="urn:ogc:data:XOR"/>
                                <xs:enumeration value="urn:ogc:data:MD5"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="compression" minOccurs="0">
                        <xs:simpleType>
                            <xs:restriction base="xs:anyURI">
                                <xs:enumeration value="urn:ogc:data:gzip"/>
                                <xs:enumeration value="urn:ogc:data:bzip"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:element>
                    <xs:element name="structure" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="swe:_Binary"/>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="AsciiBlock" substitutionGroup="swe:_Encoding">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="swe:_EncodingType">
                <xs:attribute name="tokenSeparator" type="xs:string" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!-- ============================
     Complex Types
     ============================-->
<xs:complexType name="_EncodingType">
    <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>
<xs:complexType name="_MultiplexEncodingType"/>
<xs:complexType name="_BinaryType">
    <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>
<xs:complexType name="_DataType">
    <xs:attribute name="id" type="xs:ID" use="optional"/>
</xs:complexType>
</xs:schema>
```

positionData.xsd.

The positionData.xsd schema derives position data groups used ine SensorML and throughout the Sensor Web Enablement encodings and services.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Alexandre Robin (University of Alabama in Huntsville) -->
```

```
<xs:schema targetNamespace="http://www.opengis.net/swe" xmlns:swe="http://www.opengis.net/swe"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>Definitions for some common position parameters</xs:documentation>
    </xs:annotation>
    <!--========================-->
    <!--  Includes and Imports  -->
    <!--========================-->
    <xs:include schemaLocation="./parameters.xsd"/>
    <!--========================-->
    <!--  Substitution Groups  -->
    <!--========================-->
    <xs:element name="PositionData" type="swe:_PositionType" abstract="false"
        substitutionGroup="swe:_DataGroup">
        <xs:annotation>
            <xs:documentation>Head of substitution Group for Composite Position Data</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="LocationData" type="swe:_PositionType" abstract="false"
        substitutionGroup="swe:PositionData">
        <xs:annotation>
            <xs:documentation>Head of substitution Group for Location Data</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="OrientationData" type="swe:_PositionType" abstract="false"
        substitutionGroup="swe:PositionData">
        <xs:annotation>
            <xs:documentation>Head of substitution Group for Orientation Data</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!--========================-->
    <!--  Position DataGroup  -->
    <!--========================-->
    <xs:element name="Position" substitutionGroup="swe:PositionData">
        <xs:annotation>
            <xs:documentation>Location and Orientation given by a sequence of rotations or translations.
                Transformations are applied in the order listed</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="swe:_PositionType">
                    <xs:sequence>
                        <xs:element name="time" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:group ref="swe:Time" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="location" minOccurs="0">
                            <xs:complexType>
                                <xs:choice>
                                    <xs:element ref="swe:Location"/>
                                    <xs:element ref="swe:GeoLocation"/>
                                </xs:choice>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="orientation" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="swe:Orientation" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
```

```
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="Location" substitutionGroup="swe:LocationData">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="swe:_PositionType">
                    <xs:sequence>
                        <xs:element name="coordinate" maxOccurs="3">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="swe:Quantity"/>
                                </xs:sequence>
                                <xs:attribute name="name" type="swe:qnameSimpleType" use="required"/>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="GeoLocation" substitutionGroup="swe:LocationData">
        <xs:annotation>
            <xs:documentation>Position given by latitude, longitude, altitude</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="swe:_PositionType">
                    <xs:sequence minOccurs="0">
                        <xs:element name="longitude">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="swe:Quantity" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="latitude">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="swe:Quantity" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="altitude" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="swe:Quantity" minOccurs="0"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="Orientation" substitutionGroup="swe:OrientationData">
        <xs:annotation>
            <xs:documentation>Orientation given by euler angle with an order of rotation</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="swe:_PositionType">
                    <xs:sequence minOccurs="0">
                        <xs:element name="coordinate" maxOccurs="3">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="swe:Quantity"/>
```

```
                                </xs:sequence>
                                <xs:attribute name="name" type="swe:qnameSimpleType" use="required"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="order">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="swe:Category"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--==================-->
    <!--  Complex Types  -->
    <!--==================-->
    <xs:complexType name="_PositionType">
        <xs:complexContent>
            <xs:extension base="swe:_GroupBaseType">
                <xs:attribute name="referenceFrame" type="xs:anyURI" use="optional"/>
                <xs:attribute name="localFrame" type="xs:anyURI" use="optional"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

Annex C: XML Schemas for Transducer (informative)

transducer.xsd.

The transducer.xsd schema defines a model and encoding for a general transducer. It is based on a model presented within TransducerML and is expected to be refined and perhaps defined within the TransducerML namespace (http://www.opengis/net/tml). It is presented here as a recommended common base schema for possible use in defining most if not all sensor components. *Transducer* is derived from *ProcessModelType* by restriction.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 rel. 2 U (http://www.xmlspy.com) by Mike Botts (University of Alabama in Huntsville) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Alexandre Robin (University of Alabama in Huntsville) -->
