Open Geospatial Consortium Inc.

Date: 2004-11-01

Reference number of this OpenGIS® Project Document: **OGC 04-019r2**

Version: 1.0.0 beta

Category: OpenGIS Implementation Specification

Editor: Mike Botts
University of Alabama in Huntsville

Sensor Model Language (SensorML) for In-situ and Remote Sensors

Issue Name: [**Released as Recommended Paper for Public Discussion**. (meb, 2004-11-01)]

Issue Description: This document has been approved for public release as a Recommended Paper; the schema in this document have not gone through the approval process by OGC and are released here to allow public discussion. We feel that it is important to incorporate sensor knowledge and sensor model expertise that exist outside of the OGC membership. Therefore, SensorML has been released for public review. Comments and suggestions are strongly desired and should be sent to mike.botts@uah.edu or one can join the SensorML discussion listserv at http://mail.opengis.org/mailman/listinfo/sensorML

Document type: OpenGIS Publicly Available Recommendation Paper
Document subtype: Engineering Specification
Document stage: Final
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 initiatives and provides version 1.0 beta of the SensorML core specification.

ii. Submitting Organizations

This document is being submitted to the OGC by the following organizations:

a. University of Alabama in Huntsville

b. CSIRO Australia

c. Galdos Systems, Inc

iii. Submission Contact Point

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

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@nsstc.uah.edu
Alexandre Robin	University of Alabama in Huntsville	ESSC / NSSTC Huntsville, AL 35899	+01-256-961-7978	robin@nsstc.uah.edu
Ron Lake	Galdos Systems, Inc.	Suite 200, 115 West Pender Street, Vancouver, B.C. V6E 2P4	+01-604-484-2750	rlake@galdosinc.com
Simon Cox	CSIRO, Australia	PO Box 1130, Bentley WA 6102 Australia	+61-8-6436-8639	simon.cox@csiro.au

iv. 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.

2002-04-09	v0.04b	meb	throughout	Minor typo corrections. Incorporated edits from Carl Reed.
2002-04-09	v0.04b	meb	2.2.2	Changed *ObservablePropertyType* and *ObservableProperty* to *ObservedPropertyType* and *ObservedProperty* to match the schema terms. Changed UML to match.
2002-04-16	v0.04b	meb	throughout	Added suggested additions from Stefan Falke
2002-04-18	v0.04a	meb	throughout	Completed incomplete sections, added issue boxes, added references, added Annex A with excerpts from geometry sections of original SensorML design
2002-04-19	02-026 (v0.0.4c)	HAN	throughout	Final OWS-1 review and edit; minor changes. Produced OGC 02-026.
2002-04-22	02-026 (v0.04d)	meb	throughout	Updated Table of Contents, fixed pages where figure caption had been separated from figure; moved part 3 to Annex A (normative) and Annex A to Annex B (informative)
2002-08-14	02-026 (v0.04e)	meb	throughout Part 2.1	Updated Design specifications to reflect conceptual changes regarding coordinate systems. Previous design limited sensor models to local sensor coordinate space. New model allows for other coordinate spaces to support models such as RPC.
2002-08-14	02-026 (v0.05)	meb	throughout Part 2.2	Updated SensorML definition to reflect overall changes to schema as well as extensions to schema for support of dynamic, remote sensors
2002-08-14	02-026 (v0.05)	meb	throughout Part 2.2	Updated SensorML definition to reflect use of ISO 19115 schema for support of sensor and document metadata and history
2002-12-09	02-026 (v0.06)	meb	throughout	Updated several sections, completed incomplete sections on Sensor Models
2002-12-09	02-026 (v0.06)	meb	locatedUsing	Incorporated coordinate and coordinate operations schema from OGC Coordinate Transform Working Group
2002-12-09	02-026 (v0.06)	meb	measures	The measures and observationsLocated properties have been combined and significantly reorganized
2002-12-09	02-026 (v0.06)	meb	Sensor Models	The sensor model concept for geolocating observations have been completed and now utilize coordinates and coordinate operations schema from GML3; Models are defined and presented for scanners/profilers, opticalCameras, and Rapid Positioning Coordinates
2002-12-09	02-026 (v0.06)	meb	Appendices	Schema listing has changed and examples presented
2002-12-12	02-026 (v0.07)	meb	Throughout	Added issue boxes in relevant section warning of expected changes in GML and O&M schema
2002-12-12	02-026 (v0.07)	meb	Throughout Part 2	Refined schema to better conform to GML design patterns for xlink:href and for gml:id
9 January 2003	03-005	meb	Various	Made changes based on comments received through Dec 2002.
18 January 2003	03-005r1 (v0.8)	Harry Niedzwiadek	throughout	Final OWS 1.2 review and edit. Produced OGC 03-005r1.
28 March 2004	04-019	meb	throughout	This is a major revision of the SensorML documentation; The conceptual model section has been completely replaced and now supports UML model diagrams; changes have been made to the specification including addition of Component element as a base for Platform and Sensor; addition of new properties such as hasInterface, hasCapabilities; SensorModel base class and specific implementations of SensorModel have undergone significant change
25 May 2004	04-19r3	meb	throughout	Minor edits
25 May 2004	04-19r3	meb	9.1.1	Adding poweredBy property to base Component

24 Sept 2004	04-019r4	meb	throughout	Separated SensorML core components from SensorML extensions; SensorML extensions, such as SensorModels and RadiationMOdel for remote sensors have been moved to OGC document 04-068.
24 Sept 2004	04-019r4	meb	measures property	Measurand no longer value for measures property: sesnor measures Product; Products and ProcessModel developed to allow process chaining and lineage of Products. Sample is now derived from Product; ResponseModel derived from Process Model
24 Sept 2004	04-019r4	meb	parameters	Extensive development of parameters tobe used throughout schema, particularly within LocationModels and ProcessModels
24 Sept 2004	04-019r4	meb	LocationModel	Greatly simplified LocationModel base class as well as derived LocationModelTypes; removed hasActions and hasEvents from LocationModel
24 Sept 2004	04-019r4	meb	throughout	Minor edits throughout

The 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)]

v. Recommended Changes to the OpenGeospatial 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 and OWS 1.2 initiatives.

For SensorML extension components specific to particular sensor systems, such as remote sensors on dynamic platforms, see OGC Document 04-038.

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.

Introduction

OGC's Sensor Web Enablement (SWE) activity, which is being executed through the OGC Web Services (OWS) initiatives (under the Interoperability Program), is 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.

This document specifies SensorML. The other components are specified under separate cover. While SensorML serves as a component within the OGC Sensor Web Enablement framework, SensorML does not depend upon the presence of the other 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 an XML schema for defining the geometric, dynamic, and observational characteristics of a sensor.* Sensors are devices for the measurement of physical quantities. There are a great variety of sensor types from simple visual thermometers to complex electron microscopes and earth observing satellites.

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 "Observables". Each of these can be classified as an "ObservableType" and can be referenced in an "ObservablesDictionary". ObservableType definitions 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.
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.

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.)
- archive fundamental properties and assumptions regarding sensor.

To this end, the information provided by *SensorML* includes:

Observation characteristics

- Physical properties measured (e.g. radiometry, temperature, concentration, etc.)
- Quality characteristics (e.g. accuracy, precision)
- Response characteristics (e.g. spectral curve, temporal response, etc.)

Geometry Characteristics

- Size, shape, spatial weight function (e.g. point spread function) of individual samples
- Geometric and temporal characteristics of sensor and sample collections (e.g. scans or arrays) that are required for metric exploitation

Description and Documentation

- Overall information about the sensor
- History and reference information supporting the SensorML document.

The use of such a sensor model arises in several possible ways, including:

- The sensor description (model) is used to process the raw values obtained by the sensor element into a useful physical quantity and to provide support for georegistration of remotely sensed data.
- The sensor internally performs the "conversion" to physical quantities. The sensor description is used to construct the conversion equations. In some cases the sensor description might be used "post facto" to examine the data obtained from a sensor when it is behaving in an apparently "unreasonable" manner, or to assess inherent limits in the sensor's sensitivity and quality of measurements.
- The sensor does some internal processing, but additional external processing is required to take account of things that the "sensor" element and description may not be not aware of. The latter can include the measurement of other quantities, as well as the position and orientation of the sensor itself.

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 sensor models and mathematical sensor 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 ("sensor space"), the local frame for the associated platform ("platform space"), or a geographic coordinate reference frame ("object space").

For the case of rigorous sensor models, we allow one to separate the description of the sensor from that of its platform. In cases where confusion is likely, we will use the term "sensor system" to refer to the sensor and its associated platform(s). The platform is the carrier for the mounted sensor and is of major concern for mobile sensors. 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.

While the description of the platform is may not be a part of the sensor description, per se, it is of vital importance to our ability to georegister sensor measurements. Thus, a schema for defining platforms is provided in this document, as well.

As will be discussed further, *SensorML* is suitable for both in-situ and remote sensors, whether mounted to static or mobile platforms. The original version of *SensorML*, developed by the author before there was any involvement of OGC, focused primarily on defining the geometric and dynamic properties of remote sensors. Because of a focus on in-situ sensors within the OWS 1.1 initiative, the initial release of this document focused primarily on static, in-situ sensors. In line with the focus of OWS 1.2, the previous release refined and extended the initial schema and description to support dynamic, remote sensors, as well.

This release provides better documentation of the conceptual models on which the SensorML XML schema encoding is based, as well as provides some additions and slight modification to some areas of the schema. Within the area of Sensor Models used for geolocation of remote sensors, these areas have undergone more significant changes.

There is also a more distinct division between core SensorML classes and SensorML

extensions. Most of the classes that relate specifically to remote sensors or specific sensor types have been moved into SensorML extensions including all Sensor Models excluding the InSitu Model and most Response Models excluding the General Response Model.

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.

OpenGeospatial® *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 3.0, 18 December 2002, OGC document 02-023r4. Available to members [Online] under pending documents.
Coordinate Transformation Services Implementation Specification, Version 1.0, 12 January 2001, OGC document 01-009. Available [Online]: <http://www.opengis.org/techno/specs/01-009.pdf>. Also see: *Web Coordinate Transformation Service Draft Implementation Specification,* OGC document *02-061r2.* Available to members under pending documents.
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 can be defined.

4.2 Location Model
A model that allows one to locate objects within one CRS (source) relative to another CRS (reference).

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

4.4 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.

4.5 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.

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

4.7 Process Model
A process that takes one or more Products or observed Samples as input, and based on parameters and methodologies, outputs one or more new Products.

4.8 Product
A collection of values that are either observed by a sensor or derived from one or more Process Models.

4.9 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.

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

4.11 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.

4.12 Sensor Model

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

4.13 (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.

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
ows:*	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
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.

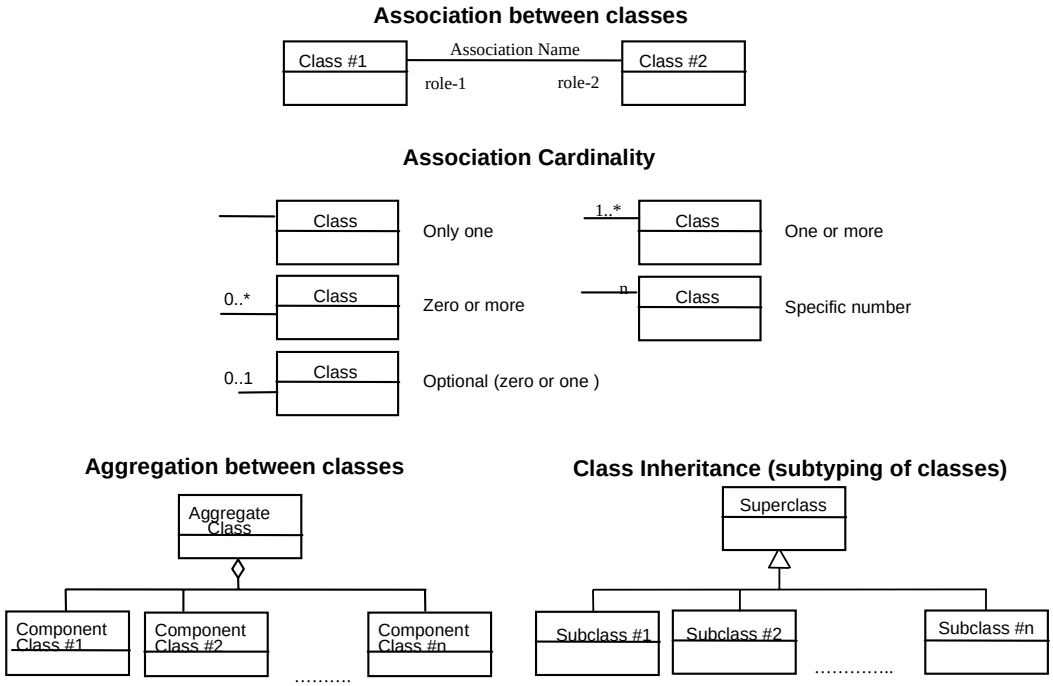


Figure 1.1 — UML notation

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 Earth observation 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, the availability of standard SensorML files allows for the development of general navigation software capable of geolocating and transforming any sensor data for which a SensorML file exists. Referred to as an Observation Dynamics Model (ODM), this concept is built around the availability of separate 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.

The second part of the Observation Dynamics Model concept consists of a generic software library for parsing these files, and calculating the transformations required to geolocate and perhaps process the sensor data. A significant advantage of the ODM concept for the developer and ultimately the end user, is that it provides a single source, single API, 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, the ODM allows for tools that can provide all the benefits of on-demand geolocation and mapping. As ODM capable 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, with an ODM, 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 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 SensorWeb 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 SensorWeb 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 Sensor Web 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.

8 Design Criteria and Assumptions for SensorML

8.1 Basic definition of a sensor

Sensors are devices for the measurement of physical quantities. 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.

A SensorML document can be considered a "living" description of a sensor. The SensorML document can begin as a template document, which is initially created using the sensor model design and is then appended or altered during the manufacturing, calibration, deployment, maintenance, and ultimately the removal of the sensor from service. Much of the specification of a sensor is shared by all sensor instances of the same model-number from the same manufacturer. This will typically include a description of measured properties, sample geometry, and the geometry and dynamics of any internal sampling arrays (such as scan patterns or frame camera properties). This initial template may include in addition some calibration parameters.

However, a SensorML document describing a particular sensor instance will acquire additional information that will distinguish it from other instances of the same model. In particular it may acquire unique identifiers such as ID and serial number. It will further be attached to some platform that will provide it with location and orientation within a known geospatial-temporal frame. In many cases the sensor instance will also have (for example) additional calibration information specific to the instance. As a sensor progresses through these stages, it's document will not only gain additional property information, but it will also record the changes to the sensor and the document itself through the inclusion of a history description.

8.2 Sensor Collection Concepts

Within SensorML, a sensor collection can itself be defined as a sensor. A sensor group can be of two types, herein referred to as “sensor group” and “sensor array”.

A sensor group is composed of multiple sensors that operate together to provide a collective observation or related group of observations. For example, a collection of temperature sensors (i.e. thermistors) can be used in a combined fashion to create a sensor that measures wind velocity and direction. Similarly, a group of individual sensors that measure different chemical species can be grouped as one sensor that provides “water quality”. A sensor group produces either a single observation or a composite observation of multiple related properties.

A sensor array is a set of sensors of the same type at different locations. These locations may be within a single sensor frame, a different location on a single mount, or on different platforms. A sensor array produces observations that are used to build a spatial coverage.

8.3 Relationship of the sensor to a platform

A sensor system is composed of the sensor and its 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 that share an association with one another.

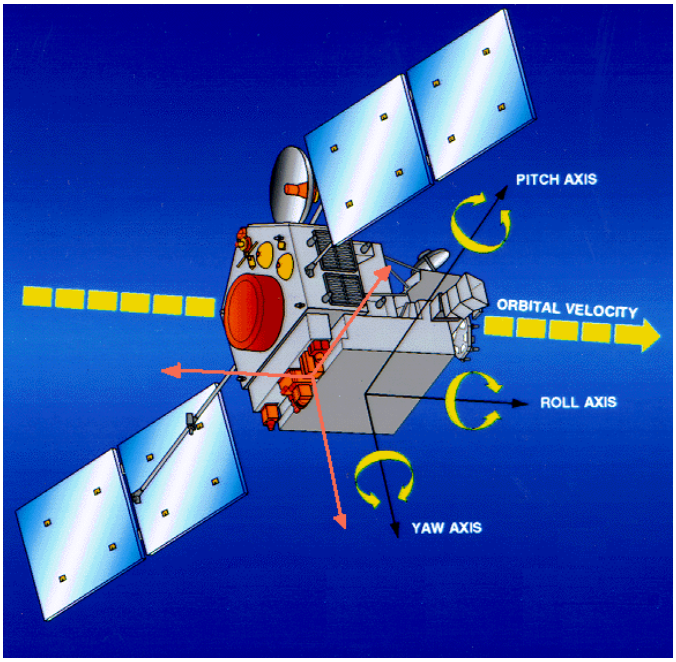


Figure 8.1. Relationship of sensor frame (pink) to the moving platform frame (black).

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.

8.4 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 through the use of defining CRSes and describing their relationships to one another. The relationship between CRSes can be accomplished either by describing a transform 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 Fixed (ECF) 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.

8.5 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. [OGC 03-022].

8.6 Sensor Response Characteristics

The response characteristics of a sensor determine how the sensor will react to a particular stimulus (i.e. Observable) 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 (e.g. survivable range and operational range).

While some of these parameters are relevant for a wide range of sensor types, other sensor types may require their own collection of response characteristics. For example, many of the response characteristics describing a radiation-based sensor (e.g. peak wavelength, band width, polarization angle, spectral response curve, etc.) will be different from those defining a water temperature probe. It is anticipated that while many sensors will be able to reuse some base level response characteristics, others may require appropriately defined response schemas for specifying different or additional parameters.

8.7 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.

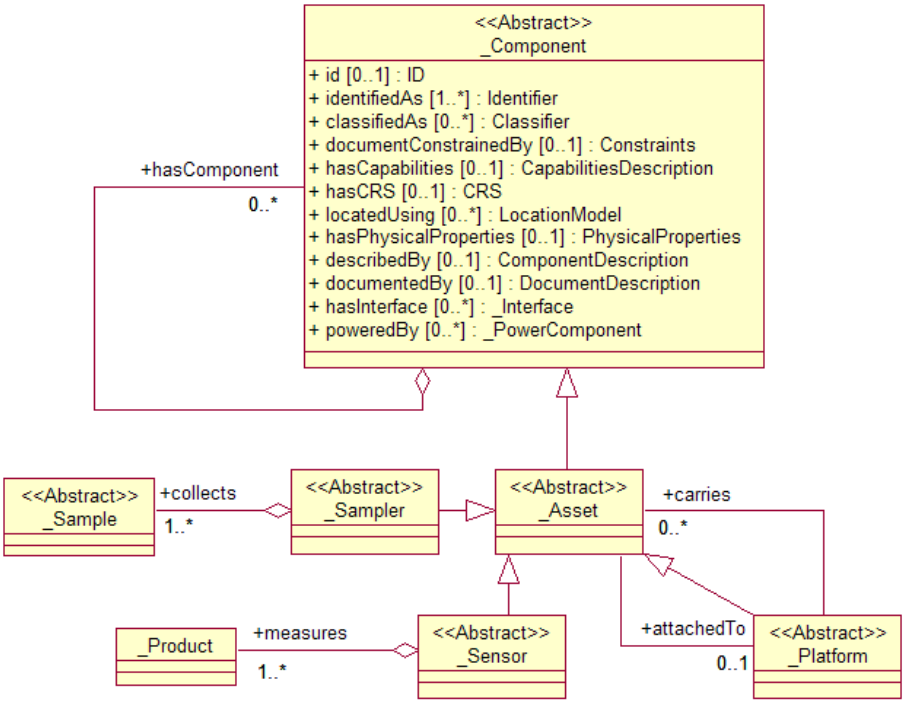
This sample frame can be related to the sensor's frame by either a simple transformation or in the case of collection of samples, by a more complex transformation involving arrays or scan patterns. Possible transformations for sample collections include unstructured grids, regular arrays, scanners, frame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

9 SensorML Conceptual Models

9.1 Basic Model

9.1.1 Fundamental Component Class

The basic framework for sensors and platforms is the abstract class *_Component* note: all abstract class names begin with an underscore). The *_Component* class utilizes a composite design pattern, such that any component can be an aggregate of other components which are described using the *hasComponent* property. The base *_Component* class has several optional properties, many of which take abstract classes for their value. These properties will be described individually below. The use of abstract classes throughout the SensorML specification allows for extensibility of the standard, as well as the ability to provide "plug-and-play" components for meeting a wide range of needs.



9.1.2 Asset base: _Sensor and _Platform

An asset is an independent component that may be tasked, and is described using the abstract class *_Asset*. *_Sampler*, *_Sensor* and *_Platform* are derived from *_Asset*. In addition to inheriting all *_Component* and *_Asset* properties, a *_Platform* has a *carries* property which takes another *_Asset (*platform or sensor) as its value. All assets including sensors and samplers can have an *attachedTo* property, which takes a *_Platform* as its value.

A sampler, which is described by the *_Sampler* class, is an asset that collects one or more samples, but does not take measurements on these samples. These samples could consist

of anything from which a measurement may be taken, such as a rock specimen, a volume of air, or a vial of ocean water. As will be discussed later, sample descriptions can include information regarding location and physical properties.

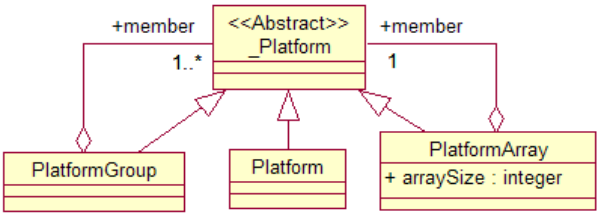
The *_Sensor* class has a *measures* property, which takes a *Product* as its value. The *Product* will be the main element that provides vital information needed for processing and geolocating the observation data. The observed or derived values can be included or linked to within the *Product* class.

9.1.3 ID, URI, and Linkable Properties

Implementations of sensor and platform descriptions should be able to reference property values that might exist in other locations in the document or even externally. To support this in SensorML, we have made extensive use of providing optional or required unique id's in most objects. These id's provide the ability in XML, for example, for properties to link to internally or externally defined values through the use of the *xlink:href* construct (which takes a URI for its value). Also, many properties (e.g. definitions, types, etc.) take a URI as their argument, which provides soft-typing for these values. We may wish to define harder-type definitions that can be linked to.

9.1.4 Asset Collections

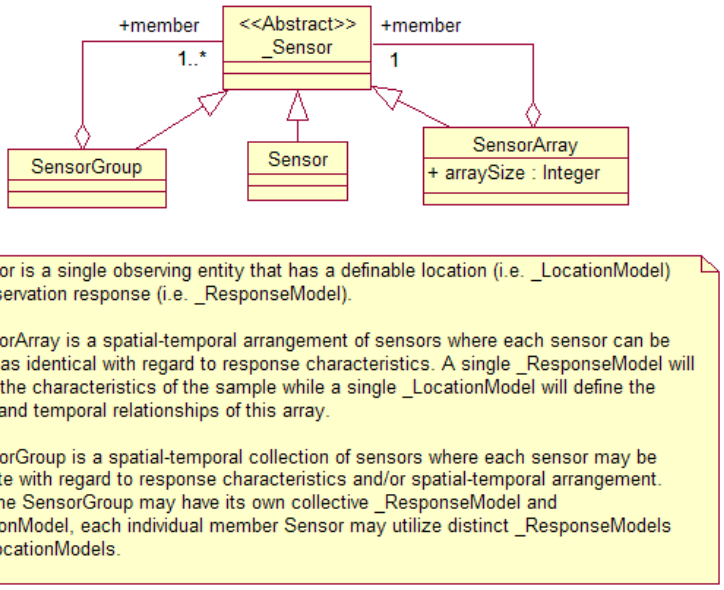
Several components, including *_Sensor*, *_Platform*, and later *_Sample* utilize a composite pattern. For example, *Platform* and platform collections (*PlatformGroup* and *PlatformArray*) are all of the abstract type *_Platform*. Thus, any platform collection is itself a platform, having all the properties of a platform. The difference between the two derived platform collections, is that a *PlatformGroup* can be a collection of disparate individual *Platform* members which can be described individually, but which can collectively act as one platform. In contrast, while a *PlatformArray* can also act as a single platform, it is a collection of identical platforms arranged within a definable spatial-temporal array.



The same composite pattern is used for *Sensor* and sensor collections (*SensorGroup* and *SensorArray*). A *Sensor* is an individual measurement device that measures a phenomenon at a single location, or possibly using an collection array (e.g. a charged couple device [CCD] with an array of pixels). A *SensorArray* is a collection of identical sensors with a definable spatial-temporal arrangement such that the collection provides a coverage of identical observations or an average observation of the network. Some may argue, and reasonably so, that an individual *Sensor* with a CCD is fundamentally no different that a *SensorArray* where each component of the CCD is considered a *Sensor*. In fact, the choice to use a Sensor or SensorArray may strictly be based on how much the document author wishes to disclose about the individual components. The basic models

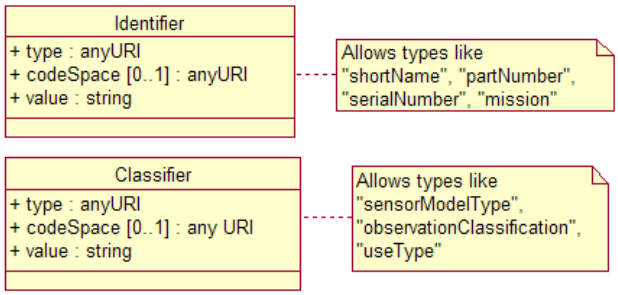
for *Sensor* and *SensorArray* will be identical, indicating the similarity between them.

A *SensorGroup* in contrast is a collection of disparate sensors which together measure a phenomenon based on the combined results of all the sensors. An example, might be a sensor group that provides a measure of water quality based on the combined observations from sensors that measure water temperature, pressure, percent dissolved constituents, and chemical concentrations. In this case, each member sensor can be considered and described as an individual *Sensor* with its own characteristic response and location models. Combined, they also measure a phenomenon that cannot be measured by a individual sensor.



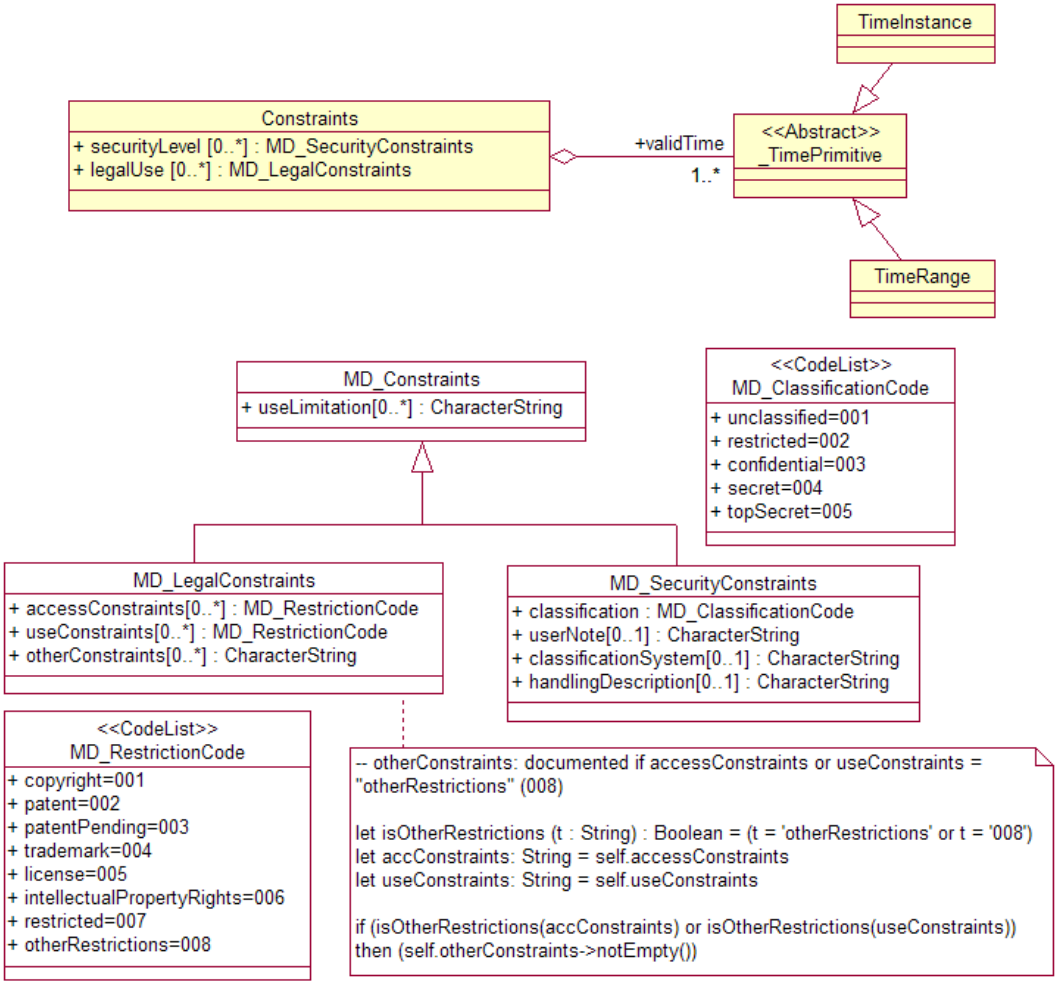
9.1.5 Identifier and Classifier

The *identifiedAs* takes property an *Identifier* as its value. In addition to text value, an *Identifier* includes a *type* definition (e.g. shortName, longName, serialNumber, noradID, missionID, etc.) and a *codeSpace* which takes a URI. The *codeSpace* identifies the authority source (typically an online Dictionary or Registry) for the value. Similarly, the *Classifier* object provides a means of providing several classification tags with the description. For instance, a single sensor might be classified as “remote observing”, “infrared detector”, “airborne”, “civilian”, and “atmosphere observing”. Such classifications could assist in sensor discovery.



9.1.6 Constraints

It is envisioned that SensorML descriptions will be self-describing. That is, information about the document can be contained in the document. The properties *constrainedBy* and *documentedBy* are two properties that provide such information. SensorML documents have three possible constraints, including *validTime*, the observation time over which the description is valid, as well as the *securityLevel* and *legalUse* constraints which take ISO-19115 constraint objects (shown below with white backgrounds) as their values.

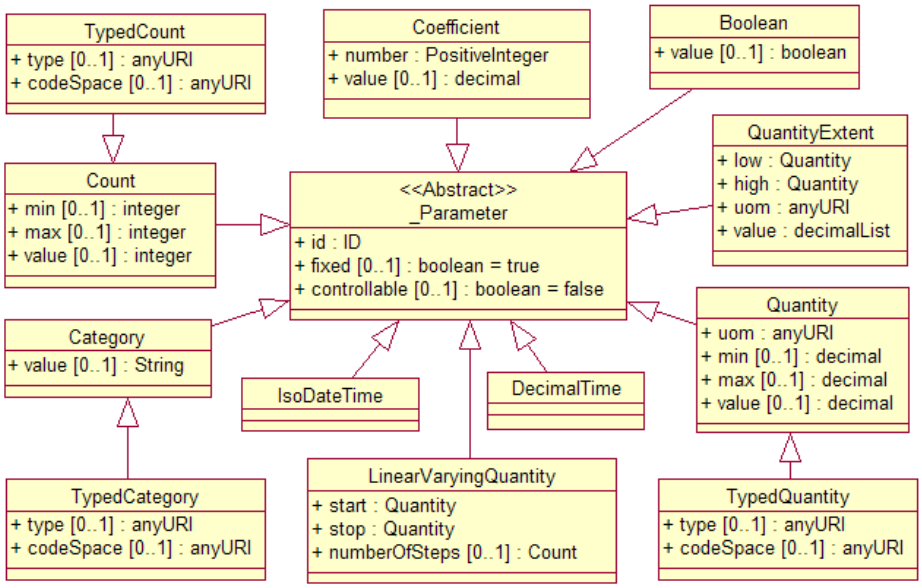


9.1.7 Parameters

_Parameter is an abstract class from which fundamental parameter values can be defined for various models. These parameters will be used extensively within *_ResponseModel* and *LocationModel*, to be described below. The base *_Parameter* class includes an *id*, and two boolean properties, *fixed* and *controllable* which indicate whether the parameter value is constant and whether it is a value that can be set through tasking, respectively. If these properties are absent, the default is that the parameter is fixed and is not controllable.

The most basic parameters include *Count*, *Category*, *Boolean*, *Coefficient*, and *Quantity*. There are also classes to support soft-typing of parameters at run-time using, for example, *TypedQuantity*, *TypedCount*, and *TypedCategory*. These typed classes include the additional properties, *type* and *codeSpace*, which allow one to specify a type definition (e.g. waterTemperature) and an authoritative dictionary, respectively.

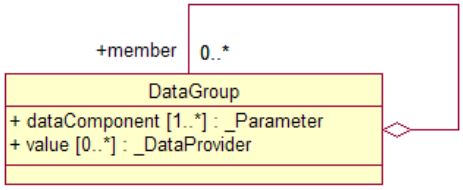
Parameters are used within various SensorML classes as values for various properties of the type, *dataComponent*, as well as *physicalPropertyTypes*. These might include, for instance, such properties as latitude, speed, stepAngle, wavelength, or timeStamp.



9.1.8 Data Groups

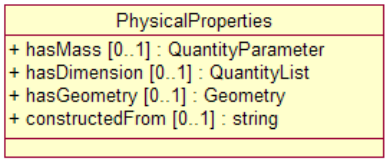
Parameters and their values can be collected within a *DataGroup* object. As will be discussed in more detail later, *DataGroup* provides a means for defining a collection of *dataComponents*, and optionally specifying either single values for these each component, or a collection of values through a *DataProvider*. Anticipated *DataProvider* types include tuples, streaming clusters, and data files. The *DataGroup* object uses the composite design such that any *DataGroup* can be composed of a collection of smaller *DataGroup* instances.

DataGroup instances will be used to provide input parameters for all models (e.g. *LocationModel*, *SensorModel*, *ProcessModel*, *ResponseModel*), as well as provide a means for defining product values.



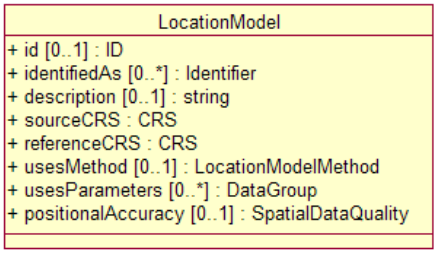
9.1.9 Physical Properties

A component can describe its physical properties using the hasPhysicalProperties property. These physical properties can include mass, dimensions, geometry, or construction material.



9.1.10 Coordinate Reference Systems and Location Models

Defining the state of platforms, sensors, and samples (e.g. location, orientation, etc.) within SensorML relies heavily on the concept of coordinate reference systems (CRS) and the relationships between them. Each component, platform, sensor, and sample has its own local CRS (specified by the *hasCRS* property), which must ultimately be related to some geodetic CRS (e.g. latitude, longitude, altitude). The process by which this occurs is dependent on the *LocationModel* used.



A location model typically describes the position (location and orientation) of a local coordinate reference system (*sourceCRS*) relative to an external CRS (*referenceCRS*). The base *LocationModel* class has a required *referenceCRS* and *sourceCRS*. The use of both of these CRS specifications allows one to define a series of locations or transforms, starting with the component's CRS and progressing through a collection of intermediate CRSes.

There are two important properties within the *LocationModel* class. These include:

usesMethod – defines the parameters and describes the assumptions and perhaps the algorithms that one might wish to implement in software

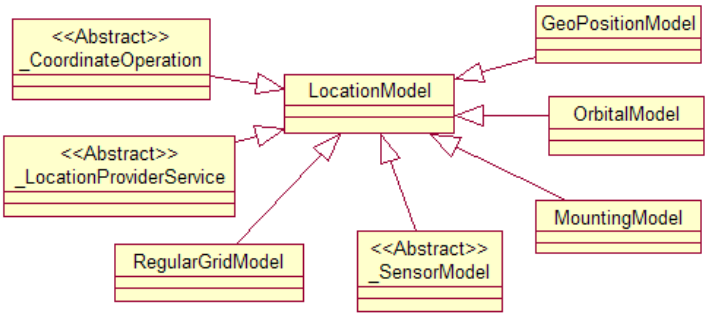
usesParameters – provides a list of *_Parameters* that are used by the model, their default values, and whether these parameters are variable and controllable;

These will be discussed in more detail in the *SensorModel* class definition, which derives from *LocationModel*.

The *LocationModel* will be used to derive several classes for specifying position. At a minimum, these will include *GeoPositionModel* (static or dynamic position relative to a geospatial CRS), *MountingPosition* (static or dynamic position relative to some engineering CRS), *RegularGridModel* (position of array members within a regularly arranged grid), and *OrbitalModel* (position based on Keplerian orbital elements).

For a simple static, in-situ sensor, an object's position can simply be expressed as a location within a geodetic CRS, using the *GeoPositionModel* class. Alternatively, a sensor's location and orientation might be related to it's platform's CRS using the *MountPosition* class, with the platform's CRS would be specified as the *referenceCRS*.

For rigorous geolocation of observations (i.e. pixels) from a dynamic remote sensor, the process is typically more complicated, involving a series of transformations from one CRS to another in order to determine the geodetic location of each observation. For dynamic assets, such as an aircraft or truck, the location model might include properties such as velocity, acceleration, angular velocity, and angular acceleration, in addition to time-tagged position and orientation.



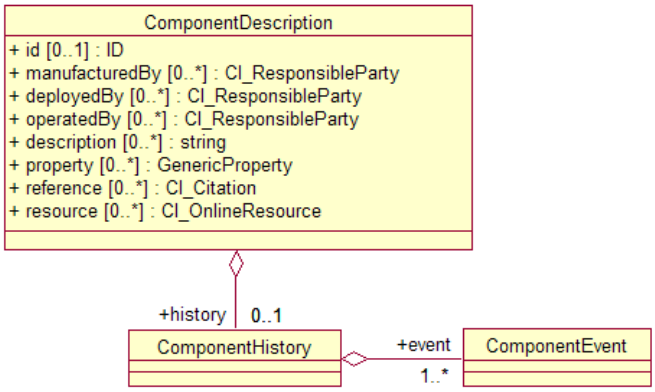
The state of an asset or sample can be expressed by either defining its position (e.g. location and orientation) within a given CRS, or by defining the transformation between the local CRS and the reference CRS (e.g. its translation and rotation). The *_CoordinateOperation* class provides a means for describing various transformations such as rotation, translation, and scale. Furthermore, it is envisioned that definitions of state might be included as part of the SensorML document or might be obtained from an on-line service (as provided for by the abstract *_LocationProviderService* class.

Finally, sensor models that provide the means to georeference observations from remote sensors, are considered as location models. Thus the class, *_SensorModel*, will be used to support models for sensors such as frame cameras, scanners, range finders, and profilers. The classes for these models will be included as part of SensorML extensions (described later in this document).

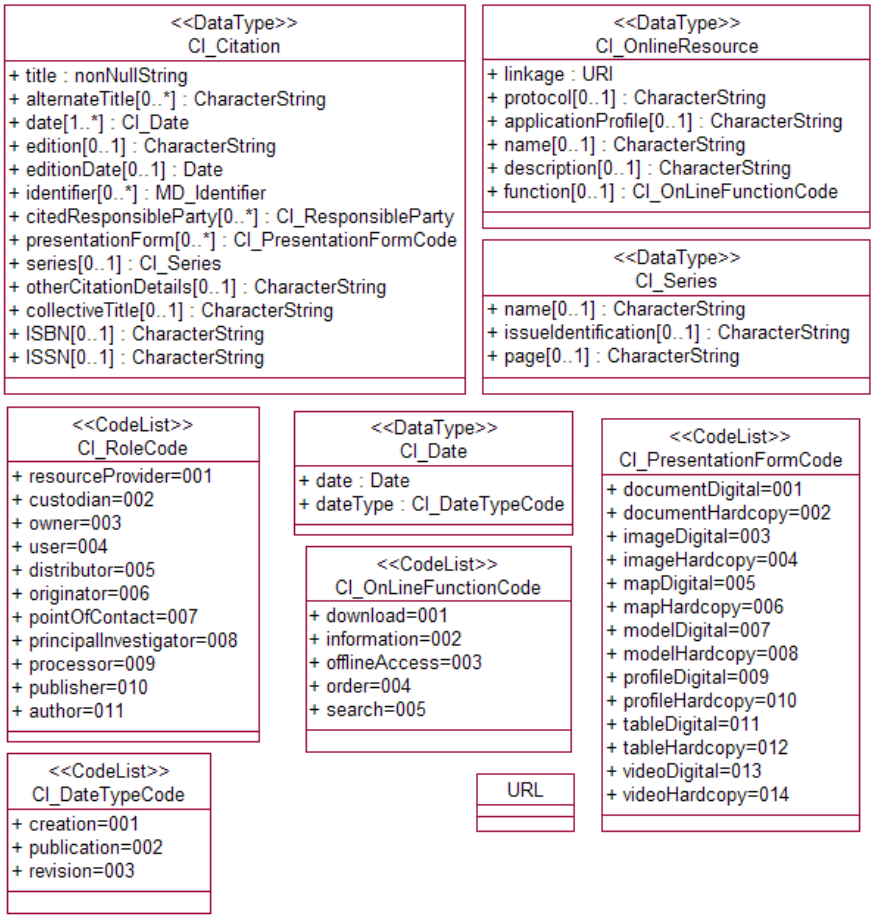
9.1.11 Component Descriptions

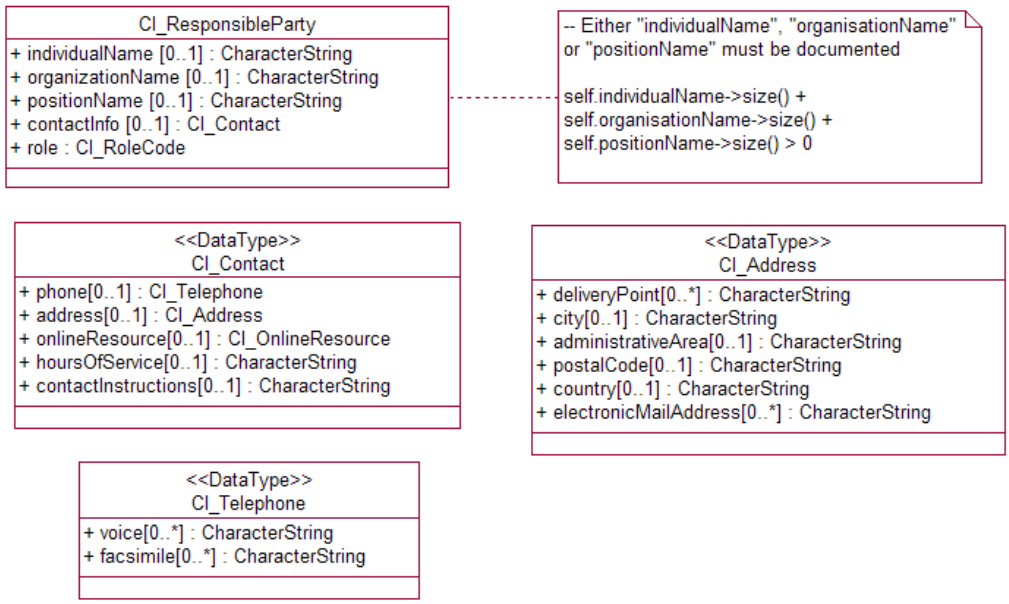
Within SensorML, information or metadata about the component can be provided by the *describedBy* property which takes a *ComponentDescription* as its value. This typically includes information that might be useful, but is generally not required for geolocating the platform or for processing or geolocating sensor observations (these are handled by the *locatedUsing* and *measures* properties, respectively). As will be discussed below, the

history of the asset can be provided within the *history* property of the *ComponentDescription* object. Many of the properties of the ComponentDescription class, as well as other classes to be defined below, take the ISO-19115 CI_ResponsibleParty as their argument.



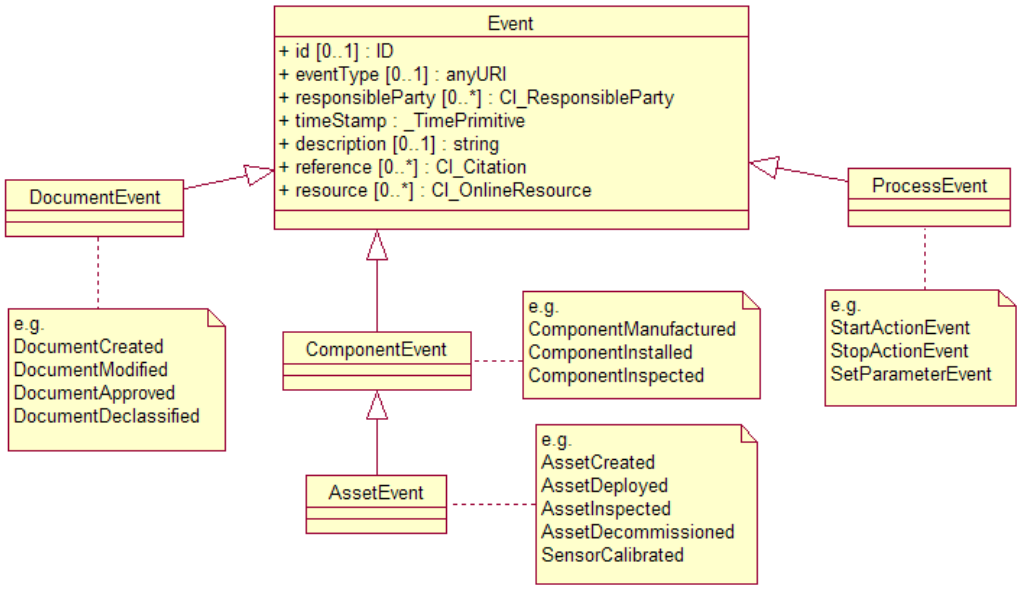
Additionally, these classes utilize the ISO-19115 classes for citing references or linking to online resources.