<xs:schema targetNamespace="http://www.opengis.net/sensorML" xmlns:sml="http://www.opengis.net/sensorML"
xmlns:swe="http://www.opengis.net/swe" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <!--==========================-->
    <xs:include schemaLocation="./process.xsd"/>
    <xs:import namespace=http://www.w3.org/1999/xlink
        schemaLocation="http://schemas.opengis.net/gml/3.1.0/xlink/xlinks.xsd"/>
    <xs:import namespace="http://www.opengis.net/swe" schemaLocation="./sweImports.xsd"/>
    <!--====================-->
    <xs:element name="Transducer" type="sml:TransducerType" substitutionGroup="sml:_Process"/>
    <!--====================-->
    <xs:complexType name="TransducerType">
        <xs:complexContent>
            <xs:restriction base="sml:_ProcessModelType">
                <xs:sequence>
                    <xs:group ref="sml:metadataGroup"/>
                    <xs:element ref="sml:referenceFrame" minOccurs="0" maxOccurs="2"/>
                    <xs:element name="inputs">
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:restriction base="sml:inputsType">
                                    <xs:sequence minOccurs="0">
                                        <xs:element name="InputList">
                                            <xs:complexType>
                                                <xs:complexContent>
                                                    <xs:restriction base="sml:_AbstractListType">
                                                        <xs:sequence>
                                                            <xs:element name="input">
                                                                <xs:complexType>
                                                                    <xs:group ref="swe:AnyScalar"/>
                                                                    <xs:attribute name="name"
                                                                        type="swe:qnameSimpleType"/>
                                                                </xs:complexType>
                                                            </xs:element>
                                                            <xs:element name="index" minOccurs="0"
                                                                maxOccurs="3">
                                                                <xs:complexType>
                                                                    <xs:sequence>
                                                                        <xs:element ref="swe:Count"/>
                                                                    </xs:sequence>
                                                                    <xs:attribute name="name"
                                                                        type="swe:qnameSimpleType"/>
                                                                </xs:complexType>
                                                            </xs:element>
```

```
                                        </xs:sequence>
                                    </xs:restriction>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="outputs">
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="sml:outputsType">
                    <xs:sequence minOccurs="0">
                        <xs:element name="OutputList">
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:restriction base="sml:_AbstractListType">
                                        <xs:sequence>
                                            <xs:element name="output">
                                                <xs:complexType>
                                                    <xs:group ref="swe:AnyScalar"/>
                                                    <xs:attribute name="name"
                                                        type="swe:qnameSimpleType"/>
                                                </xs:complexType>
                                            </xs:element>
                                        </xs:sequence>
                                    </xs:restriction>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>

<!- left shift of tabs for documentation →
    <xs:element name="parameters">
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="sml:parametersType">
                    <xs:sequence minOccurs="0">
                        <xs:element name="ParameterList">
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:restriction base="sml:_AbstractListType">
                                        <xs:sequence>
                                            <xs:element name="steadyStateResponse"
                                                maxOccurs="2">
                                                <xs:annotation>
                                                    <xs:documentation>Calibration function. This
                                                        curve describes how to map a given
                                                        transducer input to the corresponding
                                                        output within the working range of the
                                                        transducer and in steady state. One may
                                                        provide two calibration curves, one with
                                                        increasing input values (forward direction)
                                                        and one for decreasing (reverse direction)
                                                        input values</xs:documentation>
                                                </xs:annotation>
                                                <xs:complexType>
                                                    <xs:sequence minOccurs="0">
                                                        <xs:element ref="sml:NormalizedCurve"/>
                                                    </xs:sequence>
                                                    <xs:attributeGroup
```

```
                ref="swe:AssociationAttributes"/>
            <xs:attribute name="direction" use="optional"
                default="both">
                <xs:simpleType>
                    <xs:restriction base="xs:token">
                        <xs:enumeration
                            value="forward"/>
                        <xs:enumeration
                            value="reverse"/>
                        <xs:enumeration value="both"/>
                    </xs:restriction>
                </xs:simpleType>
            </xs:attribute>
        </xs:complexType>
    </xs:element>
    <xs:element name="latencyTime" minOccurs="0">
        <xs:annotation>