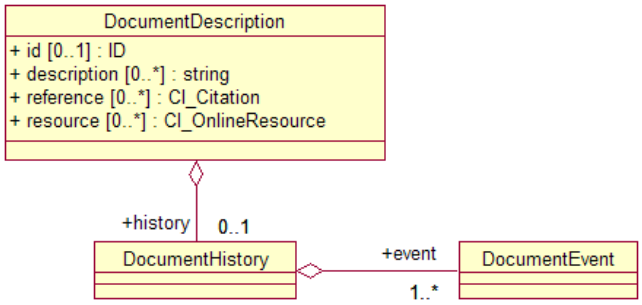
9.1.12 History and Events

Within SensorML, history of the component, or of the SensorML document itself, can be provided through a collections of *Event* objects. Events might for instance document the creation, inspection, calibration, deployment, or decommissioning of a sensor, or perhaps the creation, modification, or approval of the SensorML document itself. A SensorML document can therefore act as a "living" document, which can be appended as addition events happen to the component or the SensorML document describing that component. *ProcessEvent* will be discussed further as part of the *SensorModel* class.



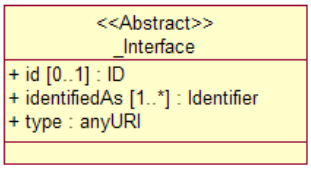
9.1.13 Document Metadata

A SensorML document can be self-describing through its *documentedBy* property. The *DocumentDescription* object includes the document's history, such as when and by whom it was created, modified, or validated. In addition, references can be provided to provide additional information or to support the information within the SensorML document.



9.1.14 Component Interface

Components, including sensors and platforms, can have various public or private interfaces through which communication to and from the component can occur. These interfaces can be used, for example, to output measurements and other information, or to provide input or tasking commands. For example, some sensors might support the IEEE-P1451 interface for low-level data transfer, or perhaps a specific web service interface for tasking the sensor or retrieving data. These can be described using a concrete implementation of an *_Interface* object for the *hasInterface* property. Specific interface definitions will not be a part of the core SensorML but will reside in SensorML extensions or within Application schemas.

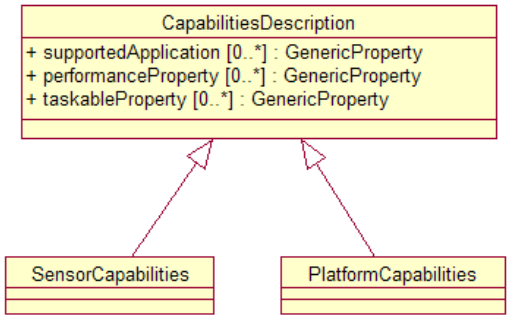


9.1.15 Component Capabilities Specifications

One important application for SensorML will be the discovery of observations from sensors, as well as the discovery of assets, themselves. While sophisticated software could mine a SensorML description to retrieve discoverable properties (e.g. from Response Models or SensorModel), some of the properties of the sensor are hidden too deeply within the document or are difficult to determine without significant effort. For example, the sensor model of a remote sensors will allow one to determine the expected

resolution of the observations on the ground, but executing sophisticated transforms to derive geolocation is not a typical operation for software associated with searchable registries or catalogs. Similarly, one can determine that a particular sensor measures radiation with a given infrared frequency, but it might not be obvious to registry software that this sensor is useful for observing wildfires.

For these reasons, SensorML includes a *hasCapabilities* property which allows one to provide information that might be particularly useful for the purpose of discovery. This property takes an abstract _Capabilities object as its value. This base class includes properties for *supportedApplication*, *performanceProperties*, and *taskableProperty*. It is expected that there will be derived classes or profiles that are specific to the type of assets.

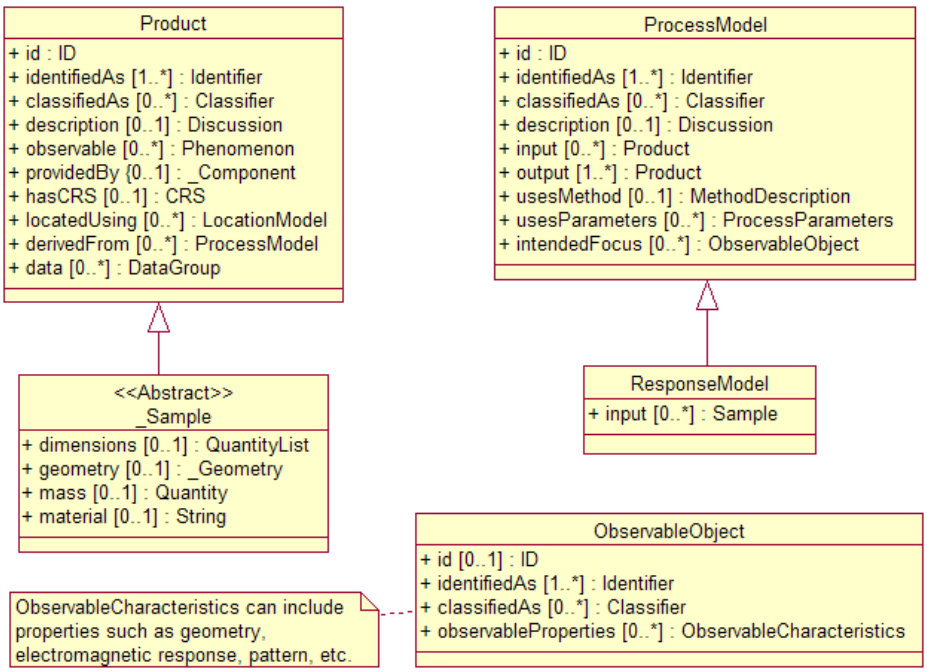


9.2 Sensor Measurement Characteristics

9.2.1 Product and Sample

The purpose of any sensor is to detect or provide measurements of particular phenomena or environmental properties. Each sensor is designed to measure one or more phenomena. This might, for example, include atmospheric temperature, the concentration of Calcium in water, the number of skunks in Madison County, or the amount of infrared radiation reflected from a surface. The values returned by these measurements are dependent on the sensing characteristics (or response model) of the sensor, as well as on the processes applied to the data.

A *Product* is the result of a *ProcessModel*, whether the process is the observation of a sample by a sensor, or computational processing to higher-level data products. The *ProcessModel* is specified within the *derivedFrom* property, while the *observable* property (which takes a *Phenomenon* as its value) provides a description of the type of product (e.g. temperature, voltage, wind speed). A *Product* can be associated with a location through its *locatedUsing* property, and can have its own *CRS* specified within *hasCRS*. The data associated with a *Product* is specified with the *data* property which takes a *DataGroup* as its value.



A *Sample* is a special type of *Product* in that it represents the actual entity of the environment on which observations were made and processed by a sensor and its computational components. In addition to the inherited properties of *Product*, a *Sample* can have properties related to its physical form, including *dimensions*, *geometry*, *mass*,

and *material*.

A sensor's sample is defined here as the subset of a physical entity on which an observation is made. A sample might include, for example, (1) a volume of air surrounding a thermometer, (2) the conic volume of atmosphere, vegetation, and ground that is observed and recorded within a pixel ("picture element") of an image, (3) a specimen of rock observed by a spectrometer, or (4) the volume of water contained within a bucket pulled up from a river for chemical concentration measurements. Spatially the sample might be 1D (e.g. an infinitely small point-source sample), 2D (e.g. a surface area of reflection), or 3D (e.g. a volume of water).

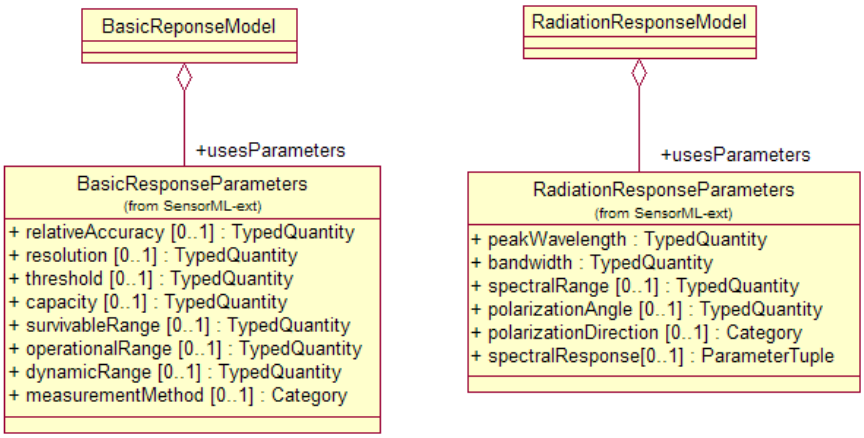
A sample's geometry, dynamics, and location can be defined within any coordinate system. Ultimately, the desire is to determine the location of the sample within some geographic coordinate reference system. However, particularly for remote sensors it is often desirable to provide descriptions of geometry, dynamics, and location within a local coordinate system and to utilize software to derive the georeferenced locations from the data.

9.2.2 Process Model and Response Model

As shown above, a sensor's *Product* is the result of a *ProcessModel*. The ProcessModel can be described by the *usesMethod* and *usesParameters* properties introduced for the *LocationModel* class. A *ProcessModel* takes a *Product* as its input and generates a new *Product* as its output. It does this by applying algorithms and methods described in *usesMethod*, based on parameters described in *usesParameters*. These parameters may be static or dynamic. A *ProcessModel* may also define the *intendedFocus* of the process in the form of an *ObservableObject* description. For example, if one describes the characteristics of a typical forest fire as seen by an IR sensor on a UAV (i.e. intensity, geometry, texture), this can be used for real-time data mining of the sensor's observations such that the output of a ProcessModel might be the detection of a fire, as well as its outer edge, intensity, and direction of motion.

The *ResponseModel* is a particular type of *ProcessModel* that describes the sensor's response to some phenomenon. That is, it describes the process by which a sensor measures a *Phenomenon* and converts that observation to an output *Product*. In essence, the *ResponseModel* provides information regarding the sensor's sensitivity to a phenomenon and the quality of its measurements. A *ResponseModel* thus takes one or more *Sample* instances as it input. The output *Product* of a *ResponseModel* can be at whatever data level one chooses to expose; for instance one may expose the raw voltages as an output *Product* and then perhaps the subsequent chain of processes (e.g. calibration and aggregation) that result in higher level products, or one may choose to specify a model that defines the higher-level product only (e.g. wind speed).

A *BasicResponseModel* which specifies common sensor parameters for sensitivity, accuracy, and survivability is an example of a derived *ResponseModel* that may be sufficient to meet the needs of a large number of in-situ sensor types.



However, specific sensor types might require more appropriately designed response parameters. One example would be a *RadiationResponseModel* which would be useful for many remote sensors which measure reflected or emitted radiation from a observable object. Other envisioned response models could include, for example, those for chemical concentrations, animal populations, or audio sources.

9.2.3 Stimulus

Many sensors, both active and passive, rely on a particular stimulus to act on the target (or possibly the sensor) before it is capable of measuring a response. For example, an Ampmeter requires a certain voltage to be applied across a wire before it can measure a response. A radiometer requires a certain spectrum of x-rays to illuminate a target before it can measure absorption or reflection by the target body. A LIDAR fires a timed pulse of light and measures the intensity and time of response. Even a passive remote sensor viewing the Earth typically requires solar illumination on the target before measuring the reflectance.

Within SensorML, a stimulus will be defined as a *Product* that can then serve as one of the input values of a sensor's *ResponseModel*. The parameters of the stimulus *Product* would define the characteristics of the active sensor or transmitter, or the assumed stimulus affecting a passive sensor.

9.3 Sensor Models

The term "sensor model" is commonly used within the remote sensing community to refer to the model used for geolocating observations measured some distance from the source of the phenomenon. 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 sometimes complex models for determining the location of individual observations.

Remote sensor types 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 model, Rational Polynomial Coefficients, and Replacement Sensor Model.

In any case, sensor models provide the fundamental information needed to locate or georegister observation to some local or external CRS. Within SensorML, a sensor model is defined as a simply a special case of a *LocationModel* and thus specifies the methods and parameters needed to go from the *sourceCRS* (typically a sample CRS) to the *referenceCRS* (e.g. geodetic, sensor, or platform CRS).

The actual definitions of these sensor models, as well as other SensorML schema derived to meet the needs of specific sensor communities, are considered as extensions to SensorML and thus not defined within this base document. These models and schema can be found in the OGC document 04-068.

10 SensorML XML Schema Encoding

10.1 Core Models

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 sensor and platform based upon the framework and rules provided by the XML Schema. It is also probable that either application schemas or XML template documents will be created for particular classes or models of sensors or platforms.

10.1.1 XML Encoding Conventions

The SensorML schemas are divided into two namespaces. The *sml* namespace is used for fundamental core SensorML elements that can be applicable for a wide range of sensors, including in-situ sensors. Elements that will probably be useful only to a select group of sensor and platforms, or only within particular sensor communities, are and will continue to be developed under the *sml-x* namespace.

The SensorML schema also utilizes components from two external XML schemas: the Geographic Markup Language (GML), version 3.0.1, under the *gml* namespace, and the ISO-19115 schema using the *iso19115* namespace.

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. decimal, boolean, 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, and codeSpace, or units of measure (uom), are encoded as attributes within an element. Most all 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 GML attribute group, *gml:AssociationGroup*, within property elements.

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

```
<operatedBy>
    <iso19115:CI_ResponsibleParty id="JoeBlow">
    ….
    </iso19115:CI_ResponsibleParty>
</operatedBy>
```

Or one can reference an object within the same document:

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

or an object within an external document:

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

One may also specify the expected schema for the object using the *gml:remoteSchema* attribute:

```
<attachedTo xlink:href="http://myDomain.com/platforms.xml#nsstc01"
        gml:remoteSchema="http://vast.uah.edu/SensorML/schema/platform.xsd"/>
```

10.1.3 Compactness and Completeness

Since all of the properties of a *_Component*, excluding *identifiedAs*, are optional, an *Component* instance can be very simple:

```
<Component>
    <identifiedAs>
        <Identifier>Wind Cup</Identifier>
    </identifiedAs>
    <hasPhysicalProperties>
        <PhysicalProperties>
            <material>
                <StringParameter>Polycarbonate</StringParameter>
            </ material >
        </PhysicalProperties>
    </hasPhysicalProperties>
</Component>
```

Similarly, one may choose to provide minimal information regarding a sensor:

```
<Sensor id="davis7911">
    <identifiedAs>
        <Identifier type=" http://myTermsDictionary.xml#longName">
            Davis Anemometer 7911
        </Identifier>
    </identifiedAs>
```

```
    <locatedUsing>
        <GeoPositionModel>
            <description>
                <Discussion>Location of weather station on roof of the National Space Science
                    and Technology Center Building at 320 Sparkman Dr., Huntsville, AL 35805
                </Discussion>
            </description>
            <sourceCRS xlink:href="http://myPlatform.xml#platformCRS"/>
            <referenceCRS xlink:href=" http://myCRSDictionary.xml #EPSG:4329"/>
            <usesParameters>
                <LatLonAlt id="platformLocation">
                    <latitude>
                        <Quantity uom="http://myUnitsDictionary.xml#degrees">34.72450</Quantity>
                    </latitude>
                    <longitude>
                        <Quantity uom="http://myUnitsDictionary.xml#degrees">-86.94533</Quantity>
                    </longitude>
                    <altitude>
                        <Quantity uom=" http://myUnitsDictionary.xml#degrees">20.1169</Quantity>
                    </altitude>
                </LatLonAlt>
            </usesParameters>
        </GeoPositionModel>
    </locatedUsing>
    <measures>
        <Product id="windSpeed">
            <identifiedAs>
                <Identifier type="shortName">Wind Speed</Identifier>
            </identifiedAs>
            <observable xlink:href=" http://myTermsDictionary.xml#shortName >
                <Phenomenon>wind speed</Phenomenon>
            </observable>
            <derivedFrom>
                <ResponseModel>
                    <usesParameters>
                        <BasicResponse>
                            <resolution>
                                <TypedQuantity uom="http://myUnitsDictionary.xml#metersPerSecond">
                                    0.1
                                </TypedQuantity>
                            </resolution>
                            <samplePeriod>
                                <Quantity uom=" http://myUnitsDictionary.xml#seconds">2.25</Quantity>
                            </samplePeriod>
                        </ BasicResponse>
                    </usesParameters>
                </ResponseModel >
            </ derivedFrom >
        </ Product >
    </measures>
</Sensor>
```

However, SensorML also supports much more complete descriptions of sensor systems as illustrated by the UML models described in the previous sections. For illustration purposes, we will utilize pieces of a wind sensor (i.e. anemometer) description for examples of property encodings within the following sections.

It is important to understand that it is possible within SensorML to be both complete and compact by making extensive use of links to external documents and services within the sensor or platform description. For instance, SensorML documents describing responsible parties, reference materials, sensor history, CRS definitions, and so forth can be either

included in the sensor document or can be provided online and linked to within the sensor document. Likewise parameters that are highly dynamic, such as positional information or camera settings can be provided as either within the document or through URI links to external documents or services. Furthermore, the creator of documents that describe highly dynamic sensors can choose to "granularize" the sensor descriptions such that a given instance document is only valid over a specific, narrow time period (indicated using the *validTime* property within the *Constraints* object).

10.1.4 The Base Schema and Abstract Types

Unless one is defining an object to be re-used as a property value within SensorML, one will typically use either *Component*, *Platform*, *PlatformArray*, *PlatformGroup*, *Sensor*, *SensorArray*, or *SensorGroup* as the root element in a SensorML instance document. While *Component* is defined within the component.xsd schema, sensor and platform types are defined within sensor.xsd and platform.xsd schemas, respectively.

The basic elements shared by most SensorML documents are defined with the base.xsd document within the core SensorML namespace. Many of the elements defined in the SensorML schemas define either abstract objects or substitution group base elements. SensorML makes extensive use of abstract types and substitution groups in order to provide maximum flexibility for extension of the standard and to support "plug-n-play" objects.

For clarification, an abstract object provides base properties for any concrete element that extends or restricts it. Any element that is of a particular substitution group merely states that it is an object of that type and can therefore be used as a value for any property requiring that type. For example, any property that takes *Component* as its value, can utilize either *Component*, *Sensor*, *SensorGroup*, *Platform*, or *PlatformGroup* as its value, since all of these, either directly or through inheritance, are of the substitution type *Component*.

In addition, the component.xsd schema defines the concrete *Component* object from which all platform and sensor elements are derived, as well as the objects utilized with the *Component* description.

A typical instance document describing a Sensor will look like:

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Michael E Botts (University of Alabama in
Huntsville) -->
<Sensor xmlns="http://www.opengis.net/sensorML"
  xmlns:gml="http://www.opengis.net/gml"
  xmlns:iso19115="http://www.isotc211.org/iso19115/"
  xmlns:xlink="http://www.w3.org/1999/xlink"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:schemaLocation="http://www.opengis.net/sensorML
      http/vast.uah.edu/SensorML/schema/sensorML/sensor.xsd"
  version="1.0">
  id="mySensor">
```

The sensor.xsd schema imports most dependent schemas internally, so it is typically only necessary to import extension schemas that might provide sensor or community-specific

definitions (e.g. specific sensor or response model definitions). Similarly, platform.xsd typically provides all core schemas needed for *Platform* definitions.

10.1.5 Platform and Sensor Identification and Classification

The *Identifier* and *Classification* objects provide a means for both human and software to quickly recognize names, types, and applications of the *Asset* or *Component*. Typical name types might include, for instance, a shortName (for use perhaps in menus and data trees), longName (a more complete name), missionID, and modelNumbers (OEM or community number for this asset). Examples of type attribute values for *Classifiers* might include sensorType, application, phenomenon, or accessibility (i.e. private or public). The *identifiedAs* and *classifiedAs* properties provide two of several SensorML components from which relevant information can be mined to support asset discovery and cataloging.

The *type* attribute in *Identifier* and *Classifier* allows one to define its relevance, while the optional *codeSpace* attribute defines under whose authority the value is defined. Typically, the *codeSpace* attribute will point to some authoritative dictionary within which the value is defined. For the example anemometer, the following might apply:

```
<identifiedAs>
    <Identifier type="longName">Davis Anemometer 7911</Identifier>
</identifiedAs>
<identifiedAs>
    <Identifier type="modelNumber">7911</Identifier>
</identifiedAs>
<classifiedAs>
    <Classifier type="sensorType"
                codeSpace="http://vast.uah.edu/SensorML/sensorDictionary.xml">
        anemometer
    </Classifier>
</classifiedAs>
<classifiedAs>
    <Classifier type="application"
                codeSpace="http://vast.uah.edu/SensorML/applicationDictionary.xml">
        weather
    </Classifier>
</classifiedAs>
<classifiedAs>
    <Classifier type="phenomenon"
                codeSpace="http://vast.uah.edu/SensorML/phenomenonDictionary.xml">
        wind speed
    </Classifier>
</classifiedAs>
<classifiedAs>
    <Classifier type="phenomenon"
                codeSpace="http://vast.uah.edu/SensorML/phenomenonDictionary.xml">
        wind direction
    </Classifier>
</classifiedAs>
```

10.1.6 Constraints

SensorML instance documents can be constrained by their time validity, security level, or legal use. In this encoding, time constraints utilize the abstract *gml:_TimePrimitive*

object, which be either a *gml:TimePeriod* or *gml:TimeInstant*. The anemometer example shows the use of a time period:

```
<documentConstrainedBy>
    <Constraints>
        <validTime>
            <TimeExtent>
                <startTime>
                    <IsoDateTime id="launchDate">1998-05-13</IsoDateTime>
                </startTime>
                <stopTime>
                    <IsoDateTime>currentTime</IsoDateTime>
                </stopTime>
            </TimeExtent>
        </validTime>
    </Constraints>
</documentConstrainedBy>
```

An example of a validTime consisting of a time instance is:

```
<validTime>
    <DecimalTime id="#julianTime" origin="1970-01-01T00:00:00" uom="#seconds" >
        50234.0293
    < DecimalTime>
</validTime>
```

The security level and legal use constraints utilize the MD_SecurityConstraints and MD_LegalConstraints objects defined within the ISO-19115 schema.

10.1.7 Parameters

A wide variety of simple parameters and parameter types are defined within the parameter.xsd schema. These are used extensively within sensor, response, and location models, as well as within *Product* and *Sample* objects. They are typically values of association elements of the type *_propertyType*. The most basic parameters and parameter groups are defined within the parameters.xsd schema, while those specific to location models, sensor models, or response models may be defined in those respective schemas, namely LocationModel.xsd, SensorModel.xsd, and ResponseModel.xsd.

Fundamental parameters and parameter types include Quantity, Count, Boolean, Category, String, and URI. These reflect the classification of values defined with the OGC Observations and Measurement (O&M) schema, although the SensorML parameters are not currently derived from them (see related Issue box). As has been defined within the O&M schema, most of the basic parameters have associated Typed equivalents in which one can define a *type* subclassification along with an authoritative *codeSpace* attribute.

One may utilize parameters as values “hard-typed” properties, such as *focalLength*, *distance*, *mass*, and so forth if they derive these from *_propertyType* and define them well within a *usesMethod* property. Therefore, if one chooses to define f*ocalLength* within a model method definition, derive it from *_propertyType* is, and use the hard-typed definition as in:

```
<focalLength>
    <Quantity id=”fov” uom="http://myUnitsDictionary.xml#mm" > 0.1</ Quantity>
</focalLength>
```

Otherwise, one may choose to use the *dataComponent* property which is derived from *_propertyType* and rely on “soft-typing” at run-time using the appropriate typed parameter (e.g. *TypedQuantity*):

```
<dataComponent>
    <TypedQuantity id=”fov” uom=" http://myUnitsDictionary.xml#mm"
          axis="http://myObservablesDictionary.xml#focalLength">0.1</TypedQuantity>
</dataComponent>
```

Typically, within SensorML, it is anticipated that *Parameters* within models will be hard-typed, with their definitions well defined within the model description (using the *usesMethod* property). In contrast, many or all parameters within *Products* may typically be soft-typed as values for a *dataComponent* property since these may be specific to particular product types or user communities and thus need to be defined within online dictionaries.

All models within SensorML utilize parameters and parameter groups. These can be specified as being constant or variable. In addition, one can specify whether a variable parameter can be controlled through tasking commands.

The attributes for a parameters include:

id – a unique id of the type *xs:ID*

fixed – a boolean designating whether the parameter has a constant or variable value (defaults to true)

controllable – a boolean designating whether the value can be altered using commands to the component (defaults to false)

Numerical parameters (e.g. parameters such as *Quantity*, or *Count*) have added attributes for *min* and *max* which designate the range of possible values, if the parameter is variable. In cases where *min* and *max* are used, the value given for the object is the current default value. Parameters of *QuantityType* also have the units of measure attribute, *uom*, which take a URI. An example of a parameter based on *QuantityType* would be:

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

10.1.8 DataGroup, TupleData, and _DataProvider

The *DataGroup* class provides a powerful and flexible means of listing properties and data components, as well as their values. The *DataGroup* is used as a value for the *usesParameters* property within all models, and for the value of the *data* property of *Product* and its derivatives. When combined with the *TupleData* or *_DataProvider* classes, *DataGroup* supports not only the ability to provide single static values, but also an efficient means for supporting highly dynamic data values.

As an example we will use *GeoPosition* which is derived from *DataGroupType* and

specifies several required and optional *dataComponent* properties. As with traditional XML, one may specify the values for the each data component between the element's opening and closing brackets, as shown below. For static data, this should be the preferred method of specifying values. For dynamic data, one may also utilize this construct by providing several such blocks of Geoposition, each with their time specification.

```
<GeoPosition>
    <time>
        <DecimalTime origin="1998-04-01T00:00:00.0Z" uom="http://myUnitsDictionary#seconds">
            50561.0
        </DecimalTime>
    </time>
    <latitude crs="http://myCrsDictionary#EPSG4329" axisCode="Y">
        <Quantity uom="http://myUnitsDictionary#degrees">36.9274</Quantity>
    </latitude>
    <longitude crs="http://myCrsDictionary#EPSG4329" axisCode="X">
        <Quantity uom="http://myUnitsDictionary#degrees">-76.1564</Quantity>
    </longitude>
    <altitude crs="http://myCrsDictionary#EPSG4329" axisCode="Z">
        <Quantity uom="http://myUnitsDictionary#feet">3347.0</Quantity>
    </altitude>
    <trueHeading crs="http://myCrsDictionary#EPSG4329" axisCode="Z">
        <Quantity uom="http://myUnitsDictionary#degrees/>-144.86</Quantity>
    </trueHeading>
    <pitch crs="#platformCRS" axisCode="X">
        <Quantity uom=" http://myUnitsDictionary#degrees">-54.69</Quantity>
    </pitch>
</GeoPosition>
```

However, for more efficiency in data volume and parsing speed, one may first specify the dataComponents expected and then provide the time-dependent values as a tupleData block as shown below. In this case, each tuple is separated by "white space", while values within each tuple are separated by commas. The order of the values within a tuple must follow the order of dataComponents specified and must have the same number of values.

```
<GeoPosition>
    <time>
        <DecimalTime origin="1998-04-01T00:00:00.0Z" uom="http://myUnitDictionary#seconds"/>
    </time>
    <latitude crs="http://myCRSDictionary#EPSG4329" axisCode="Y">
        <Quantity uom="http://myUnitsDictionary#degrees"/>
    </latitude>
    <longitude crs="http://myCRSDictionary#EPSG4329" axisCode="X">
        <Quantity uom="http://myUnitsDictionary#degrees"/>
    </longitude>
    <altitude crs="http://myCRSDictionary#EPSG4329" axisCode="Z">
        <Quantity uom="http://myUnitsDictionary#feet"/>
    </altitude>
    <trueHeading crs="http://myCRSDictionary#EPSG4329" axisCode="Z" direction="-">
        <Quantity uom="http://myUnitsDictionary#degrees"/>
    </trueHeading>
    <pitch crs="#platformCRS" axisCode="X">
        <Quantity uom="http://myUnitsDictionary#degrees"/>
    </pitch>
    <values>
        <TupleData>
            50561.0,36.9274,-76.1564,3347.0,-144.86,-54.69
            50562.0,36.9276,-76.156,3355.0,-145.33,-52.34
```

```
                50563.0,36.9278,-76.1556,3361.0,-146.0,-50.2
                50564.0,36.9281,-76.1551,3368.0,-145.78,-47.95
                50565.0,36.9284,-76.1548,3374.0,-146.43,-46.68
                50566.0,36.9287,-76.1544,3383.0,-146.35,-45.8
                50567.0,36.929,-76.1541,3389.0,-146.43,-44.82
                50568.0,36.9293,-76.1538,3395.0,-145.66,-44.43
                50569.0,36.9296,-76.1535,3401.0,-144.45,-43.26
                50569.0,36.9299,-76.1532,3406.0,-143.56,-42.68
                50571.0,36.9303,-76.1529,3411.0,-143.32,-42.58
                50572.0,36.9306,-76.1527,3414.0,-142.46,-41.7
                50572.0,36.931,-76.1525,3418.0,-141.64,-40.62
                50573.0,36.9313,-76.1523,3422.0,-141.0,-39.45
                50574.0,36.9317,-76.1522,3424.0,-140.53,-38.48
                50576.0,36.9321,-76.152,3429.0,-140.06,-37.6
                50577.0,36.9325,-76.1519,3433.0,-139.57,-36.52
                50577.0,36.9329,-76.1518,3436.0,-139.22,-35.45
                50579.0,36.9333,-76.1517,3441.0,-138.75,-34.37
                50579.0,36.9338,-76.1516,3446.0,-138.33,-33.2
                50581.0,36.9341,-76.1516,3451.0,-137.54,-31.84
                50582.0,36.9346,-76.1516,3457.0,-136.69,-30.76
            </TupleData>
        </values>
    </GeoPosition>
```

Additionally, as shown below, one may choose to receive values through some other object derived from the _DataProvider class. For example, in order to link to a real-time stream of data clusters coming off of a sensor, one could utilize a TransducerML data provider, to be defined in a separate document. Other anticipated data providers would include objects based on OGC's Observations and Measurements, the Earth System Markup Language, or a compatible web service.

```
    <GeoPosition>
        <time>
            <DecimalTime origin="1998-04-01T00:00:00.0Z" uom="http://myUnitDictionary#seconds"/>
        </time>
        <latitude crs="http://myCRSDictionary#EPSG4329" axisCode="Y">
            <Quantity uom="http://myUnitsDictionary#degrees"/>
        </latitude>
        <longitude crs="http://myCRSDictionary#EPSG4329" axisCode="X">
            <Quantity uom="http://myUnitsDictionary#degrees"/>
        </longitude>
        <altitude crs="http://myCRSDictionary#EPSG4329" axisCode="Z">
            <Quantity uom="http://myUnitsDictionary#feet"/>
        </altitude>
        <trueHeading crs="http://myCRSDictionary#EPSG4329" axisCode="Z" direction="-">
            <Quantity uom="http://myUnitsDictionary#degrees"/>
        </trueHeading>
        <pitch crs="#platformCRS" axisCode="X">
            <Quantity uom="http://myUnitsDictionary#degrees"/>
        </pitch>
        <values>
            <TMLDataStream>
                ...

            </TMLDataStream>
        </values>
    </GeoPosition>
```

As discussed in the Data Parameter section one may typically wish to utilize hard-typed *propertyTypes*, when the *DataGroup* is used within a model object (e.g.

LocationModel, *ResponseModel*, etc.), since these properties can be well-defined within the *usesMethod* property value. However, when used within Product, one may wish to use a soft-typed form of dataComponent, where a parameter attributes provide a link to its definition within an online dictionary. Further examples will be provided in the later section on Products.

10.1.9 Coordinate Reference Systems

In order for sensor observations to be useful, they generally need to be referenced to some appropriate coordinate reference system (CRS). In some cases, such as for a simple in-situ sensors, observations can be located directly to a geographic reference system, such as latitude-longitude-altitude, Earth-Centered-Earth-Fixed, or Earth-Centered-Inertial. For other sensors, particularly remote sensors, it is often useful to reference its geometric properties first to a relevant engineering coordinate reference system, such as one associated with the sensor itself or the sensor's platform. Then through a series of state (location and orientation) definitions or transformations, these observation can ultimately be related to a desired geographic reference system

SensorML utilizes definitions for coordinate systems, coordinate reference systems, and coordinate operations provided by the GML3.0 schemas: coordinateSystems.xsd, coordinateReferenceSystems.xsd, and operations.xsd, respectively. In addition to using spatial coordinate systems, SensorML utilizes GML definitions for temporal coordinate systems. These allow, for example, one to reference scan characteristics or sampling times to local and geographic temporal frames.

All *Component*, *Sensor*, *Platform*, *Product*, and *Sample* objects include a *hasCRS* property which takes a *gml:EngineeringCRS* for its value. In addition, they also include a *locatedUsing* property that accepts any object of the *_LocationModel* type. Both the *LocationModelType* and *LocationModelGroupType* classes, from which all concrete *_LocationModel* classes are derived, include the properties *hasSourceCRS* and *hasReferenceCRS*, which take a *gml:_CRS* as their value.

A CRS includes two parts: a Coordinate System (CS) that defines the axes used in the CRS, and a Datum definition which ties defines an origin for the CRS. A new CRS can be defined inline within the SensorML instance document or by linking to predefined Coordinate System and Datum instances.

Within SensorML, platform positions are typically related to a local coordinate system whose origin will be at the current location, such as a North-East-Down or East-North-Up Coordinate System:

```
<!-- Local North-East-Down (NED) CS -->
<gml:CartesianCS gml:id="NED-CS">
    <gml:remarks>defines a planetary local cartesian CS with the Z axis being in the vertical Down
     direction (orthogonal to the local ellipsoid), X axis being orthogonal to Z and in the direction of true
     North, and Y completing the right hand coordinate system (i.e. Z cross X) and pointing
     East</gml:remarks>
    <gml:csName>North-East-Down CS</gml:csName>
    <gml:csID>
        <gml:code>NED-CS</gml:code>
    </gml:csID>
    <gml:usesAxis>
        <gml:CoordinateSystemAxis gml:id="NED-North" gml:uom="#meters">
```

```
            <gml:axisName>True North</gml:axisName>
            <gml:axisID>
                <gml:code>X</gml:code>
            </gml:axisID>
            <gml:axisAbbrev>N</gml:axisAbbrev>
            <gml:axisDirection>
                axis orthogonal to the Down axis and pointing in the direction of planetary true North
            </gml:axisDirection>
        </gml:CoordinateSystemAxis>
    </gml:usesAxis>
    <gml:usesAxis>
        <gml:CoordinateSystemAxis gml:id="NED-East" gml:uom="#meters">
            <gml:axisName>East </gml:axisName>
            <gml:axisID>
                <gml:code>Y</gml:code>
            </gml:axisID>
            <gml:axisAbbrev>E</gml:axisAbbrev>
            <gml:axisDirection>
                axis orthogonal to North and Down axis and pointing in the East direction
            </gml:axisDirection>
        </gml:CoordinateSystemAxis>
    </gml:usesAxis>
    <gml:usesAxis>
        <gml:CoordinateSystemAxis gml:id="NED-Down" gml:uom="#meters">
            <gml:axisName>Down</gml:axisName>
            <gml:axisID>
                <gml:code>Z</gml:code>
            </gml:axisID>
            <gml:axisAbbrev>D</gml:axisAbbrev>
            <gml:axisDirection>
                axis orthogonal to the local planetary ellipsoid, and in the Down direction (i.e. nadir or
                the direction of gravitation acceleration)
            </gml:axisDirection>
        </gml:CoordinateSystemAxis>
    </gml:usesAxis>
</gml:CartesianCS>
```

Often the definitions can utilize the XLink components to refer to a CS and Datum definitions in an external document.

```
<hasCRS>
    <gml:EngineeringCRS gml:id="sensorCRS">
        <gml:srsName>sensor CRS</gml:srsName>
        <gml:usesCS xlink:href="http://myCrsDefinitions#RFT"/>
        <gml:usesEngineeringDatum xlink:href="http://myCrsDefinitions#inSituPlatformDatum"/>
    </gml:EngineeringCRS>
</hasCRS>
```

10.1.10 Location Models

Within the *locatedUsing* property of a *sensor* or *sample* object, there may be a collection of *LocationModel* definitions that describe a series of locations or transformations through several intermediate CRSes. For instance, a sampleCRS may be related to a sensorCRS which is turn is related to a mountCRS, platformCRS, and ultimately a geospatial CRS. A LocationModel defines the methods and parameters that allow one to relate the sourceCRS to the referenceCRS. Many of these CRSes will be referenced internally within the document, while others might be external:

```
<sourceCRS xlink:href="#mountCRS"/>
```

```
<referenceCRS xlink:href="http://vast.uah.edu/SensorML/platforms/KF4TTB-2.xml#platformCRS"/>
```

Software that is responsible for providing geolocation of complex systems will be responsible for chaining these various models together by connecting the appropriate sourceCRS and referenceCRS combinations.

Within SensorML one can define an object's spatial-temporal properties by either stating its state (i.e. location, orientation, velocity, etc.) or by describing one or more coordinate transformations that describe how one might obtain that state. Mathematically, there is only a fine distinction between describing a state and defining a coordinate transform. One might consider an object's state to be the result of one or more coordinate operations. In terms of description, an object state might utilize terms such as location, orientation, velocity, acceleration, while the description of a coordinate operation might utilize terms such as translation, rotation, scaling, intersection, or projection. Still, both a location and a translation can be described as a vector relative to some CRS.

There are a wide variety of types of location models that can describe either the position (or state) of an object or the transformation of the object's CRS to a reference CRS. Location models are mainly distinguished from one another based on the method to be applied (*usesMethod* property) and the parameters that are required (*usesParameters* property).

The *usesParameters* property within the *LocationModel* takes a *DataGroup* as its value. Several *DataGroup* definitions suitable for location models are provided in *positionData.xsd*. In addition, several general *LocationModelType* classes have been defined in basicLocationModels.xsd, including *GeoPositionModel*, *MountingModel*, *AttitudeModel*, and *RegularGridModel*.

For many in-situ sensors, it may be sufficient to utilize a simple position model within a geospatial CRS:

```
<hasCRS>
    <gml:EngineeringCRS gml:id="sensorCRS">
        <gml:srsName>sensor CRS</gml:srsName>
        <gml:usesCS xlink:href="http://myCrsDefinitions#RFT"/>
        <gml:usesEngineeringDatum xlink:href=" http://myCrsDefinitions#inSituPlatformDatum"/>
    </gml:EngineeringCRS>
</hasCRS>
<locatedUsing>
    <GeoPositionModel id="sensorLocation">
        <sourceCRS xlink:href="#sensorCRS"/>
        <referenceCRS xlink:href="#EPSG:4329"/>
        <usesParameters>
            <GeoPosition id="platformLocation">
                <latitude>
                    <Quantity uom="http://myUnitsDictionary#degrees">34.72450</Quantity>
                </latitude>
                <longitude>
                    <Quantity uom="http://myUnitsDictionary#degrees">-86.94533</Quantity>
                </longitude>
                <altitude>
                    <Quantity uom="http://myUnitsDictionary#meters”>20.1169</Quantity>
                </altitude>
            </ GeoPosition>
        </usesParameters>
    </ GeoPositionModel>
```

```
</locatedUsing>
```

In many cases, it is useful to define a sensor's position relative to the platform:

```
<hasCRS>
    <gml:EngineeringCRS gml:id="sensorCRS">
        <gml:srsName>AMSU-A sensor CRS</gml:srsName>
        <gml:usesCS xlink:href="http://myCrsDefinitions#RFT"/>
        <gml:usesEngineeringDatum xlink:href="http://myCrsDefinitions#inSituPlatformDatum"/>
    </gml:EngineeringCRS>
</hasCRS>
<locatedUsing>
    <MountingModel id="sensorLocation">
        <description>location 2m up from the platform base</description>
        <sourceCRS xlink:href="#sensorCRS"/>
        <referenceCRS xlink:href="http://myPlatform.xml#platformCRS"/>
        <usesParameters>
            <Position>
                <distance crs="http://myPlatform#platformCRS" axisCode="Z">
                    <Quantity uom="http://myUnitsDictionary#meters">2.0</Quantity>
                </distance>
            </Position>
        </usesParameters>
    </MountingModel>
</locatedUsing>
```

Definitions for commonly used location models are provided by the schema basicLocationModels.xsd, which is part of core SensorML standard. These definitions include:

1. *GeoPositionModel* - defines the position (i.e. both location and orientation) of an engineering CRS relative to a geospatial CRS; useful for most platforms, whether static or dynamic.
2. *AttitudeModel* – defines an orientation change in terms of pitch, roll, and yaw
3. *MountingModel* – defines the distance and angular relationship of one engineering CRS to another (e.g. the position of a sensorCRS relative to the platformCRS)
4. *RegularGridModel* – defines a series of positions within a regular grid pattern
5. *IdentityModel* – specifies that two coordinate reference system are coincident; equivalent to a MountingModel where all translation and rotation values are zero

Other special location models will be defined within SensorML extensions. An *OrbitalModel* is one such *LocationModelType* that specifies orbital elements within its parameters and utilizes a Keplerian propagation model as its method for generating location and local orientation along a satellite path.

As stated previously, a component may be located using a collection of *LocationModel* definitions, each responsible for providing the information needed to transform a component's CRS through several intermediate step-wise positions, and ultimately to a meaningful geospatial CRS. The *LocationModelGroup* should be used to encapsulate all transforms needed to transform from the component's CRS to a geospatial CRS:

```
<hasCRS>
    <gml:EngineeringCRS gml:id="sensorCRS">
        <gml:srsName>HRVIR1 Sensor CRS</gml:srsName>
        <gml:usesCS xlink:href="http://myCrsGlossary#RFT-CS"/>
        <gml:usesEngineeringDatum xlink:href="http://myCrsGlossary#mobileSensorDatum"/>
    </gml:EngineeringCRS>
</hasCRS>
<locatedUsing>
    <LocationModelGroup>
        <description>
            <Discussion>Group transforms sensorCRS to an Earth-Centered-Earth-Fixed position
            </Discussion>
        </description>
        <sourceCRS xlink:href="#sensorCRS"/>
        <referenceCRS xlink:href="http://myCrsGlossary#ECEF-CRS"/>
        <member>
            <MountingModel>
                <description>
                    <Discussion>Position of Sensor relative to Platform</Discussion>
                </description>
                <sourceCRS xlink:href “#sensorCRS"/>
                <referenceCRS xlink:href="#platformCRS"/>
                <usesParameters>
                    <Position>
                        <distance crs="#platformCRS" axisCode="X">
                            <Quantity uom="http://myUnitsGlossary#meters">0.53</Quantity>
                        </distance>
                        <distance crs="#platformCRS" axisCode="Y">
                            <Quantity uom="http://myUnitsGlossary#meters">1.32</Quantity>
                        </distance>
                        <distance crs="#platformCRS" axisCode="Z">
                            <Quantity uom="http://myUnitsGlossary#meters">0.003</Quantity>
                        </distance>
                        <angle crs="#platformCRS" axisCode="Z">
                            <Quantity uom="http://myUnitsGlossary#degrees">35.4</Quantity>
                        </angle >
                    </Position>
                </usesParameters>
            </MountingModel>
        </member>
        <member>
            <AttitudeModel>
                <description>
                    <Discussion>Attitude of the platform with respect to a local orbital path CRS
                    </Discussion>
                </description>
                <sourceCRS xlink:href="#platformCRS"/>
                <referenceCRS xlink:href="http://myCrsGlossary#GeocentricOrbital-CRS"/>
                <usesParameters>
                    <Attitude>
                        <time>
                            <IsoDateTime/>
                        </time>
                        <yaw crs="#sensorCRS" axisCode="Z">
                            <Quantity uom="http://myUnitsGlossary#degrees"/>
                        </yaw>
                        <pitch crs="#sensorCRS" axisCode="X">
                            <Quantity uom="http://myUnitsGlossary#degrees"/>
                        </pitch>
                        <roll crs="#sensorCRS" axisCode="Y">
                            <Quantity uom="http://myUnitsGlossary#degrees"/>
                        </roll>
                        <values>
```

```
                        <TupleData>
                            2001-03-01T02-14.32Z,1.224,0.003,0.00001
                            2001-03-01T02-14.33Z,1.224,0.002,0.00004
                        </TupleData>
                    </values>
                </Attitude>
            </usesParameters>
        </AttitudeModel>
    </member>
    <member>
        <OrbitalModel>
            <description>
                <Discussion>Position of orbital CRS using orbit parameters</Discussion>
            </description>
            <sourceCRS xlink:href="http://myCrsGlossary#GeocentricOrbital-CRS"/>
            <referenceCRS xlink:href="http://myCrsGlossary#ECEF-CRS"/>
            <usesParameters>
                <NoradElements>
                    <time>
                        <IsoDateTime>2001-03-01T02-14.32994Z</IsoDateTime>
                    </time>
                    <firstDerivMeanMotion>
                        <Quantity>0.00000571</Quantity>
                    </firstDerivMeanMotion>
                    <secondDerivMeanMotion>
                        <Quantity>0.0</Quantity>
                    </secondDerivMeanMotion>
                    <bstarDrag>
                        <Quantity>13711E-3</Quantity>
                    </bstarDrag>
                    <ephemerisType>
                        <Count>0</Count>
                    </ephemerisType>
                    <elementNumber>
                        <Count>7336</Count>
                    </elementNumber>
                    <inclination>
                        <Quantity uom="#degrees">98.2840</Quantity>
                    </inclination>
                    <rightAscensionOfNode>
                        <Quantity uom="#degrees">122.3556</Quantity>
                    </rightAscensionOfNode>
                    <eccentricity>
                        <Quantity>0.0005764</Quantity>
                    </eccentricity>
                    <argumentOfPerigee>
                        <Quantity uom="#degrees">57.1320</Quantity>
                    </argumentOfPerigee>
                    <meanAnomaly>
                        <Quantity uom="#degrees">303.0432</Quantity>
                    </meanAnomaly>
                    <meanMotion>
                        <Quantity uom="#revsPerDay">14.57081436</Quantity>
                    </meanMotion>
                    <revNumber>
                        <Count>99087</Count>
                    </revNumber>
                </NoradElements>
            </usesParameters>
        </OrbitalModel>
    </member>
</LocationModelGroup>
</locatedUsing>
```

It is also possible to specify component location using two different methods within a single document, for the purpose perhaps of supporting quick location and rigorous location. In such a case, each method would be separated using two *locatedUsing* properties:

```
<locatedUsing>
    <LocationModelGroup>
        <description>
            <Discussion>Method one: quick location method</Discussion>
        </description>
        <sourceCRS xlink:href="#sensorCRS"/>
        <referenceCRS xlink:href="http://myCrsGlossary#ECEF-CRS"/>
        <member>
            ... deleted for brevity …
        </member>
    </LocationModelGroup>
</locatedUsing>
<locatedUsing>
    <LocationModelGroup>
        <description>
            <Discussion>Method two: rigorous location method  </Discussion>
        </description>
        <sourceCRS xlink:href="#sensorCRS"/>
        <referenceCRS xlink:href="http://myCrsGlossary#ECEF-CRS"/>
        <member>
            ... deleted for brevity …
        </member>
    </LocationModelGroup>
</locatedUsing>
```

10.1.11 Component Description

The *describedBy* property of a *Component* provides general descriptive information regarding the component. The ComponentDescription is informative and not intended to contain data needed for processing, geolocating, or discovering the component. It does however provide important information regarding responsible contacts, references, and component history.