            <xs:documentation>Constant or Curve giving
                delay between sample collection and
                availability of the data on the output.
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:choice minOccurs="0">
                <xs:element ref="sml:NormalizedCurve"/>
                <xs:element ref="swe:Quantity"/>
            </xs:choice>
            <xs:attributeGroup
                ref="swe:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="integrationTime" minOccurs="0">
        <xs:annotation>
            <xs:documentation>Constant or Curve giving
                sampling integration time
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:choice minOccurs="0">
                <xs:element ref="sml:NormalizedCurve"/>
                <xs:element ref="swe:Quantity"/>
            </xs:choice>
            <xs:attributeGroup
                ref="swe:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="accuracy" minOccurs="0">
        <xs:annotation>
            <xs:documentation>Constant or Curve giving
                accuracy of measurement
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:choice minOccurs="0">
                <xs:element ref="sml:NormalizedCurve"/>
                <xs:element ref="swe:Quantity"/>
            </xs:choice>
            <xs:attributeGroup
                ref="swe:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="impulseResponse"
        minOccurs="0">
        <xs:annotation>
            <xs:documentation> Gives the amplitude of
                the transducer output vs. time when a
                impulse (Delta function) signal is applied to
```

```
                its input. This is the equivalent of the
                frequency response in the time domain.
            </xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:sequence minOccurs="0">
                    <xs:element ref="sml:NormalizedCurve"/>
                </xs:sequence>
                <xs:attributeGroup
                    ref="swe:AssociationAttributes"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="frequencyResponse"
            minOccurs="0">
            <xs:annotation>
                <xs:documentation>  Specifies the power
                    density of the output of the transducer
                    when its input is exposed to the full
                    spectrum. This is the equivalent of the
                    impulse response in the frequency
                    domain. </xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:sequence minOccurs="0">
                    <xs:element ref="sml:NormalizedCurve"/>
                </xs:sequence>
                <xs:attributeGroup
                    ref="swe:AssociationAttributes"/>
                <xs:attribute name="type" use="optional"
                    default="carrier">
                    <xs:simpleType>
                        <xs:restriction base="xs:token">
                            <xs:enumeration value="carrier"/>
                            <xs:enumeration
                                value="modulation"/>
                        </xs:restriction>
                    </xs:simpleType>
                </xs:attribute>
            </xs:complexType>
        </xs:element>
        <xs:element name="ambiguityShape" minOccurs="0">
            <xs:annotation>
                <xs:documentation> Gives power pattern
                    curves used to define the ambiguity shape
                    (or instantaneous field of influence) of the
                    transducer. </xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:sequence minOccurs="0">
                    <xs:element
                    name="PowerPatternCurves">
                        <xs:complexType>
                            <xs:complexContent>
                               <xs:extension
                            base="swe:_GroupBaseType">
                                <xs:sequence>
                                    <xs:element
                                     name="curve"
                                    maxOccurs="7">
                            <xs:annotation>
                                <xs:documentation> Each curve specifies
                                the power pattern relative to one spatial or
                                temporal coordinate expressed relative to
                                sample frame. </xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                    <xs:sequence minOccurs="0">
```

```
                                                    <xs:element
                                                        ref="sml:NormalizedCurve"/>
                                                </xs:sequence>
                                                <xs:attribute name="name"
                                                    type="swe:qnameSimpleType"
                                                    use="required"/>
                                                <xs:attributeGroup
                                                    ref="swe:AssociationAttributes"/>
                                                </xs:complexType>
                                            </xs:element>
                                            <xs:element name="rotationOrder"
                                                minOccurs="0">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element ref="swe:Category"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                </xs:extension>
                            </xs:complexContent>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attributeGroup ref="swe:AssociationAttributes"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="samplePosition" minOccurs="0">
            <xs:annotation>
                <xs:documentation> List of curves each giving one spatial or temporal
                    coordinate vs. an index. This defines the sample frame position for
                    each index (internal geometry). </xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:sequence minOccurs="0">
                    <xs:element name="PositionCurves">
                        <xs:complexType>
                            <xs:complexContent>
                                <xs:extension base="swe:_PositionType">
                                    <xs:sequence>
                                        <xs:element name="curve" maxOccurs="7">
                                            <xs:annotation>
                                                <xs:documentation> Each curves specifies
                                                 one coordinate (spatial or temporal) of the
                                                full position. </xs:documentation>
                                            </xs:annotation>
                                            <xs:complexType>
                                                <xs:sequence minOccurs="0">
                                                    <xs:element