```
<describedBy>
    <ComponentDescription>
        <manufacturedBy>
            ... deleted for brevity …
        </manufacturedBy>
        <deployedBy>
            <iso19115:CI_ResponsibleParty id="PaulMeyer">
                <iso19115:contactInfo>
                    <iso19115:phone>
                        <iso19115:voice>+01 (256) 961-7892</iso19115:voice>
                    </iso19115:phone>
                    <iso19115:address>
                    <iso19115:electronicMailAddress>
                        paul.meyer@msfc.nasa.gov
                    </iso19115:electronicMailAddress>
                    </iso19115:address>
                </iso19115:contactInfo>
                <iso19115:role>
                    <iso19115:CI_RoleCode_CodeList>custodian</iso19115:CI_RoleCode_CodeList>
                </iso19115:role>
                <iso19115:individualName>Paul Meyer</iso19115:individualName>
                <iso19115:organisationName>
```

```
                    NASA Marshall Space Flight Center
                </iso19115:organisationName>
            </iso19115:CI_ResponsibleParty>
        </deployedBy>

        <! – utilizes URI link to responsible party defined above-->
        <operatedBy xlink:href="#PaulMeyer"/>

        <description>Standard Anemometer for measuring both wind speed and direction</description>

        <reference>
            <iso19115:CI_OnlineResource>
                <iso19115:linkage>
                    http://www.davisnet.com/weather/products/weather_product.asp?pnum=7911
                </iso19115:linkage>
                <iso19115:description>General description</iso19115:description>
            </iso19115:CI_OnlineResource>
        </reference>
        <reference>
            <iso19115:CI_OnlineResource>
                <iso19115:linkage>
                http://www.davisnet.com/product_documents/weather/spec_sheets/anemometer_std.pdf
                </iso19115:linkage>
                <iso19115:description>Specification Document</iso19115:description>
            </iso19115:CI_OnlineResource>
        </reference>
    </ComponentDescription>
</describedBy>
```

10.1.12 Document Metadata

As discussed in the previous description of the models, a SensorML instance document can be self-describing, providing, for instance, information regarding the document's history, author's, and so forth.

```
<documentedBy>
    <DocumentDescription>
        <history>
            <DocumentHistory>
                <event>
                    <DocumentEvent eventType="DocumentCreated">
                        <timeStamp>
                            <gml:TimeInstant frame="#ISO-8601">
                                <gml:name>Document Creation</gml:name>
                                <gml:timePosition>2004-03-30</gml:timePosition>
                            </gml:TimeInstant>
                        </timeStamp>
                        <responsibleParty>
                            <iso19115:CI_ResponsibleParty>
                                <iso19115:contactInfo>
                                    <iso19115:phone>
                                        <iso19115:voice>+01 (256) 961-7760</iso19115:voice>
                                    </iso19115:phone>
                                    <iso19115:address>
                                        <iso19115:electronicMailAddress>
                                            mike.botts@nsstc.uah.edu
                                        </iso19115:electronicMailAddress>
                                    </iso19115:address>
                                    <iso19115:onlineResource>
                                        <iso19115:linkage>http://vast.uah.edu</iso19115:linkage>
                                    </iso19115:onlineResource>
                                </iso19115:contactInfo>
```

```
                                <iso19115:role>
                                    <iso19115:CI_RoleCode_CodeList>
                                        author
                                    </iso19115:CI_RoleCode_CodeList>
                                </iso19115:role>
                                <iso19115:individualName>Mike Botts</iso19115:individualName>
                                <iso19115:organisationName>
                                    University of Alabama in Huntsville
                                </iso19115:organisationName>
                            </iso19115:CI_ResponsibleParty>
                        </responsibleParty>
                    </DocumentEvent>
                </event>
            </DocumentHistory>
        </history>
    </DocumentDescription>
</documentedBy>
```

10.1.13 Component Interface and Capabilities

The definition for interfaces to the Component is currently abstract, allowing for future definitions of _Interface types. The intent of the component hasInterface property is to describe electrical or web-service interfaces for interacting with the component. Such interfaces can describe the means for tasking a component or for directly obtaining output from the component.

The *CapabilitiesDescription* object is not yet fully defined within this specification. The intent is to allow one to collect in one place, information about the asset that might be useful for discovery. Properties that would be desirable for discovery would probably fall into three categories: supported applications, performance characteristics, and taskability. In many cases, the properties needed to support these categories will have been defined in other sections of the instance document, and will simply need to be referenced using the XLink components. Other properties, such as those dealing with intended applications or design specifications (e.g. intended ground resolution of a remote sensor), might not be explicitly stated elsewhere or can only be derived using complex software. The hasCapabilities properties will provide an single node for easily mining discoverable properties of the asset.

10.1.14 Sensor Measurements and Products

These measurement characteristics of a sensor are described by the *measures* property which takes a *Product* as its value. The *Product*, in essence, provides most of the information required to process, analyze, and geolocate observations from the sensor.

The *Product* has an *id* attribute that is particularly important in that it provides the means by which observations can be linked to the appropriate description of the measurement characteristics. In additional, the *Product* has one or more *Identifier* objects to provide appropriate names for use in product list and so forth.

```
<measures>
    <Product id="windSpeed">
        <identifiedAs>
            <Identifier type="shortName">Wind Speed</Identifier>
        </identifiedAs>
        <!—continued below -->
```

The *observable* property defines the type of physical phenomenon being measures and thus takes a *Phenomenon* as its value. The string value of the *Phenomenon* object provides a keyname of the observable property measured by the sensor. The codeSpace attribute will typically link to an authoritative dictionary that defines this term.

```
<observable>
    <Phenomenon codeSpace="myPhenomenonDictionary.xml">
        wind speed
    </Phenomenon>
</observable>
<!—continued below -->
```

Since a *Product* definition can exist separate from the sensor or device that produced it, a *Product* has an optional *providedBy* property that can link to or include a description of the *Component* that generated it. Being optional, this may typically be omitted if the Product is being described within the sensor description.

```
<providedBy xlink:href="http://mySensor.xml">
<!—continued below -->
```

10.1.15 Geolocation of Observation Products

A *Product* description includes *hasCRS* and *locatedUsing* as optional properties, similar to various *_Component* objects in SensorML. As with *Components*, these properties take an *EngineeringCRS* and *LocationModel* as their respective values. The CRS for a *Product* might define, for example, the coordinate reference system for a point sample, image, or volume. The *locatedUsing* property essentially defines how to locate this data CRS to some other meaningful CRS (e.g. geospatial, sensor, or platform CRS).

For an in-situ sensor, its *Product*(s) may simply be locatedUsing the sensors location,

```
<locatedUsing xlink:href="#sensorLocation"/>
```

or by the location of its *Sample*, if the sample is described within the document.

For remote sensors, the geolocation of a *Product* is more complicated. Remote sensor products typically utilize at least one *SensorModel* description for its *locatedUsing* property. A *SensorModel* generally provides the methodology and parameters needed to relate an individual sample (or pixel) CRS to some other appropriate CRS (typically the sensor CRS or directly to a geospatial CRS). SensorModel used for remote sensors are defined in OGC document 04-068 and will not be discussed in detail here.

10.1.16 Samples

Sensors base their measurements on a sample or a collection of samples of a real-world phenomenon. A sample could be as diverse as a rock sample, a vial of water from the ocean, or a conic volume of atmosphere sampled by a remote sensor. Within SensorML, a *Sample* is derived from *Product*. In addition to other properties inherited from *Product*, a *Sample* can provide information required to determine the geometry, dimensions, and

location of the sample.

Both a *_SensorModel* and *ResponseModel* take one or more *Sample* descriptions as their input. Typically, the same *Sample* definition would be used for each model, with one of the properties linking to the previous definition. In the case of the *SensorModel*, the *Sample* may provide additional information needed for geolocating sensor observations (e.g. an effective field of view, EFOV, for a pixel). In the case of a ResponseModel, the *Sample* might provide information necessary to interpret the returned measurement (e.g. the total volume of an air sample).

10.1.17 Process Models and Sensor Response Model

The values of a *Product* measures by a sensor system (e.g. the instrument plus processors), are dependent on the characteristics of the sensing device and on the process chain that has acted on the observations. A *Product* thus has a *derivedBy* property that defines these processes. The *derivedBy* property takes as its value, one or more *ProcessModel* instances. The *ResponseModel* is a special case *ProcessModel* that describes the process of the sensor taking a measurement based on a *Sample*.

The sensitivity of the sensor to an observable phenomenon, as well as other environmental phenomena, is defined within the sensor's *ResponseModel*. The *ResponseModel* can provide parameters needed to define the sensor's sensitivity, performance characteristics, survivability, and calibration properties. These are parameters that are needed to adequately process and interpret the observations from the sensor.

Further processing of the sensor's observations might involve, perhaps, such operations as converting radiance or voltage values to geophysical values, aggregating observations, or determining the presence of a feature based on the observations. These processes, including all processes often associated with image processing, for example, could be describe using derived classes of the *ProcessModelType*.

Like the *LocationModel*, the *ProcessModel* and *ResponseModel* include the *usesMethod* and *usesParameters* properties. The usesMethod property, if included will typically link to a description of the parameter definitions and the assumptions and suggested algorithms used to process the data. Additional discussion regarding specific ProcessModel definitions will be included in future documentation.

With regard to the ResponseModel, the core SensorML specification includes general model properties such as *dynamicRange*, *survivableRange*, *relativeAccuracy*, *operationalRange*, *samplingPeriod*, and others as defined within the ResponseModel.xsd schema. Their use is illustrated using an anemometer (wind speed) sensor:

```
<measures>
    <Product id="windSpeed">
        <identifiedAs>
            <Identifier type="shortName">Wind Speed</Identifier>
        </identifiedAs>
        <observable xlink:href="http://myObservableDictionary.xml#wind speed"/>
        <locatedUsing xlink:href="#sensorLocation"/>
```

```
            <derivedFrom>
                <ResponseModel id="windSpeedResponse">
                    <description>
                        <Discussion id="7911_cableLength" topic="cable length">
                        On Monitor and Wizard stations, cable lengths longer than 140' (42 m) between
                        sensors and console may artificially limit wind speed readings. That is, beyond that
                        length, maximum recordable wind speed decreases as cable length increases. For
                        example, with a length of 140' (42 m), the maximum recordable speed exceeds
                        175 mph. At 240' (72 m), however, the maximum recordable speed drops to less than
                        140 mph. Below that upper limit, however, the anemometer's accuracy is not affected.
                        On GroWeather, Energy and Health EnviroMonitor stations, the maximum recordable
                        wind speed is 175 mph regardless of cable length.
                        </Discussion>
                    </description>
                    <description>
                        <Discussion id="7911_survivability" topic="survivability">
                            A model 7911 Anemometer reported wind speeds of 78 m/s before the tower
                            collapsed during Hurricane Andrew.
                        </Discussion>
                    </description>
                    <usesParameters>
                        <BasicResponse>
                            <dynamicRange>
                                <TypedQuantityRange uom="http://myUnitsDictionary.xml#meterPerSecond"
                                    type="http://myObservableDictionary.xml#windSpeed"> 0 78
                                </TypedQuantityRange>
                            </dynamicRange>

                            <survivableRange>
                                <TypedQuantityRange uom="http://myUnitsDictionary.xml#metersPerSecond"
                                    type="http://myObservableDictionary.xml#windSpeed">  0 78
                                </TypedQuantityRange>
                            </survivableRange>

                            <relativeAccuracy>
                                <TypedQuantityRange uom="http://myUnitsDictionary.xml#percentage"
                                    type="http://myObservableDictionary.xml#windSpeed"> -5 +5
                                </TypedQuantityRange>
                            </relativeAccuracy>

                            <resolution>
                                <TypedQuantity uom="http://myUnitsDictionary.xml#metersPerSecond"
                                    type="http://myObservableDictionary#windSpeed"> 0.1
                                </TypedQuantity>
                            </resolution>

                            <samplePeriod>
                                <Quantity uom="http://myUnitsDictionary.xml#seconds">2.25</Quantity>
                            </samplePeriod>

                        </BasicResponse>
                    </usesParameters>
                </ResponseModel>
            </derivedFrom>
        </Product>
    </measures>
```

The limited number of general characteristics available in the responseModel.xsd may be sufficient to support a large number of simple sensors. Other general characteristics may be added as part of SensorML core. Also, it is assumed that there will be a need to define

response characteristics definitions that are specific to particular sensor types or user communities. Examples might include specialized response models for radiation, chemical concentration, or biological populations. These will be included as part of SensorML extensions.

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, and processing and geolocating observation data based on SensorML descriptions. 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 sensor-related terms, coordinate reference systems, and coordinate operations.
2. Finish the initial development and publishing of SensorML extension schema for community-specific types of SensorModel, LocationModel, ProcessModel, ResponseModel, and DataProvider objects (see OGC document 04-068 for current definitions).
3. Investigate avenues and implement solutions for increased synergy between SensorML and GML. This will probably involve some modifications to both SensorML and GML.
4. Investigate avenues and implement solutions for increased synergy between SensorML and the OGC Observations and Measurements (O&M) schema. This will probably involve some modifications to both SensorML and O&M.
5. Provide better base definitions for interfaces (including web services and electrical interfaces) and for describing sensor/platform capabilities. Implement specific interface models for developing standards such as IEEE P1451 and RFID.
6. Assist in the development, testing, and demonstration of SensorML-aware software that is capable of sensor discovery, and geolocation and processing of observation.
7. Investigate the capability of utilizing Product and ProcessModel base definitions for general use in processing chains.

Annex A: XML Schemas for SensorML (normative)

A.1 base.xsd.

The base.xsd schema provides the basic definitions and abstract elements used by several SensorML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>base class definitions for core SensorML</xs:documentation>
    </xs:annotation>
    <!-- ========================
        includes and imports
    ========================== -->
    <xs:import namespace="http://www.isotc211.org/iso19115/"
        schemaLocation="../iso19115/0.1.21/iso19115.xsd"/>
    <xs:include schemaLocation="./parameters.xsd"/>
    <xs:include schemaLocation="./events.xsd"/>
    <!-- ========================
        element values
    ========================== -->
    <xs:element name="Constraints" type="sml:ConstraintsType"/>
    <xs:element name="Identifier">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:string">
                    <xs:attribute name="type" use="optional">
                        <xs:simpleType>
                            <xs:restriction base="xs:string">
                                <xs:enumeration value="longName"/>
                                <xs:enumeration value="shortName"/>
                                <xs:enumeration value="serialNumber"/>
                                <xs:enumeration value="modelNumber"/>
                                <xs:enumeration value="missionNumber"/>
                                <xs:enumeration value="partNumber"/>
                            </xs:restriction>
                        </xs:simpleType>
                    </xs:attribute>
                    <xs:attribute name="codeSpace" type="xs:anyURI" use="optional"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="Classifier">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="xs:string">
                    <xs:attribute name="type" use="required">
                        <xs:simpleType>
                            <xs:restriction base="xs:string"/>
```

```
                    </xs:simpleType>
                </xs:attribute>
                <xs:attribute name="codeSpace" type="xs:anyURI" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<xs:element name="Discussion">
    <xs:complexType>
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attributeGroup ref="sml:IdentificationAttributes"/>
                <xs:attribute name="topic" type="xs:anyURI" use="optional"/>
                <xs:attribute name="codeSpace" type="xs:anyURI" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:element>
<!-- ============================
    global concrete properties
============================== -->
<xs:element name="description">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="sml:Discussion" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="sml:AssociationAttributes"/>
    </xs:complexType>
</xs:element>
<xs:element name="identifiedAs">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="sml:Identifier"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="classifiedAs">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="sml:Classifier"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
<xs:element name="reference">
    <xs:complexType>
        <xs:choice minOccurs="0">
            <xs:element ref="iso19115:CI_Citation"/>
            <xs:element ref="iso19115:CI_OnlineResource"/>
        </xs:choice>
        <xs:attributeGroup ref="sml:AssociationAttributes"/>
    </xs:complexType>
</xs:element>
<xs:element name="documentedBy">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="DocumentDescription">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element ref="sml:reference" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="history" minOccurs="0">
                            <xs:complexType>
                                <xs:sequence>
```

```
                                    <xs:element name="DocumentHistory">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element name="event" maxOccurs="unbounded">
                                                    <xs:complexType>
                                                        <xs:sequence>
                                                            <xs:element ref="sml:DocumentEvent"
                                                                minOccurs="0"/>
                                                        </xs:sequence>
                                                    </xs:complexType>
                                                </xs:element>
                                            </xs:sequence>
                                        </xs:complexType>
                                    </xs:element>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element name="documentConstrainedBy">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sml:Constraints"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <!-- ============================
                Complex Types
            ============================ -->
            <xs:complexType name="ConstraintsType">
                <xs:sequence>
                    <xs:element name="validTime" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:choice>
                                <xs:group ref="sml:TimeParameter"/>
                                <xs:sequence>
                                    <xs:element name="TimeExtent">
                                        <xs:complexType>
                                            <xs:sequence>
                                                <xs:element ref="sml:description" minOccurs="0"/>
                                                <xs:element name="startTime">
                                                    <xs:complexType>
                                                        <xs:sequence>
                                                            <xs:group ref="sml:TimeInstantParameter"/>
                                                        </xs:sequence>
                                                    </xs:complexType>
                                                </xs:element>
                                                <xs:element name="stopTime">
                                                    <xs:complexType>
                                                        <xs:sequence>
                                                            <xs:group ref="sml:TimeInstantParameter"/>
                                                        </xs:sequence>
                                                    </xs:complexType>
                                                </xs:element>
                                            </xs:sequence>
                                            <xs:attribute name="id" type="xs:ID" use="optional"/>
                                        </xs:complexType>
                                    </xs:element>
                                </xs:sequence>
```

```
                </xs:choice>
            </xs:complexType>
        </xs:element>
        <xs:element name="securityLevel" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="iso19115:MD_SecurityConstraints"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
        <xs:element name="legalUse" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="iso19115:MD_LegalConstraints"/>
                </xs:sequence>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
</xs:complexType>
<!-- ==========================
    Simple Types
    ========================== -->
<xs:simpleType name="IdentifierSimpleType">
    <xs:restriction base="xs:string"/>
</xs:simpleType>
</xs:schema>
```

A.2 component.xsd

The component.xsd schema provides the definition for the base Component and Asset classes used from which Sensor and Platform are derived. It also provides property definition used by components. It can be used to derive other component types in the future and as the base schema for any instance that describes a general Component other than a Sensor or Platform.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:sml="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>base class definitions for core SensorML</xs:documentation>
    </xs:annotation>
    <!-- ==========================
        includes and imports
    ========================== -->
    <xs:include schemaLocation="./base.xsd"/>
    <xs:include schemaLocation="./positionData.xsd"/>
    <xs:include schemaLocation="./locationModel.xsd"/>
    <xs:include schemaLocation="./specificationProperties.xsd"/>
    <!-- ==========================
        global substitution group elements
    ========================== -->
    <xs:element name="_Component" type="sml:ComponentType" abstract="true">
```

```
        <xs:annotation>
            <xs:documentation>Serves as head of a substitution group for all components
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="_Asset" abstract="true" substitutionGroup="sml:_Component">
        <xs:annotation>
            <xs:documentation>Serves as head of a substitution group for all assets</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:AssetType"/>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="_Interface" abstract="true">
        <xs:annotation>
            <xs:documentation>SubstitutionGroup for all Interfaces</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ============================
        Component Elements
    ============================== -->
    <xs:element name="Component" type="sml:ComponentType" substitutionGroup="sml:_Component"/>
    <xs:element name="ComponentArray" type="sml:ComponentArrayType"
substitutionGroup="sml:_Component"/>
    <xs:element name="ComponentGroup" type="sml:ComponentGroupType"
substitutionGroup="sml:_Component"/>
    <!-- ============================
        Asset Elements
    ============================== -->
    <xs:element name="Asset" type="sml:AssetType" substitutionGroup="sml:_Asset"/>
    <xs:element name="AssetArray" type="sml:AssetArrayType" substitutionGroup="sml:_Asset"/>
    <xs:element name="AssetGroup" type="sml:AssetGroupType" substitutionGroup="sml:_Asset"/>
    <!-- ============================
        Complex Types
    ============================== -->
    <xs:complexType name="ComponentType">
        <xs:sequence>
            <xs:element ref="sml:identifiedAs" maxOccurs="unbounded"/>
            <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:documentConstrainedBy" minOccurs="0"/>
            <xs:element ref="sml:hasCapabilities" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:hasCRS" minOccurs="0"/>
            <xs:element ref="sml:locatedUsing" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:hasPhysicalProperties" minOccurs="0"/>
            <xs:element ref="sml:describedBy" minOccurs="0"/>
            <xs:element ref="sml:documentedBy" minOccurs="0"/>
            <xs:element ref="sml:hasInterface" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:hasComponent" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attributeGroup ref="sml:IdentificationAttributes"/>
        <xs:attribute name="version" type="xs:string" use="optional" fixed="0.9b"/>
    </xs:complexType>
    <xs:complexType name="ComponentArrayType">
        <xs:complexContent>
            <xs:extension base="sml:ComponentType">
                <xs:sequence>
                    <xs:element name="member">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_Component" minOccurs="0"/>
                            </xs:sequence>
```

```
                        <xs:attributeGroup ref="sml:AssociationAttributes"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="ComponentGroupType">
    <xs:complexContent>
        <xs:extension base="sml:ComponentType">
            <xs:sequence>
                <xs:element name="member" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:_Component" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="sml:AssociationAttributes"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="AssetType">
    <xs:complexContent>
        <xs:extension base="sml:ComponentType">
            <xs:sequence>
                <xs:element ref="sml:attachedTo" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="AssetArrayType">
    <xs:complexContent>
        <xs:extension base="sml:AssetArrayType"/>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="AssetGroupType">
    <xs:complexContent>
        <xs:extension base="sml:ComponentGroupType"/>
    </xs:complexContent>
</xs:complexType>
<!--============================
    global properties
============================== -->
<xs:element name="describedBy">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="ComponentDescription" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element name="manufacturedBy" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="iso19115:CI_ResponsibleParty" minOccurs="0"/>
                                </xs:sequence>
                                <xs:attributeGroup ref="sml:AssociationAttributes"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="deployedBy" minOccurs="0" maxOccurs="unbounded">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="iso19115:CI_ResponsibleParty" minOccurs="0"/>
```

```
                        </xs:sequence>
                        <xs:attributeGroup ref="sml:AssociationAttributes"/>
                    </xs:complexType>
                </xs:element>
                <xs:element name="operatedBy" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="iso19115:CI_ResponsibleParty" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="sml:AssociationAttributes"/>
                    </xs:complexType>
                </xs:element>
                <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="sml:reference" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element name="history" minOccurs="0">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element name="ComponentHistory" minOccurs="0">
                                <xs:complexType>
                                    <xs:sequence>
                                        <xs:element name="event" maxOccurs="unbounded">
                                            <xs:complexType>
                                                <xs:sequence>
                                                    <xs:element ref="sml:ComponentEvent"
                                                        minOccurs="0"/>
                                                </xs:sequence>
                                                <xs:attributeGroup
                                                    ref="sml:AssociationAttributes"/>
                                            </xs:complexType>
                                        </xs:element>
                                    </xs:sequence>
                                    <xs:attributeGroup ref="sml:IdentificationAttributes"/>
                                </xs:complexType>
                            </xs:element>
                        </xs:sequence>
                        <xs:attributeGroup ref="sml:AssociationAttributes"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attributeGroup ref="sml:IdentificationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:attributeGroup ref="sml:AssociationAttributes"/>
</xs:complexType>
</xs:element>
<xs:element name="hasCRS">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="gml:EngineeringCRS" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="sml:AssociationAttributes"/>
    </xs:complexType>
</xs:element>
<xs:element name="hasPosition">
    <xs:complexType>
        <xs:sequence>
            <xs:element ref="sml:_PositionData" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="sml:AssociationAttributes"/>
    </xs:complexType>
</xs:element>
<xs:element name="locatedUsing">
```

```
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_LocationModel" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="hasPhysicalProperties">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="PhysicalProperties" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>will take physical properties such as dimension, geometry,
                            weight, etc.</xs:documentation>
                    </xs:annotation>
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:_physicalProperty" maxOccurs="unbounded"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="hasComponent">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_Component" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="hasInterface">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_Interface" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="hasCapabilities">
        <xs:complexType>
            <xs:choice minOccurs="0">
                <xs:element name="Applications">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:_capability"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="DesignSpecifications">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:_capability"/>
                        </xs:sequence>
                    </xs:complexType>
                </xs:element>
                <xs:element name="TaskableOperations">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:_capability"/>
                        </xs:sequence>
```

```
                    </xs:complexType>
                </xs:element>
            </xs:choice>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="attachedTo">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_Component" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

A.3 sensor.xsd.