                                                        ref="sml:NormalizedCurve"/>
                                                </xs:sequence>
                                                <xs:attribute name="name"
                                                    type="swe:qnameSimpleType"
                                                    use="required"/>
                                                <xs:attributeGroup
                                                    ref="swe:AssociationAttributes"/>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="rotationOrder"
                                            minOccurs="0">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element ref="swe:Category"/>
                                                </xs:sequence>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
```

```
                                                    </xs:extension>
                                                </xs:complexContent>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attributeGroup ref="swe:AssociationAttributes"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                    </xs:restriction>
                </xs:complexContent>
            </xs:complexType>
        </xs:element>
        </xs:sequence>
    </xs:restriction>
</xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="method">
    <xs:complexType>
        <xs:complexContent>
            <xs:restriction base="sml:methodType">
                <xs:attribute ref="xlink:href" use="required" fixed="urn:ogc:process:transducer"/>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
</xs:sequence>
</xs:restriction>
</xs:complexContent>
</xs:complexType>
<!--========================-->
<xs:element name="NormalizedCurve" type="sml:NormalizedCurveType">
    <xs:annotation>
        <xs:documentation>Defines a curve which can be translated and scaled using input/output bias and gain.
            It also provides interpolation method and order to allow a more compact and precise representation of
            functions.  Default values: inputBias = 0, outputBias=0, inputGain=1, outputGain=1
            interpolationMethod = polynomial, interpolationOrder=1        </xs:documentation>
    </xs:annotation>
</xs:element>
<!--========================-->
    <xs:complexType name="NormalizedCurveType">
        <xs:complexContent>
            <xs:extension base="swe:_GroupBaseType">
                <xs:sequence>
                    <xs:element name="inputBias" minOccurs="0">
                        <xs:complexType>
                            <xs:choice minOccurs="0">
                                <xs:element ref="swe:Quantity"/>
                                <xs:element ref="sml:IndexedCurve"/>
                            </xs:choice>
                            <xs:attributeGroup ref="swe:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="inputGain" minOccurs="0">
                        <xs:complexType>
                            <xs:choice minOccurs="0">
                                <xs:element ref="swe:Quantity"/>
                                <xs:element ref="sml:IndexedCurve"/>
                            </xs:choice>
                            <xs:attributeGroup ref="swe:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="outputBias" minOccurs="0">
                        <xs:complexType>
                            <xs:choice minOccurs="0">
                                <xs:element ref="swe:Quantity"/>
```

```
                        <xs:element ref="sml:IndexedCurve"/>
                    </xs:choice>
                    <xs:attributeGroup ref="swe:AssociationAttributes"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="outputGain" minOccurs="0">
                <xs:complexType>
                    <xs:choice minOccurs="0">
                        <xs:element ref="swe:Quantity"/>
                        <xs:element ref="sml:IndexedCurve"/>
                    </xs:choice>
                    <xs:attributeGroup ref="swe:AssociationAttributes"/>
                </xs:complexType>
            </xs:element>
            <xs:element name="interpolationMethod" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="swe:Category"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="interpolationOrder" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="swe:Count"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="function">
                <xs:complexType>
                    <xs:choice minOccurs="0">
                        <xs:element ref="swe:Curve"/>
                        <xs:element ref="sml:IndexedCurve"/>
                    </xs:choice>
                    <xs:attributeGroup ref="swe:AssociationAttributes"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:extension>
  </xs:complexContent>
</xs:complexType>
<!--========================-->
<xs:element name="IndexedCurve" type="sml:IndexedCurveType"/>
<!--========================-->
<xs:complexType name="IndexedCurveType">
    <xs:complexContent>
        <xs:extension base="swe:_ArrayBaseType">
            <xs:sequence>
                <xs:element name="definition">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="Coordinates">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="index" maxOccurs="unbounded">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element ref="swe:Count"/>
                                                </xs:sequence>
                                                <xs:attribute name="name" type="swe:qnameSimpleType"/>
                                            </xs:complexType>
                                        </xs:element>
                                        <xs:element name="value" maxOccurs="unbounded">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:group ref="swe:AnyScalar"/>
                                                </xs:sequence>
```

```
                                        <xs:attribute name="name" type="swe:qnameSimpleType"/>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element ref="swe:tupleValues"/>
    </xs:sequence>
</xs:extension>
</xs:complexContent>
</xs:complexType>
</xs:schema>
```

References

[ISO19115] ISO TC 211Geographic Information – Metadata – Implementation Specification ISO 19115 http://www.isotc211.org/pow.htm

[OGC99] The OpenGIS Abstract Specification. Topic 7: The Earth Imagery Case. OGC Document 99-107r4. http://www.opengis.org/public/abstract/99-107r4.pdf

[O&M] Simon Cox. [ed.], *Observations and Measurements*, OGC 03-022.

[SCS] Tom McCarty [ed.], *Sensor Collection Service*, OGC 03-023.

[SPS] Jeff Lansing [ed.], *Sensor Planning Service*, OGC 03-011r1.

[SeiCorp03] SeiCorp, Inc. (2003). Sensor Model Standardization: Frame Sensor Model Formulation, Draft Report prepared for NIMA, Contract NMA201-02-F-0220, May 8, 2003.

[ZI99] ZI Imaging (1999), ImageStation Open Photogrammetry Iniative; Open Photogrammetry Interface Specification, September 1999, doc no. ZI990014.

Additional SensorML information available at http://vast.uah.edu/SensorML