Sensor.xsd provides the basic definitions for Sensors and Sensor Collections.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>sensor object for core SensorML</xs:documentation>
    </xs:annotation>
    <!-- ========================
        includes and imports
    ========================== -->
    <xs:include schemaLocation="./component.xsd"/>
    <xs:include schemaLocation="./sampleAndResponse.xsd"/>
    <!-- ========================
        global substitution group elements
    ========================== -->
    <xs:element name="_Sensor" type="sml:SensorType" abstract="true"
            substitutionGroup="sml:_Component">
        <xs:annotation>
            <xs:documentation>SubstitutionGroup for all Sensors</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ========================
        global concrete values
    ========================== -->
    <xs:element name="Sensor" type="sml:SensorType" substitutionGroup="sml:_Sensor">
        <xs:annotation>
            <xs:documentation>A component that provides measurements</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="SensorArray" type="sml:SensorArrayType" substitutionGroup="sml:_Sensor">
        <xs:annotation>
            <xs:documentation>Collection of identical sensors that together provide a common
                measurement</xs:documentation>
        </xs:annotation>
    </xs:element>
```

```
    <xs:element name="SensorGroup" type="sml:SensorGroupType" substitutionGroup="sml:_Sensor">
        <xs:annotation>
            <xs:documentation>Collection of disparate sensors that together provide a common
                measurement</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ============================
        Complex Types
    ============================== -->
    <xs:complexType name="SensorType">
        <xs:complexContent>
            <xs:extension base="sml:AssetType">
                <xs:sequence>
                    <xs:element name="measures" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>One measure for each different kind of measurement
                            </xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_Product" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="sml:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SensorGroupType">
        <xs:complexContent>
            <xs:extension base="sml:SensorType">
                <xs:sequence>
                    <xs:element name="member" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_Sensor" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SensorArrayType">
        <xs:complexContent>
            <xs:extension base="sml:SensorType">
                <xs:sequence>
                    <xs:element name="member">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_Sensor" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="arraySize" type="xs:integer" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

A.4 platform.xsd

Provides schema for defining platforms as they relate to sensors.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:sml="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>platform definition for core SensorML</xs:documentation>
    </xs:annotation>
    <!-- ========================
        includes and imports
    ========================== -->
    <xs:include schemaLocation="./component.xsd"/>
    <!-- ========================
        global abstract element
    ========================== -->
    <xs:element name="_Platform" type="sml:PlatformType" abstract="true"
            substitutionGroup="sml:_Component">
        <xs:annotation>
            <xs:documentation>SubstitutionGroup for all Platforms</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ========================
        global concrete values
    ========================== -->
    <xs:element name="Platform" type="sml:PlatformType" substitutionGroup="sml:_Platform"/>
    <xs:element name="PlatformArray" type="sml:PlatformArrayType" substitutionGroup="sml:_Platform"/>
    <xs:element name="PlatformGroup" type="sml:PlatformGroupType"
        substitutionGroup="sml:_Platform"/>
    <!-- ========================
        Complex Types
    ========================== -->
    <xs:complexType name="PlatformType">
        <xs:complexContent>
            <xs:extension base="sml:AssetType">
                <xs:sequence>
                    <xs:element name="carries" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_Asset" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="PlatformGroupType">
        <xs:complexContent>
            <xs:extension base="sml:PlatformType">
                <xs:sequence>
                    <xs:element name="member" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
```

```
                        <xs:element ref="sml:_Platform" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:extension>
</xs:complexContent>
</xs:complexType>
<xs:complexType name="PlatformArrayType">
    <xs:complexContent>
        <xs:extension base="sml:PlatformType">
            <xs:sequence>
                <xs:element name="member">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:_Platform" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="arraySize" type="xs:integer" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</xs:schema>
```

A.5 sampler.xsd.

Provides definitions for assets that primarily of collect samples.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>sensor object for core SensorML</xs:documentation>
    </xs:annotation>
    <!-- ========================
        includes and imports
    ========================== -->
    <xs:include schemaLocation="./component.xsd"/>
    <xs:include schemaLocation="./productAndProcess.xsd"/>
    <!-- ========================
        global substitution group elements
    ========================== -->
    <xs:element name="_Sampler" type="sml:SamplerType" abstract="true"
            substitutionGroup="sml:_Component">
        <xs:annotation>
            <xs:documentation>SubstitutionGroup for all Sampler</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ========================
```

```
    global concrete values
============================== -->
<xs:element name="Sampler" type="sml:SamplerType" substitutionGroup="sml:_Sampler">
    <xs:annotation>
        <xs:documentation>A component that collects samples but does not make measures on them
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="SamplerArray" type="sml:SamplerArrayType" substitutionGroup="sml:_Sampler">
    <xs:annotation>
        <xs:documentation>Collection of identical samplers that together provide a sample collection
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="SamplerGroup" type="sml:SamplerGroupType" substitutionGroup="sml:_Sampler">
    <xs:annotation>
        <xs:documentation>Collection of disparate samplers that together provide a sample collection
        </xs:documentation>
    </xs:annotation>
</xs:element>
<!-- ============================
    Complex Types
============================== -->
<xs:complexType name="SamplerType">
    <xs:complexContent>
        <xs:extension base="sml:AssetType">
            <xs:sequence>
                <xs:element name="collects" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:_Product" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="sml:AssociationAttributes"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="SamplerGroupType">
    <xs:complexContent>
        <xs:extension base="sml:SamplerType">
            <xs:sequence>
                <xs:element name="member" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:_Sampler" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<xs:complexType name="SamplerArrayType">
    <xs:complexContent>
        <xs:extension base="sml:SamplerType">
            <xs:sequence>
                <xs:element name="member">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:_Sampler" minOccurs="0"/>
```

```
                        </xs:sequence>
                        <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                    </xs:complexType>
                </xs:element>
            </xs:sequence>
            <xs:attribute name="arraySize" type="xs:integer" use="required"/>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</xs:schema>
```

A.6 transmitter.xsd.

Provides definitions for assets that primarily of transmit radiation or provides other stimuli in conjunction with sensors.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>Transmitter object for core SensorML</xs:documentation>
    </xs:annotation>
    <!-- ============================
        includes and imports
    ============================== -->
    <xs:include schemaLocation="./component.xsd"/>
    <xs:include schemaLocation="./productAndProcess.xsd"/>
    <!-- ============================
        global substitution group elements
    ============================== -->
    <xs:element name="_Transmitter" type="sml:TransmitterType" abstract="true"
            substitutionGroup="sml:_Component">
        <xs:annotation>
            <xs:documentation>SubstitutionGroup for all transmitters</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ============================
        global concrete values
    ============================== -->
    <xs:element name="Transmitter" type="sml:TransmitterType" substitutionGroup="sml:_Transmitter">
        <xs:annotation>
            <xs:documentation>A component that provides a signal</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="TransmitterArray" type="sml:TransmitterArrayType"
            substitutionGroup="sml:_Transmitter">
        <xs:annotation>
            <xs:documentation>Collection of identical transmitters that together provide a common signal
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="TransmitterGroup" type="sml:TransmitterGroupType"
            substitutionGroup="sml:_Transmitter">
        <xs:annotation>
```

```
            <xs:documentation>Collection of disparate transmitters that together provide a common signal
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ============================
        Complex Types
    ============================== -->
    <xs:complexType name="TransmitterType">
        <xs:complexContent>
            <xs:extension base="sml:AssetType">
                <xs:sequence>
                    <xs:element name="transmits" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_Product" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="sml:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TransmitterGroupType">
        <xs:complexContent>
            <xs:extension base="sml:TransmitterType">
                <xs:sequence>
                    <xs:element name="member" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_Transmitter" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="TransmitterArrayType">
        <xs:complexContent>
            <xs:extension base="sml:TransmitterType">
                <xs:sequence>
                    <xs:element name="member">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_Transmitter" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
                <xs:attribute name="arraySize" type="xs:integer" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

A.7 events.xsd

Provides definitons for events and history used in component descriptions.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:om="http://www.opengis.net/om"
xmlns:gml="http://www.opengis.net/gml" xmlns:sml="http://www.opengis.net/sensorML"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>events definitions for core SensorML</xs:documentation>
    </xs:annotation>
    <!-- ========================
        includes and imports
    ========================== -->
    <xs:include schemaLocation="./base.xsd"/>
    <!-- ========================
        global abstract elements
    ========================== -->
    <xs:element name="_Event" type="sml:EventType" abstract="true">
        <xs:annotation>
            <xs:documentation>SubstitutionGroup for all events</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ========================
        global concrete values
    ========================== -->
    <xs:element name="DocumentEvent" type="sml:EventType"/>
    <xs:element name="ComponentEvent" type="sml:EventType"/>
    <!-- ========================
        Complex Types
    ========================== -->
    <xs:complexType name="EventType">
        <xs:sequence>
            <xs:element name="time" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:group ref="sml:TimeParameter"/>
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element ref="sml:identifiedAs" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="responsibleParty" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence minOccurs="0">
                        <xs:element ref="iso19115:CI_ResponsibleParty"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                </xs:complexType>
            </xs:element>
            <xs:element ref="sml:description" minOccurs="0"/>
            <xs:element ref="sml:reference" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="optional"/>
        <xs:attribute name="eventType" type="xs:anyURI" use="optional"/>
    </xs:complexType>
</xs:schema>
```

A.8 parameters.xsd

Provides simple parameters and data groups used throughout SensorML, buts particularly within models.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:sml="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>parameters for core SensorML</xs:documentation>
    </xs:annotation>
    <!-- ==========================
        includes and imports
    ============================ -->
    <xs:import namespace="http://www.opengis.net/gml" schemaLocation="./gmlStub.xsd"/>
    <xs:include schemaLocation="./dataProvider.xsd"/>
    <!-- ==========================
        parameter groups
    ============================ -->
    <xs:group name="SimpleParameter">
        <xs:choice>
            <xs:element ref="sml:Boolean"/>
            <xs:element ref="sml:Category"/>
            <xs:element ref="sml:Coefficient"/>
            <xs:element ref="sml:Count"/>
            <xs:element ref="sml:CountList"/>
            <xs:element ref="sml:CountRange"/>
            <xs:element ref="sml:Quantity"/>
            <xs:element ref="sml:QuantityList"/>
            <xs:element ref="sml:QuantityRange"/>
        </xs:choice>
    </xs:group>
    <xs:group name="TypedParameter">
        <xs:choice>
            <xs:element ref="sml:TypedBoolean"/>
            <xs:element ref="sml:TypedCount"/>
            <xs:element ref="sml:TypedCountList"/>
            <xs:element ref="sml:TypedCountRange"/>
            <xs:element ref="sml:TypedCategory"/>
            <xs:element ref="sml:TypedQuantity"/>
            <xs:element ref="sml:TypedQuantityList"/>
            <xs:element ref="sml:TypedQuantityRange"/>
        </xs:choice>
    </xs:group>
    <xs:group name="TimeParameter">
        <xs:choice>
            <xs:group ref="sml:TimeInstantParameter"/>
            <xs:group ref="sml:TimeRangeParameter"/>
        </xs:choice>
    </xs:group>
    <xs:group name="TimeRangeParameter">
        <xs:choice>
            <xs:element ref="sml:DecimalTimeRange"/>
```

```
            <xs:element ref="sml:IsoDateTimeRange"/>
        </xs:choice>
    </xs:group>
    <xs:group name="TimeInstantParameter">
        <xs:choice>
            <xs:element ref="sml:IsoDateTime"/>
            <xs:element ref="sml:DecimalTime"/>
        </xs:choice>
    </xs:group>
    <!-- ==========================
        attribute groups
    ============================ -->
    <xs:attributeGroup name="AssociationAttributes">
        <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="IdentificationAttributes">
        <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="ParameterAttributes">
        <xs:annotation>
            <xs:documentation>All parameters have an id, and the fixed and controllable
                flags</xs:documentation>
        </xs:annotation>
        <xs:attributeGroup ref="sml:IdentificationAttributes"/>
        <xs:attribute name="fixed" type="xs:boolean" use="optional" default="true"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="DefinitionAttributes">
        <xs:annotation>
            <xs:documentation>Used for typed quantities needing an axis</xs:documentation>
        </xs:annotation>
        <xs:attribute name="type" type="xs:anyURI" use="required"/>
        <xs:attribute name="codespace" type="xs:anyURI" use="optional"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="QuantityAttributes">
        <xs:attribute name="uom" type="xs:anyURI" use="optional"/>
        <xs:attribute name="min" type="xs:double" use="optional"/>
        <xs:attribute name="max" type="xs:double" use="optional"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="CountAttributes">
        <xs:attribute name="min" type="xs:integer" use="optional"/>
        <xs:attribute name="max" type="xs:integer" use="optional"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="AxisAttributes">
        <xs:attribute name="crs" type="xs:anyURI" use="required"/>
        <xs:attribute name="axisCode" use="required">
            <xs:simpleType>
                <xs:restriction base="xs:string">
                    <xs:enumeration value="X"/>
                    <xs:enumeration value="Y"/>
                    <xs:enumeration value="Z"/>
                </xs:restriction>
            </xs:simpleType>
        </xs:attribute>
        <xs:attribute name="direction" type="gml:SignType" use="optional" default="+"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="TimeOriginAttributes">
        <xs:attribute name="origin" type="sml:IsoDateTimeSimpleType" use="optional"/>
    </xs:attributeGroup>
    <xs:attributeGroup name="ListAttributes">
        <xs:attribute name="length" type="xs:positiveInteger" use="required"/>
    </xs:attributeGroup>
    <!-- ==========================
        simple parameters
```

```
    ============================== -->
    <xs:element name="Boolean" type="sml:BooleanParameterType">
        <xs:annotation>
            <xs:documentation>True or False, 0 or 1</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="TypedBoolean" type="sml:TypedBooleanType">
        <xs:annotation>
            <xs:documentation>Boolean with type and codespace pointing to a dictionnary
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Count" type="sml:CountParameterType">
        <xs:annotation>
            <xs:documentation>Integer number used for a counting value with optional bounds
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="CountList" type="sml:CountListType">
        <xs:annotation>
            <xs:documentation>Ordered list of count values separated by spaces</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="CountRange" type="sml:CountRangeType">
        <xs:annotation>
            <xs:documentation>Pair of count values specifying a range (bounds apply to both of them)
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="TypedCount" type="sml:TypedCountType">
        <xs:annotation>
            <xs:documentation>Count with type and codespace pointing to a dictionnary
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="TypedCountList" type="sml:TypedCountListType">
        <xs:annotation>
            <xs:documentation>CountList with type and codespace pointing to a dictionnary
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="TypedCountRange" type="sml:TypedCountRangeType">
        <xs:annotation>
            <xs:documentation>CountRange with type and codespace pointing to a dictionnary
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="Quantity" type="sml:QuantityParameterType">
        <xs:annotation>
            <xs:documentation>Decimal number with optional unit and bounds</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="QuantityList" type="sml:QuantityListType">
        <xs:annotation>
            <xs:documentation>Ordered list of decimal values separated by spaces</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="QuantityRange" type="sml:QuantityRangeType">
        <xs:annotation>
            <xs:documentation>Pair of decimal values specifying a range (bounds apply to both of them)
            </xs:documentation>
        </xs:annotation>
    </xs:element>
```

```
<xs:element name="TypedQuantity" type="sml:TypedQuantityType">
    <xs:annotation>
        <xs:documentation>Quantity with type and codespace pointing to a dictionnary
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="TypedQuantityList" type="sml:TypedQuantityListType">
    <xs:annotation>
        <xs:documentation>QuantityList with type and codespace pointing to a dictionnary
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="TypedQuantityRange" type="sml:TypedQuantityRangeType">
    <xs:annotation>
        <xs:documentation>QuantityRange with type and codespace pointing to a dictionnary
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="String" type="sml:StringParameterType">
    <xs:annotation>
        <xs:documentation>Unrestricted string (not allowed in tuples !!)</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="Category" type="sml:StringParameterType">
    <xs:annotation>
        <xs:documentation>String with only letters and numbers identifying a category
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="TypedCategory" type="sml:TypedCategoryType">
    <xs:annotation>
        <xs:documentation>Catedory with type and codespace pointing to a dictionnary
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="Coefficient" type="sml:CoefficientParameterType">
    <xs:annotation>
        <xs:documentation>Decimal value with an index number (e.g. for polynomial coeffs)
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="IsoDateTime" type="sml:IsoDateTimeType">
    <xs:annotation>
        <xs:documentation>Time instance given in ISO 8601 format (e.g. 2004-04-18T12:03:04.6Z) or
            currentTime, notValid</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="IsoDateTimeList" type="sml:IsoDateTimeListType">
    <xs:annotation>
        <xs:documentation>Ordered list of IsoDateTime values separated by spaces
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="IsoDateTimeRange" type="sml:IsoDateTimeRangeType">
    <xs:annotation>
        <xs:documentation>Pair of IsoDateTime values specifying a range</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="DecimalTime" type="sml:DecimalTimeParameterType">
    <xs:annotation>
        <xs:documentation>Time (with unit: seconds, minutes, days, or years) past or before an Epoch
            (origin)</xs:documentation>
    </xs:annotation>
```

```
</xs:element>
<xs:element name="DecimalTimeList" type="sml:DecimalTimeListType">
    <xs:annotation>
        <xs:documentation>Ordered list of decimal time values all relative to the same origin
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="DecimalTimeRange" type="sml:DecimalTimeRangeType">
    <xs:annotation>
        <xs:documentation>Pair of decimal time values specifying a range</xs:documentation>
    </xs:annotation>
</xs:element>
<!-- ============================
  global properties
============================== -->
<xs:element name="_property" type="sml:_propertyType"/>
<xs:element name="timeInstant">
    <xs:annotation>
        <xs:documentation>Absolute or relative time</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_propertyType">
                <xs:sequence>
                    <xs:group ref="sml:TimeInstantParameter" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="time">
    <xs:annotation>
        <xs:documentation>Absolute or relative time or time range</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_propertyType">
                <xs:sequence>
                    <xs:group ref="sml:TimeParameter" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!--============================        data structure elements
  ============================-->
<xs:element name="_DataGroup" type="sml:_DataGroupType" abstract="true"/>
<xs:element name="DataGroup" type="sml:DataGroupType" substitutionGroup="sml:_DataGroup">
    <xs:annotation>
        <xs:documentation>Data Group containing data definition (with soft typed data components),
            data values and/or models to derive the values.</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="dataComponent" substitutionGroup="sml:_property">
    <xs:annotation>
        <xs:documentation>Description of data component. The parameter value should be right there if
            'fixed=true' (default) or in the values part if 'fixed=false'</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_dataComponentType">
                <xs:choice minOccurs="0">
                    <xs:group ref="sml:TypedParameter"/>
```

```
                    <xs:group ref="sml:TimeParameter"/>
                </xs:choice>
                <xs:attribute name="name" type="xs:string" use="required"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="values">
    <xs:annotation>
        <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming data provider like
            TML.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:sequence>
            <xs:group ref="sml:DataProviders" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="sml:AssociationAttributes"/>
    </xs:complexType>
</xs:element>
<!-- ========================
   simple types
========================== -->
<xs:simpleType name="CountListSimpleType">
    <xs:restriction base="gml:integerOrNullList"/>
</xs:simpleType>
<xs:simpleType name="CountPairSimpleType">
    <xs:restriction base="sml:CountListSimpleType">
        <xs:length value="2"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="QuantityListSimpleType">
    <xs:list itemType="gml:doubleOrNullList"/>
</xs:simpleType>
<xs:simpleType name="QuantityPairSimpleType">
    <xs:restriction base="sml:QuantityListSimpleType">
        <xs:length value="2"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="TimeStringSimpleType">
    <xs:restriction base="xs:string">
        <xs:enumeration value=""/>
        <xs:enumeration value="notValid"/>
        <xs:enumeration value="currentTime"/>
    </xs:restriction>
</xs:simpleType>
<xs:simpleType name="IsoDateTimeSimpleType">
    <xs:union memberTypes="xs:date xs:time xs:dateTime sml:TimeStringSimpleType"/>
</xs:simpleType>
<xs:simpleType name="IsoDateListSimpleType">
    <xs:list itemType="sml:IsoDateTimeSimpleType"/>
</xs:simpleType>
<xs:simpleType name="IsoDatePairSimpleType">
    <xs:restriction base="sml:IsoDateListSimpleType">
        <xs:length value="2"/>
    </xs:restriction>
</xs:simpleType>
<!-- ========================
   complex types
========================== -->
<xs:complexType name="_propertyType" abstract="true">
    <xs:attributeGroup ref="sml:AssociationAttributes"/>
</xs:complexType>
<xs:complexType name="_axisLinkedPropertyType">
```

```
        <xs:complexContent>
            <xs:extension base="sml:_propertyType">
                <xs:attributeGroup ref="sml:AxisAttributes"/>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="_dataComponentType">
        <xs:annotation>
            <xs:documentation>Type needed cause we derive by restriction to make crs and axisCode
                optional</xs:documentation>
        </xs:annotation>
        <xs:complexContent>
            <xs:restriction base="sml:_axisLinkedPropertyType">
                <xs:attribute name="crs" use="optional"/>
                <xs:attribute name="axisCode" use="optional"/>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="_DataGroupType" abstract="true">
        <xs:attributeGroup ref="sml:IdentificationAttributes"/>
    </xs:complexType>
    <xs:complexType name="DataGroupType">
        <xs:complexContent>
            <xs:extension base="sml:_DataGroupType">
                <xs:sequence>
                    <xs:choice maxOccurs="unbounded">
                        <xs:element name="member">
                            <xs:annotation>
                                <xs:documentation>Sub Data Group allowed here</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="sml:_DataGroup" minOccurs="0"/>
                                </xs:sequence>
                                <xs:attributeGroup ref="sml:AssociationAttributes"/>
                                <xs:attribute name="count" type="xs:positiveInteger" use="optional"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element ref="sml:dataComponent"/>
                    </xs:choice>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="BooleanParameterType">
        <xs:simpleContent>
            <xs:extension base="xs:boolean">
                <xs:attributeGroup ref="sml:ParameterAttributes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="TypedBooleanType">
        <xs:simpleContent>
            <xs:extension base="sml:BooleanParameterType">
                <xs:attributeGroup ref="sml:DefinitionAttributes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="CountParameterType">
        <xs:simpleContent>
            <xs:extension base="gml:integerOrNull">
                <xs:attributeGroup ref="sml:ParameterAttributes"/>
```

```
            <xs:attributeGroup ref="sml:CountAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="CountListType">
    <xs:simpleContent>
        <xs:extension base="sml:CountListSimpleType">
            <xs:attributeGroup ref="sml:ParameterAttributes"/>
            <xs:attributeGroup ref="sml:CountAttributes"/>
            <xs:attributeGroup ref="sml:ListAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="CountRangeType">
    <xs:simpleContent>
        <xs:extension base="sml:CountPairSimpleType">
            <xs:attributeGroup ref="sml:ParameterAttributes"/>
            <xs:attributeGroup ref="sml:CountAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="TypedCountType">
    <xs:simpleContent>
        <xs:extension base="sml:CountParameterType">
            <xs:attributeGroup ref="sml:DefinitionAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="TypedCountListType">
    <xs:simpleContent>
        <xs:extension base="sml:CountListType">
            <xs:attributeGroup ref="sml:DefinitionAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="TypedCountRangeType">
    <xs:simpleContent>
        <xs:extension base="sml:CountRangeType">
            <xs:attributeGroup ref="sml:DefinitionAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="QuantityParameterType">
    <xs:simpleContent>
        <xs:extension base="xs:double">
            <xs:attributeGroup ref="sml:ParameterAttributes"/>
            <xs:attributeGroup ref="sml:QuantityAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="QuantityListType">
    <xs:simpleContent>
        <xs:extension base="sml:QuantityListSimpleType">
            <xs:attributeGroup ref="sml:ParameterAttributes"/>
            <xs:attributeGroup ref="sml:QuantityAttributes"/>
            <xs:attributeGroup ref="sml:ListAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="QuantityRangeType">
    <xs:simpleContent>
        <xs:extension base="sml:QuantityPairSimpleType">
            <xs:attributeGroup ref="sml:ParameterAttributes"/>
```

```
            <xs:attributeGroup ref="sml:QuantityAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="TypedQuantityType">
    <xs:simpleContent>
        <xs:extension base="sml:QuantityParameterType">
            <xs:attributeGroup ref="sml:DefinitionAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="TypedQuantityListType">
    <xs:simpleContent>
        <xs:extension base="sml:QuantityListType">
            <xs:attributeGroup ref="sml:DefinitionAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="TypedQuantityRangeType">
    <xs:simpleContent>
        <xs:extension base="sml:QuantityRangeType">
            <xs:attributeGroup ref="sml:DefinitionAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="DecimalTimeParameterType">
    <xs:simpleContent>
        <xs:extension base="xs:double">
            <xs:attributeGroup ref="sml:ParameterAttributes"/>
            <xs:attributeGroup ref="sml:QuantityAttributes"/>
            <xs:attributeGroup ref="sml:TimeOriginAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="DecimalTimeListType">
    <xs:simpleContent>
        <xs:extension base="sml:QuantityListSimpleType">
            <xs:attributeGroup ref="sml:ParameterAttributes"/>
            <xs:attributeGroup ref="sml:QuantityAttributes"/>
            <xs:attributeGroup ref="sml:ListAttributes"/>
            <xs:attributeGroup ref="sml:TimeOriginAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="DecimalTimeRangeType">
    <xs:simpleContent>
        <xs:extension base="sml:QuantityPairSimpleType">
            <xs:attributeGroup ref="sml:ParameterAttributes"/>
            <xs:attributeGroup ref="sml:QuantityAttributes"/>
            <xs:attributeGroup ref="sml:TimeOriginAttributes"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="IsoDateTimeType">
    <xs:simpleContent>
        <xs:extension base="sml:IsoDateTimeSimpleType">
            <xs:attributeGroup ref="sml:ParameterAttributes"/>
            <xs:attribute name="min" type="sml:IsoDateTimeSimpleType" use="optional"/>
            <xs:attribute name="max" type="sml:IsoDateTimeSimpleType" use="optional"/>
        </xs:extension>
    </xs:simpleContent>
</xs:complexType>
<xs:complexType name="IsoDateTimeListType">
```

```
        <xs:simpleContent>
            <xs:extension base="sml:IsoDateListSimpleType">
                <xs:attributeGroup ref="sml:ParameterAttributes"/>
                <xs:attributeGroup ref="sml:CountAttributes"/>
                <xs:attributeGroup ref="sml:ListAttributes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="IsoDateTimeRangeType">
        <xs:simpleContent>
            <xs:extension base="sml:IsoDatePairSimpleType">
                <xs:attributeGroup ref="sml:ParameterAttributes"/>
                <xs:attribute name="min" type="sml:IsoDateTimeSimpleType" use="optional"/>
                <xs:attribute name="max" type="sml:IsoDateTimeSimpleType" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="URIParameterType">
        <xs:simpleContent>
            <xs:extension base="xs:anyURI">
                <xs:attributeGroup ref="sml:ParameterAttributes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="StringParameterType">
        <xs:simpleContent>
            <xs:extension base="xs:string">
                <xs:attributeGroup ref="sml:ParameterAttributes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="CoefficientParameterType">
        <xs:simpleContent>
            <xs:extension base="xs:double">
                <xs:attributeGroup ref="sml:ParameterAttributes"/>
                <xs:attribute name="number" type="xs:positiveInteger" use="required"/>
                <xs:attribute name="uom" type="xs:anyURI" use="optional"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
    <xs:complexType name="TypedCategoryType">
        <xs:simpleContent>
            <xs:extension base="sml:StringParameterType">
                <xs:attributeGroup ref="sml:DefinitionAttributes"/>
            </xs:extension>
        </xs:simpleContent>
    </xs:complexType>
</xs:schema>
```

A.9 specificationProperties.xsd

Provides base properties for use in specifications and base objects used to define capabilities.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:sml="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
```

```
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>actions definitions for core SensorML</xs:documentation>
    </xs:annotation>
    <!-- ============================
        includes and imports
    ============================== -->
    <xs:include schemaLocation="./base.xsd"/>
    <xs:include schemaLocation="./positionData.xsd"/>
    <!-- ============================
        physical properties
    ============================== -->
    <xs:element name="_physicalProperty" type="sml:_propertyType" abstract="true"
            substitutionGroup="sml:_property">
        <xs:annotation>
            <xs:documentation>Substitution group for all physical properties</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="geometry" substitutionGroup="sml:_physicalProperty">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:_Shape"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="dimensions" substitutionGroup="sml:_physicalProperty">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:QuantityList"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="mass" substitutionGroup="sml:_physicalProperty">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:Quantity"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="material" substitutionGroup="sml:_physicalProperty">
        <xs:annotation>
            <xs:documentation>material (eg. aluminum, polycarbonate)</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:String"/>
                    </xs:sequence>
                </xs:extension>
```

```
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!--========================== capabilities ==========================-->
    <xs:element name="coverage" substitutionGroup="sml:_property">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:LatLonAlt"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="application" substitutionGroup="sml:_property">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:TypedCategory"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!-- ==========================
        shape properties
    ============================ -->
    <xs:element name="_Shape" type="sml:_ShapeType"/>
    <xs:complexType name="_ShapeType" abstract="true">
        <xs:complexContent>
            <xs:extension base="sml:DataGroupType"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:element name="Cone" substitutionGroup="sml:_Shape">
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="sml:_ShapeType">
                    <xs:sequence>
                        <xs:element ref="sml:time" minOccurs="0"/>
                        <xs:element name="apex" maxOccurs="2">
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="sml:Quantity"/>
                                </xs:sequence>
                                <xs:attributeGroup ref="sml:AxisAttributes"/>
                            </xs:complexType>
                        </xs:element>
                        <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded"/>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="Spheroid" substitutionGroup="sml:_Shape">
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="sml:_ShapeType">
                    <xs:sequence>
                        <xs:element ref="sml:time" minOccurs="0"/>
                        <xs:element name="radius" maxOccurs="3">
                            <xs:complexType>
```

```
                        <xs:sequence>
                            <xs:element ref="sml:Quantity"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="sml:AxisAttributes"/>
                    </xs:complexType>
                </xs:element>
                <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
        </xs:restriction>
    </xs:complexContent>
</xs:complexType>
</xs:element>
<xs:element name="Box" substitutionGroup="sml:_Shape">
    <xs:complexType>
        <xs:complexContent>
            <xs:restriction base="sml:_ShapeType">
                <xs:sequence>
                    <xs:element ref="sml:time" minOccurs="0"/>
                    <xs:element name="length" maxOccurs="3">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:Quantity"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="sml:AxisAttributes"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="Line" substitutionGroup="sml:_Shape">
    <xs:complexType>
        <xs:complexContent>
            <xs:restriction base="sml:_ShapeType">
                <xs:sequence>
                    <xs:element ref="sml:time" minOccurs="0"/>
                    <xs:element name="length" minOccurs="0">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:Quantity"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="sml:AxisAttributes"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
</xs:schema>
```

A.10 productAndProcess.xsd

Defines schema for the base response model and general sensor characteristics, such as sensitivity, accuracy, thresholds, operating conditions, etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:sml="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
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:include schemaLocation="./base.xsd"/>
    <xs:include schemaLocation="./component.xsd"/>
    <xs:include schemaLocation="./actions.xsd"/>
    <xs:include schemaLocation="./events.xsd"/>
    <xs:include schemaLocation="./observableObject.xsd"/>
    <!-- ================================================================================
        Substitution groups
    ================================================================================-->
    <xs:element name="_Product" type="sml:ProductType" abstract="true">
        <xs:annotation>
            <xs:documentation>Substitution Group for all Products (including Observed Products
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="_ProcessModel" type="sml:ProcessModelType" abstract="true">
        <xs:annotation>
            <xs:documentation>SubstitutionGroup for all Response Models</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ================================================================================
        Concrete elements
    ================================================================================ -->
    <xs:element name="Product" type="sml:ProductType" substitutionGroup="sml:_Product">
        <xs:annotation>
            <xs:documentation>Provides a description of a measured or derived data product
            </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="ProductList" type="sml:ProductListType" substitutionGroup="sml:_Product"/>
    <xs:element name="ProcessModel" type="sml:ProcessModelType"
            substitutionGroup="sml:_ProcessModel">
        <xs:annotation>
            <xs:documentation>Defines the processes that generated a product</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="ProcessModelGroup" type="sml:ProcessModelGroupType"
            substitutionGroup="sml:ProcessModel">
        <xs:annotation>
            <xs:documentation>Allows a collection of processes</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ================================================================================
        global  types
    ================================================================================ -->
    <xs:complexType name="ProductType">
        <xs:sequence>
            <xs:element ref="sml:identifiedAs" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
```

```
            <xs:element ref="sml:observable" minOccurs="0"/>
            <xs:element ref="sml:providedBy" minOccurs="0"/>
            <xs:element ref="sml:hasCRS" minOccurs="0"/>
            <xs:element ref="sml:locatedUsing" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:derivedFrom" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:data" minOccurs="0"/>
        </xs:sequence>
        <xs:attributeGroup ref="sml:IdentificationAttributes"/>
    </xs:complexType>
    <xs:complexType name="ProductListType">
        <xs:complexContent>
            <xs:restriction base="sml:ProductType">
                <xs:sequence>
                    <xs:element name="member">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:Product" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="sml:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ProcessModelType">
        <xs:sequence>
            <xs:element ref="sml:identifiedAs" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:input" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:output" maxOccurs="unbounded"/>
            <xs:element ref="sml:usesMethod" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:usesParameters" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="intendedFocus" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sml:ObservableObject" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:complexType>
    <xs:complexType name="ProcessModelGroupType">
        <xs:complexContent>
            <xs:extension base="sml:ProcessModelType">
                <xs:sequence>
                    <xs:element name="member" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_ProcessModel" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <!--==========================================================================
        global elements
```

```
    ========================================================================== -->
    <xs:element name="observable">
        <xs:annotation>
            <xs:documentation>Observable describing what the product actually measures (if applicable)
                i.e wind speed, vehicle position...</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:Phenomenon" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="providedBy">
        <xs:annotation>
            <xs:documentation>Links to the component providing this product</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_Component" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="data">
        <xs:annotation>
            <xs:documentation/>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_DataGroup" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="derivedFrom">
        <xs:annotation>
            <xs:documentation>Model used to derive the data</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_ProcessModel" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Phenomenon">
        <xs:complexType>
            <xs:sequence>
                <xs:element name="name" type="xs:string"/>
                <xs:element name="definition" type="xs:string"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="required"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="input">
        <xs:annotation>
            <xs:documentation>Description of input data (DataGroup)</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_Product" minOccurs="0"/>
            </xs:sequence>
```

```
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="output">
        <xs:annotation>
            <xs:documentation>Description of output data (DataGroup)</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_Product" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="usesMethod">
        <xs:annotation>
            <xs:documentation>Description of method used to compute output from input using parameters.
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:ProcessMethod" minOccurs="0">
                    <xs:annotation>
                        <xs:documentation>Method describing the process (text or later MathML)
                        </xs:documentation>
                    </xs:annotation>
                </xs:element>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="usesParameters">
        <xs:annotation>
            <xs:documentation>Description of parameters used by the model (DataGroup)
            </xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_DataGroup" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="usesAction">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:_Action" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="ProcessMethod">
        <xs:annotation>
            <xs:documentation>Method describing the process (text or MathML)</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="parameterDefinitions" minOccurs="0" maxOccurs="unbounded">
                    <xs:complexType>
                        <xs:sequence>
                            <xs:element ref="sml:Discussion" minOccurs="0"/>
                        </xs:sequence>
                        <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
```

```
                </xs:complexType>
            </xs:element>
            <xs:element name="algorithm" minOccurs="0">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sml:Discussion" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                </xs:complexType>
            </xs:element>
            <xs:element ref="sml:reference" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:schema>
```

A.11 sampleAndResponse.xsd

Defines schema for Sample (derived from Product) and ResponseModel (derived from ProcessModel). A ResponseModel defines the general sensor response characteristics, such as sensitivity, accuracy, thresholds, operating conditions, etc.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML" 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:include schemaLocation="./productAndProcess.xsd"/>
    <xs:include schemaLocation="./specificationProperties.xsd"/>
    <!-- ========================================================
        Response Model types
    ========================================================================== -->
    <xs:element name="ResponseModel" type="sml:ResponseModelType"
            substitutionGroup="sml:_ProcessModel">
        <xs:annotation>
            <xs:documentation>Takes _Sample as its input and uses sensor response characteristics as its
                parameters</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="ResponseModelType">
        <xs:complexContent>
            <xs:restriction base="sml:ProcessModelType">
                <xs:sequence>
                    <xs:element ref="sml:identifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="input" minOccurs="0" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_Product" minOccurs="0"/>
```

```
                    </xs:sequence>
                </xs:complexType>
            </xs:element>
            <xs:element ref="sml:output" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element ref="sml:usesMethod" minOccurs="0" maxOccurs="unbounded"/>
            <xs:element name="usesParameters" minOccurs="0" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:sequence>
                        <xs:element ref="sml:_ResponseParameters" minOccurs="0"/>
                    </xs:sequence>
                    <xs:attributeGroup ref="sml:AssociationAttributes"/>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:restriction>
</xs:complexContent>
</xs:complexType>
<xs:element name="_ResponseParameters" type="sml:_DataGroupType"
    substitutionGroup="sml:_DataGroup"/>
<!-- ========================================================
    Sample
========================================================================== -->
<xs:element name="Sample" type="sml:SampleType" substitutionGroup="sml:_Product"/>
<xs:complexType name="SampleType">
    <xs:complexContent>
        <xs:extension base="sml:ProductType">
            <xs:sequence>
                <xs:element ref="sml:dimensions" minOccurs="0"/>
                <xs:element ref="sml:geometry" minOccurs="0"/>
                <xs:element ref="sml:mass" minOccurs="0"/>
                <xs:element ref="sml:material" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
<!-- ========================================================
    common characteristics measures
========================================================================== -->
<xs:element name="_responseModelProperty" type="sml:_propertyType" abstract="true"/>
<xs:element name="relativeAccuracy" substitutionGroup="sml:_responseModelProperty">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_propertyType">
                <xs:sequence>
                    <xs:element ref="sml:TypedQuantityRange" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="resolution" substitutionGroup="sml:_responseModelProperty">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_propertyType">
                <xs:sequence>
                    <xs:element ref="sml:TypedQuantity" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="threshold" substitutionGroup="sml:_responseModelProperty">
    <xs:complexType>
```

```
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:TypedQuantity" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="capacity" substitutionGroup="sml:_responseModelProperty">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:TypedQuantity" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="survivableRange" substitutionGroup="sml:_responseModelProperty">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:TypedQuantityRange" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="operationalRange" substitutionGroup="sml:_responseModelProperty">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:TypedQuantityRange" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="dynamicRange" substitutionGroup="sml:_responseModelProperty">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:TypedQuantityRange" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="measurementMethod" substitutionGroup="sml:_responseModelProperty">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_propertyType">
                    <xs:sequence>
                        <xs:element ref="sml:TypedCategory" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
```

```
</xs:element>
<xs:element name="sampleDuration" substitutionGroup="sml:_responseModelProperty">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_propertyType">
                <xs:sequence>
                    <xs:element ref="sml:Quantity" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="samplePeriod" substitutionGroup="sml:_responseModelProperty">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_propertyType">
                <xs:sequence>
                    <xs:element ref="sml:Quantity" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!-- ========================================================
    Response parameters
========================================================================== -->
<xs:element name="BasicResponse" substitutionGroup="sml:_ResponseParameters">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_DataGroupType">
                <xs:sequence>
                    <xs:element ref="sml:time" minOccurs="0"/>
                    <xs:element ref="sml:_responseModelProperty" maxOccurs="unbounded"/>
                    <xs:element ref="sml:values" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="ImpulseResponse" substitutionGroup="sml:_ResponseParameters">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_DataGroupType">
                <xs:sequence>
                    <xs:element name="timePoints">
                        <xs:annotation>
                            <xs:documentation>Time relative to t0 (time of the input)</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:DecimalTimeList" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="sml:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                    <xs:element name="amplitude">
                        <xs:annotation>
                            <xs:documentation>Amplitude of the output signal for the corresponding time
                                point</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:QuantityList" minOccurs="0"/>
```

```
                        </xs:sequence>
                        <xs:attributeGroup ref="sml:AssociationAttributes"/>
                    </xs:complexType>
                </xs:element>
                <xs:element ref="sml:values" minOccurs="0"/>
            </xs:sequence>
        </xs:extension>
    </xs:complexContent>
</xs:complexType>
</xs:element>
</xs:schema>
```

A.12 positionData.xsd

Parameters and DataGroups defined specifically for specifying position (location and orientation) of components, samples, and products.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:xs="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml" xmlns:sml="http://www.opengis.net/sensorML"
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"/>
    <!-- ========================
        Global substitution groups
    ==========================-->
    <xs:element name="_PositionData" type="sml:_DataGroupType" abstract="true"
            substitutionGroup="sml:_DataGroup">
        <xs:annotation>
            <xs:documentation>Head of substitution Group for Composite Position Data</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="_LocationData" type="sml:_DataGroupType" abstract="true"
            substitutionGroup="sml:_PositionData">
        <xs:annotation>
            <xs:documentation>Head of substitution Group for Location Data</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="_OrientationData" type="sml:_DataGroupType" abstract="true"
            substitutionGroup="sml:_PositionData">
        <xs:annotation>
            <xs:documentation>Head of substitution Group for Orientation Data</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ========================
        Time Data
    ==========================-->
    <xs:element name="TimeData" substitutionGroup="sml:_DataGroup">
        <xs:complexType>
            <xs:complexContent>
```

```
            <xs:extension base="sml:_DataGroupType">
                <xs:sequence>
                    <xs:element ref="sml:time"/>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming
                                data provider like TML.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!-- ==========================
Group of position data
============================-->
<xs:element name="PositionData" substitutionGroup="sml:_PositionData">
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_DataGroupType">
                <xs:sequence>
                    <xs:element ref="sml:time" minOccurs="0"/>
                    <xs:element name="member" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:choice>
                                <xs:element ref="sml:_LocationData"/>
                                <xs:element ref="sml:_OrientationData"/>
                            </xs:choice>
                        </xs:complexType>
                    </xs:element>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming
                                data provider like TML.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!-- ==========================
    Location Data
============================ -->
<xs:element name="Location" substitutionGroup="sml:_LocationData">
    <xs:annotation>
        <xs:documentation>Cartesian location</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_DataGroupType">
                <xs:sequence>
                    <xs:element ref="sml:time" minOccurs="0"/>
                    <xs:element ref="sml:distance" maxOccurs="unbounded"/>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming
                                data provider like TML.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
```

```
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="LatLonAlt" substitutionGroup="sml:_LocationData">
        <xs:annotation>
            <xs:documentation>Location given in Latitude, Longitude and optional Altitude
                coordinates.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_DataGroupType">
                    <xs:sequence>
                        <xs:element ref="sml:time" minOccurs="0"/>
                        <xs:element ref="sml:latitude"/>
                        <xs:element ref="sml:longitude"/>
                        <xs:element ref="sml:altitude" minOccurs="0"/>
                        <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded">
                            <xs:annotation>
                                <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming
                                    data provider like TML.</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="GeoLocation" substitutionGroup="sml:_LocationData">
        <xs:annotation>
            <xs:documentation>LatLonAlt location with optional true heading and speed</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_DataGroupType">
                    <xs:sequence>
                        <xs:element ref="sml:time" minOccurs="0"/>
                        <xs:element ref="sml:latitude"/>
                        <xs:element ref="sml:longitude"/>
                        <xs:element ref="sml:altitude" minOccurs="0"/>
                        <xs:element ref="sml:trueHeading" minOccurs="0"/>
                        <xs:element ref="sml:speed" minOccurs="0"/>
                        <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded">
                            <xs:annotation>
                                <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming
                                    data provider like TML.</xs:documentation>
                            </xs:annotation>
                        </xs:element>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!-- ========================
        Orientation Data
    ========================== -->
    <xs:element name="Orientation" substitutionGroup="sml:_OrientationData">
        <xs:annotation>
            <xs:documentation>Orientation given by a sequence of rotations.  Rotations are applied in the
                order listed.</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_DataGroupType">
```

```
                <xs:sequence>
                    <xs:element ref="sml:time" minOccurs="0"/>
                    <xs:element ref="sml:angle" maxOccurs="unbounded"/>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming
                                data provider like TML.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="PitchRollYaw" substitutionGroup="sml:_OrientationData">
    <xs:annotation>
        <xs:documentation>Orientation given by Pitch, Roll and Yaw. Rotations are applied in the order
            listed.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_DataGroupType">
                <xs:sequence>
                    <xs:element ref="sml:time" minOccurs="0"/>
                    <xs:choice maxOccurs="3">
                        <xs:element ref="sml:pitch"/>
                        <xs:element ref="sml:roll"/>
                        <xs:element ref="sml:yaw"/>
                    </xs:choice>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming
                                data provider like TML.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="Attitude" substitutionGroup="sml:_OrientationData">
    <xs:annotation>
        <xs:documentation>Orientation given by Pitch, Roll, Yaw, and True Heading (and optional
            derivative parameters). Rotations are applied in the order listed.</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_DataGroupType">
                <xs:sequence>
                    <xs:element ref="sml:time" minOccurs="0"/>
                    <xs:element ref="sml:trueHeading" minOccurs="0"/>
                    <xs:choice minOccurs="0" maxOccurs="3">
                        <xs:element ref="sml:pitch"/>
                        <xs:element ref="sml:roll"/>
                        <xs:element ref="sml:yaw"/>
                    </xs:choice>
                    <xs:element ref="sml:angularSpeed" minOccurs="0" maxOccurs="3"/>
                    <xs:element ref="sml:angularAcceleration" minOccurs="0" maxOccurs="3"/>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming
                                data provider like TML.</xs:documentation>
                        </xs:annotation>
```

```
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!-- ==========================
    Mounting Data
============================== -->
<xs:element name="GeoPosition" substitutionGroup="sml:_PositionData">
    <xs:annotation>
        <xs:documentation>Position given by grouping GeoLocation and Attitude</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_DataGroupType">
                <xs:sequence>
                    <xs:element ref="sml:time" minOccurs="0"/>
                    <xs:element ref="sml:latitude"/>
                    <xs:element ref="sml:longitude"/>
                    <xs:element ref="sml:altitude" minOccurs="0"/>
                    <xs:element ref="sml:trueHeading" minOccurs="0"/>
                    <xs:element ref="sml:speed" minOccurs="0"/>
                    <xs:choice minOccurs="0" maxOccurs="3">
                        <xs:element ref="sml:pitch"/>
                        <xs:element ref="sml:roll"/>
                        <xs:element ref="sml:yaw"/>
                    </xs:choice>
                    <xs:element ref="sml:angularSpeed" minOccurs="0" maxOccurs="3"/>
                    <xs:element ref="sml:angularAcceleration" minOccurs="0" maxOccurs="3"/>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming
                                data provider like TML.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="Position" substitutionGroup="sml:_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="sml:_DataGroupType">
                <xs:sequence>
                    <xs:element ref="sml:time" minOccurs="0"/>
                    <xs:choice maxOccurs="unbounded">
                        <xs:element ref="sml:distance"/>
                        <xs:element ref="sml:angle"/>
                    </xs:choice>
                    <xs:sequence minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="sml:speed"/>
                    </xs:sequence>
                    <xs:sequence minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="sml:angularSpeed"/>
                    </xs:sequence>
                    <xs:sequence minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="sml:acceleration"/>
```

```
                    </xs:sequence>
                    <xs:sequence minOccurs="0" maxOccurs="unbounded">
                        <xs:element ref="sml:angularAcceleration"/>
                    </xs:sequence>
                    <xs:element ref="sml:values" minOccurs="0" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Actual data values. Given by a Tuple, a file, a streaming
                                data provider like TML.</xs:documentation>
                        </xs:annotation>
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<!-- ========================
    General Position Properties
============================== -->
<xs:element name="distance" substitutionGroup="sml:_property">
    <xs:annotation>
        <xs:documentation>distance along a coordinate axis</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_axisLinkedPropertyType">
                <xs:sequence>
                    <xs:element ref="sml:Quantity"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="angle" substitutionGroup="sml:_property">
    <xs:annotation>
        <xs:documentation>angle of rotation about a coordinate axis</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_axisLinkedPropertyType">
                <xs:sequence>
                    <xs:element ref="sml:Quantity"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="latitude" substitutionGroup="sml:_property">
    <xs:annotation>
        <xs:documentation>WGS84 EPSG4329 Latitude</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_axisLinkedPropertyType">
                <xs:sequence>
                    <xs:element ref="sml:Quantity"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="longitude" substitutionGroup="sml:_property">
    <xs:annotation>
        <xs:documentation>WGS84 EPSG4329 Longitude</xs:documentation>
```

```
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_axisLinkedPropertyType">
                    <xs:sequence>
                        <xs:element ref="sml:Quantity"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="altitude" substitutionGroup="sml:_property">
        <xs:annotation>
            <xs:documentation>WGS84 EPSG4329 Altitude</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_axisLinkedPropertyType">
                    <xs:sequence>
                        <xs:element ref="sml:Quantity"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="trueHeading" substitutionGroup="sml:_property">
        <xs:annotation>
            <xs:documentation>Angle to North</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_axisLinkedPropertyType">
                    <xs:sequence>
                        <xs:element ref="sml:Quantity"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="phi" substitutionGroup="sml:_property">
        <xs:annotation>
            <xs:documentation>Angle used in local (euler) rotations</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_axisLinkedPropertyType">
                    <xs:sequence>
                        <xs:element ref="sml:Quantity"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="theta" substitutionGroup="sml:_property">
        <xs:annotation>
            <xs:documentation>Angle used in local (euler) rotations</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_axisLinkedPropertyType">
                    <xs:sequence>
                        <xs:element ref="sml:Quantity"/>
                    </xs:sequence>
```

```
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="psi" substitutionGroup="sml:_property">
    <xs:annotation>
        <xs:documentation>Angle used in local (euler) rotations</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_axisLinkedPropertyType">
                <xs:sequence>
                    <xs:element ref="sml:Quantity"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="pitch" substitutionGroup="sml:_property">
    <xs:annotation>
        <xs:documentation>Pitch Angle</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_axisLinkedPropertyType">
                <xs:sequence>
                    <xs:element ref="sml:Quantity"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="roll" substitutionGroup="sml:_property">
    <xs:annotation>
        <xs:documentation>Roll Angle</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_axisLinkedPropertyType">
                <xs:sequence>
                    <xs:element ref="sml:Quantity"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="yaw" substitutionGroup="sml:_property">
    <xs:annotation>
        <xs:documentation>Yaw Angle</xs:documentation>
    </xs:annotation>
    <xs:complexType>
        <xs:complexContent>
            <xs:extension base="sml:_axisLinkedPropertyType">
                <xs:sequence>
                    <xs:element ref="sml:Quantity"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
<xs:element name="speed" substitutionGroup="sml:_property">
    <xs:annotation>
        <xs:documentation>Speed along a coordinate axis</xs:documentation>
```

```
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_axisLinkedPropertyType">
                    <xs:sequence>
                        <xs:element ref="sml:Quantity"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="angularSpeed" substitutionGroup="sml:_property">
        <xs:annotation>
            <xs:documentation>Angular speed about a coordinate axis</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_axisLinkedPropertyType">
                    <xs:sequence>
                        <xs:element ref="sml:Quantity"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="acceleration" substitutionGroup="sml:_property">
        <xs:annotation>
            <xs:documentation>Acceleration along a coordinate axis</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_axisLinkedPropertyType">
                    <xs:sequence>
                        <xs:element ref="sml:Quantity"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="angularAcceleration" substitutionGroup="sml:_property">
        <xs:annotation>
            <xs:documentation>Angular acceleration about a coordinate axis</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_axisLinkedPropertyType">
                    <xs:sequence>
                        <xs:element ref="sml:Quantity"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

A.13 LocationModel.xsd

Base Location Models for all components, samples, and products.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:sml="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>LocationModels for core SensorML</xs:documentation>
    </xs:annotation>
    <!-- ==========================
        includes and imports
    ============================ -->
    <xs:include schemaLocation="./productAndProcess.xsd"/>
    <xs:include schemaLocation="./positionData.xsd"/>
    <!-- ==========================
        abstract substitution groups
    ============================ -->
    <xs:element name="_LocationModel" type="sml:LocationModelType" abstract="true"
            substitutionGroup="sml:_ProcessModel">
        <xs:annotation>
            <xs:documentation>SubstitutionGroup for all locationModels</xs:documentation>
        </xs:annotation>
    </xs:element>
    <!-- ==========================
        location models
        ============================ -->
    <xs:element name="LocationModel" type="sml:LocationModelType" abstract="true"
            substitutionGroup="sml:_LocationModel"/>
    <xs:element name="LocationModelGroup" type="sml:LocationModelGroupType"
            substitutionGroup="sml:_LocationModel"/>
    <!-- ==========================
       complex types
    ============================ -->
    <xs:complexType name="LocationModelType" abstract="true">
        <xs:complexContent>
            <xs:restriction base="sml:ProcessModelType">
                <xs:sequence>
                    <xs:element ref="sml:identifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:sourceCRS"/>
                    <xs:element ref="sml:referenceCRS"/>
                    <xs:element ref="sml:input" minOccurs="0"/>
                    <xs:element ref="sml:output" minOccurs="0"/>
                    <xs:element ref="sml:usesMethod" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:usesParameters" minOccurs="0" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="LocationModelGroupType">
        <xs:complexContent>
            <xs:extension base="sml:LocationModelType">
                <xs:sequence>
                    <xs:element name="member" maxOccurs="unbounded">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:_LocationModel" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
                        </xs:complexType>
```

```
                    </xs:element>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <!--==========================
      global elements
    ============================ -->
    <xs:element name="ElementID" substitutionGroup="sml:_DataGroup">
        <xs:complexType>
            <xs:complexContent>
                <xs:extension base="sml:_DataGroupType">
                    <xs:sequence>
                        <xs:element ref="sml:time" minOccurs="0"/>
                        <xs:element name="elementNumber" minOccurs="0">
                            <xs:annotation>
                                <xs:documentation>Number identifying an element in an array or group
                                </xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:sequence>
                                    <xs:element ref="sml:Count"/>
                                </xs:sequence>
                            </xs:complexType>
                        </xs:element>
                        <xs:element ref="sml:values" minOccurs="0"/>
                    </xs:sequence>
                </xs:extension>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="referenceCRS">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="gml:_CoordinateReferenceSystem" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="sourceCRS">
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="gml:EngineeringCRS" minOccurs="0"/>
            </xs:sequence>
            <xs:attributeGroup ref="sml:AssociationAttributes"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

A.14 basicLocationModels.xsd

Provides derived location models common for many components, samples, and products.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:sml="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
```

```
xmlns:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <!-- ==========================
        includes and imports
    ============================== -->
    <xs:include schemaLocation="./locationModel.xsd"/>
    <xs:include schemaLocation="./positionData.xsd"/>
    <!-- ==========================
       Geo Location Model
    ============================== -->
    <xs:element name="GeoPositionModel" type="sml:GeoPositionModelType"
            substitutionGroup="sml:_LocationModel">
        <xs:annotation>
            <xs:documentation>Provides location of SourceCRS in ReferenceCRS, given Lat, Lon Alt
                location. Reference CRS should always be a global planetary CRS (ECF, ECI or equivalent
                for other planets). If a state is also provided, velocity and acceleration can be used to
                interpolate location values.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="GeoPositionModelType">
        <xs:complexContent>
            <xs:restriction base="sml:LocationModelType">
                <xs:sequence>
                    <xs:element ref="sml:identifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:sourceCRS"/>
                    <xs:element ref="sml:referenceCRS"/>
                    <xs:element ref="sml:input" minOccurs="0"/>
                    <xs:element ref="sml:output" minOccurs="0"/>
                    <xs:element ref="sml:usesMethod" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="usesParameters" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Lat Lon Alt position of Source CRS</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:choice minOccurs="0">
                                <xs:element ref="sml:GeoLocation"/>
                                <xs:element ref="sml:Attitude"/>
                                <xs:element ref="sml:GeoPosition"/>
                                <xs:element ref="sml:Position"/>
                            </xs:choice>
                            <xs:attributeGroup ref="sml:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <!-- ==========================
       Mount Position Model
    ============================== -->
    <xs:element name="MountingModel" type="sml:MountingModelType"
            substitutionGroup="sml:_LocationModel">
        <xs:annotation>
            <xs:documentation>Provides location and orientation of Source CRS inside Reference CRS,
                which can be any kind of cartesian CRS. The output is derived either explicitely if parameters
                consist of position data only. If a state is provided, velocity and acceleration can be used to
                interpolate position values.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="MountingModelType">
        <xs:complexContent>
```

```
            <xs:restriction base="sml:LocationModelType">
                <xs:sequence>
                    <xs:element ref="sml:identifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:sourceCRS"/>
                    <xs:element ref="sml:referenceCRS"/>
                    <xs:element ref="sml:input" minOccurs="0"/>
                    <xs:element ref="sml:output" minOccurs="0"/>
                    <xs:element ref="sml:usesMethod" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="usesParameters" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Position (Location and Orientation) of Source CRS in
                                Reference CRS</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:Position" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="sml:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <!-- ==========================
        Mount Position Model
    ============================ -->
    <xs:element name="AttitudeModel" type="sml:AttitudeModelType"
            substitutionGroup="sml:_LocationModel">
        <xs:annotation>
            <xs:documentation>Provides location and orientation of Source CRS inside Reference CRS,
                which can be any kind of cartesian CRS. The output is derived either explicitely if parameters
                consist of position data only. If a state is provided, velocity and acceleration can be used to
                interpolate position values.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="AttitudeModelType">
        <xs:complexContent>
            <xs:restriction base="sml:LocationModelType">
                <xs:sequence>
                    <xs:element ref="sml:identifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:sourceCRS"/>
                    <xs:element ref="sml:referenceCRS"/>
                    <xs:element ref="sml:input" minOccurs="0"/>
                    <xs:element ref="sml:output" minOccurs="0"/>
                    <xs:element ref="sml:usesMethod" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="usesParameters" maxOccurs="unbounded">
                        <xs:annotation>
                            <xs:documentation>Position (Location and Orientation) of Source CRS in
                                Reference CRS</xs:documentation>
                        </xs:annotation>
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:Attitude" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="sml:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
```

```
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <!-- ==========================
      Identity Mounting Model
    ============================-->
    <xs:element name="IdentityModel" substitutionGroup="sml:_LocationModel">
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="sml:LocationModelType">
                    <xs:sequence>
                        <xs:element ref="sml:identifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element ref="sml:description" minOccurs="0"/>
                        <xs:element ref="sml:sourceCRS"/>
                        <xs:element ref="sml:referenceCRS"/>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!-- ==========================
       Grid Location Model
    ============================-->
    <xs:element name="RegularGridModel" type="sml:RegularGridModelType"
            substitutionGroup="sml:_LocationModel">
        <xs:annotation>
            <xs:documentation>Used to specify location of elements in arrays.</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:complexType name="RegularGridModelType">
        <xs:complexContent>
            <xs:restriction base="sml:LocationModelType">
                <xs:sequence>
                    <xs:element ref="sml:identifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element ref="sml:sourceCRS"/>
                    <xs:element ref="sml:referenceCRS"/>
                    <xs:element ref="sml:input" minOccurs="0"/>
                    <xs:element ref="sml:output" minOccurs="0"/>
                    <xs:element ref="sml:usesMethod" minOccurs="0" maxOccurs="unbounded"/>
                    <xs:element name="usesParameters">
                        <xs:complexType>
                            <xs:sequence>
                                <xs:element ref="sml:GridParameters" minOccurs="0"/>
                            </xs:sequence>
                            <xs:attributeGroup ref="sml:AssociationAttributes"/>
                        </xs:complexType>
                    </xs:element>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <!-- ==========================
       Grid Parameters
    ============================-->
    <xs:element name="GridParameters" substitutionGroup="sml:_DataGroup">
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="sml:_DataGroupType">
                    <xs:sequence>
                        <xs:element ref="sml:time" minOccurs="0"/>
```

```
                    <xs:sequence maxOccurs="3">
                        <xs:annotation>
                            <xs:documentation>One to Three dimensions can be specified here
                            </xs:documentation>
                        </xs:annotation>
                        <xs:element name="dimension">
                            <xs:annotation>
                                <xs:documentation>Number of elements in the grid along the specified
                                    axis</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:extension base="sml:_axisLinkedPropertyType">
                                        <xs:sequence>
                                            <xs:element ref="sml:Count" minOccurs="0"/>
                                        </xs:sequence>
                                    </xs:extension>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                        <xs:element name="spacing">
                            <xs:annotation>
                                <xs:documentation>spacing between elements along the specified
                                    axis</xs:documentation>
                            </xs:annotation>
                            <xs:complexType>
                                <xs:complexContent>
                                    <xs:extension base="sml:_axisLinkedPropertyType">
                                        <xs:sequence>
                                            <xs:element ref="sml:Quantity" minOccurs="0"/>
                                        </xs:sequence>
                                    </xs:extension>
                                </xs:complexContent>
                            </xs:complexType>
                        </xs:element>
                    </xs:sequence>
                    <xs:element ref="sml:values" minOccurs="0"/>
                </xs:sequence>
                <xs:attribute name="crs" use="required"/>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
</xs:element>
</xs:schema>
```

A.15 dataProvider.xsd

Defines the tuple structure and the base schema for deriving DataProviders.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:sml="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
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:include schemaLocation="./base.xsd"/>
    <xs:include schemaLocation="./parameters.xsd"/>
    <!-- ======================================================================
        Substitution groups
    ========================================================================-->
    <xs:element name="_DataProvider">
        <xs:annotation>
            <xs:documentation>Substitution Group for any Data Provider</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:group name="DataProviders">
        <xs:choice>
            <xs:element ref="sml:TupleData"/>
            <xs:element ref="sml:_DataProvider"/>
        </xs:choice>
    </xs:group>
    <!-- ======================================================================
        Data Values Providers
    ======================================================================= -->
    <xs:element name="TupleData">
        <xs:complexType>
            <xs:simpleContent>
                <xs:extension base="sml:tupleStringSimpleType">
                    <xs:attributeGroup ref="sml:IdentificationAttributes"/>
                    <xs:attribute name="length" type="xs:positiveInteger" use="optional"/>
                </xs:extension>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <!-- ======================================================================
        Global types
    ======================================================================= -->
    <xs:simpleType name="tupleStringSimpleType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
</xs:schema>
```

A.16 observableObject.xsd

Base schema for defining intended objects observed by sensors.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (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:xlink="http://www.w3.org/1999/xlink" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <!-- ==========================
        includes and imports
    ============================ -->
    <xs:include schemaLocation="./specificationProperties.xsd"/>
    <!-- ==========================
        global substitution group elements
    ============================ -->
```

```
<xs:element name="_ObservableObject" type="sml:ObservableObjectType" abstract="true">
    <xs:annotation>
        <xs:documentation>SubstitutionGroup for all ObservableObjects</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="_ObservableCharacteristics" abstract="true">
    <xs:annotation>
        <xs:documentation>SubstitutionGroup for all  ObservableObject Characteristics
        </xs:documentation>
    </xs:annotation>
</xs:element>
<!-- ============================
    global concrete values
============================== -->
<xs:element name="ObservableObject" type="sml:ObservableObjectType">
    <xs:annotation>
        <xs:documentation>An object capable of being sensed by a sensor (e.g. the Earth's surface, a
            tank, a person)</xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="ObservableObjectArray" type="sml:ObservableObjectArrayType"
        substitutionGroup="sml:_ObservableObject">
    <xs:annotation>
        <xs:documentation>Collection of disparate samples capable of being sensed by a sensor
        </xs:documentation>
    </xs:annotation>
</xs:element>
<xs:element name="ObservableObjectGroup" type="sml:ObservableObjectGroupType"
        substitutionGroup="sml:_ObservableObject">
    <xs:annotation>
        <xs:documentation>Collection of disparate objects capable of being sensed by a sensor
        </xs:documentation>
    </xs:annotation>
</xs:element>
<!-- ============================
    Complex Types
============================== -->
<xs:complexType name="ObservableObjectType">
    <xs:sequence>
        <xs:element ref="sml:identifiedAs" maxOccurs="unbounded"/>
        <xs:element ref="sml:classifiedAs" minOccurs="0" maxOccurs="unbounded"/>
        <xs:element name="observableProperties" minOccurs="0">
            <xs:annotation>
                <xs:documentation>Can include properties such as geometry, electromagnetic response,
                    pattern, etc.</xs:documentation>
            </xs:annotation>
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="sml:_ObservableCharacteristics"/>
                </xs:sequence>
                <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
            </xs:complexType>
        </xs:element>
        <xs:element name="hasGeometry" minOccurs="0" maxOccurs="unbounded">
            <xs:complexType>
                <xs:sequence>
                    <xs:element ref="gml:_Geometry"/>
                </xs:sequence>
                <xs:attributeGroup ref="gml:AssociationAttributeGroup"/>
            </xs:complexType>
        </xs:element>
    </xs:sequence>
    <xs:attribute name="id" type="xs:ID" use="optional"/>
```

```
    </xs:complexType>
    <xs:complexType name="ObservableObjectGroupType">
        <xs:complexContent>
            <xs:extension base="sml:ObservableObjectType"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="ObservableObjectArrayType">
        <xs:complexContent>
            <xs:extension base="sml:ObservableObjectType"/>
        </xs:complexContent>
    </xs:complexType>
</xs:schema>
```

A.17 gmlStub.xsd

Allows import of multiple schemas from GML.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- 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/gml" xmlns:gml="http://www.opengis.net/gml" xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:include schemaLocation="../gml/3.0.1/base/temporal.xsd"/>
    <xs:include schemaLocation="../gml/3.0.1/base/coordinateReferenceSystems.xsd"/>
    <xs:include schemaLocation="../gml/3.0.1/base/coordinateSystems.xsd"/>
</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 .