Open GIS Consortium Inc.

Date: 2002-12-20

Reference number of this OpenGIS© Project Document: **OGC 02-026r4**

Version: 0.7

Category: OpenGIS© Discussion Paper

Editor: Mike Botts
University of Alabama in Huntsville

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

Issue Name: [**Schema is released for Public Discussion only**. (meb, 2002-12-05)]

Issue Description:

The SensorML schemas in this document have not gone through the approval process by OGC and are released here only 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. Comments should be sent to mike.botts@nsstc.uah.edu.

Resolution:

Document type: OpenGIS© Interoperability Program Report
Document subtype: Engineering Specification
Document stage: Report
Document language: English

Table of Contents

Preface

Submitting organizations

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

CSIRO Australia
Galdos Systems, Inc
University of Alabama in Huntsville

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	University of Alabama in Huntsville	ESSC / NSSTC Huntsville, AL 35899	+01-256-961-7760	mike.botts@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

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

Recommended Changes to the OpenGIS Abstract Specification

The OpenGIS© 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 part of OGC 02-026 may be the subject of patent rights. The Open GIS Consortium Inc. shall not be held responsible for identifying any or all such patent rights.

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

Part 1. Introduction

1.1. SCOPE

For the purpose of this discussion, SensorML is considered as one key component of the Sensor Web Enablement. 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 OGC Open Web Services 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, the sensor system has 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 to 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 initiative, values returned by sensors are accounted for within the Observations and Measurements schemas.

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.

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.

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 the 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 does not provide a detailed description of the hardware design of a sensor but rather it is a general schema for describing a functional model 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 specialized with 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 mathematical models typically hide the characteristics of the sensor, and allow for geolocation of sensor data through the use of polynomial functions. Different mathematical models can be designed to define sample location within a variety of coordinate systems, including the local sensor frame, the platforms local frame, or a geographic reference coordinate frame.

For the case of rigorous sensor models, we typically separate the description of the sensor from that of its platform. In cases where confusion is likely we will uses 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 to a static or dynamic platform (or an assembly of nested platforms).

The detailed description of the mechanical structure of the platform(s) is outside the scope of the sensor description. From our perspective, the main role of the platform is to define the (possibly dynamic) coordinate system(s) in which the sensor measurements are taken, so that these measurements can be related to some relevant external coordinate system. 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 not a part of the sensor description, per se, it is of vital importance to our ability to georegister sensor measurements. Thus, an initial simple schema for defining platforms is provided in this document.

As will be discussed further, SensorML is suitable for both in-situ and remote sensor, whether mounted to static or mobile platform. The original SensorML, developed before 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 OGC OWS 1.0 program, the previous release of this document focused primarily on static, in-situ sensors. In line with the focus of OGC OWS 1.2, this release refines and extends the previous schema and description to support dynamic, remote sensors, as well.

1.2. CONFORMANCE

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

1.3. NORMATIVE REFERENCES

The following normative documents contain provisions, which through reference in this text, constitute provisions of this part of OGC 02-026. For dated references, subsequent amendments to, or revisions of, any of these publications do not apply. However, parties to agreements based on this part of OGC 02-026 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.

OGC Abstract Specification:

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

OGC Implementation Specifications:

OGC Observation and Measurements DIPR, OGC 02-027r3 (Version 1.2.0)

OGC Coordinate Transformation Services Implementation Specification, Version 1.0

OGC Geography Markup Language, Version 3.0

1.4. TERMS AND DEFINITIONS

For the purposes of this document, the following terms and definitions apply /the terms and definitions given in … and the following apply.

Observable
A phenomenon to that can be observed and measured, such as temperature, gravity, chemical concentration, orientation, number-of-individuals. The equivalent term is **measurand** when the values are determined by measurement.

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.

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.

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.

(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 frame and to which the frames of attached sensors and platforms can be referenced.

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

1.5. CONVENTIONS

1.5.1. 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 GIS Consortium
OWS	Open Web Services
ows:*	schema namespace for Observations and Measurement
UML	Unified Modeling Language

SensorML	Sensor Model Language
sml:*	schema namespace for SensorML
WGS84	World Geodetic System 84
XML	eXtended Markup Language
xs:*	schema namespace for XMLSchema.2002
1D	One Dimensional
2D	Two Dimensional
3D	Three Dimensional

1.5.2. 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.

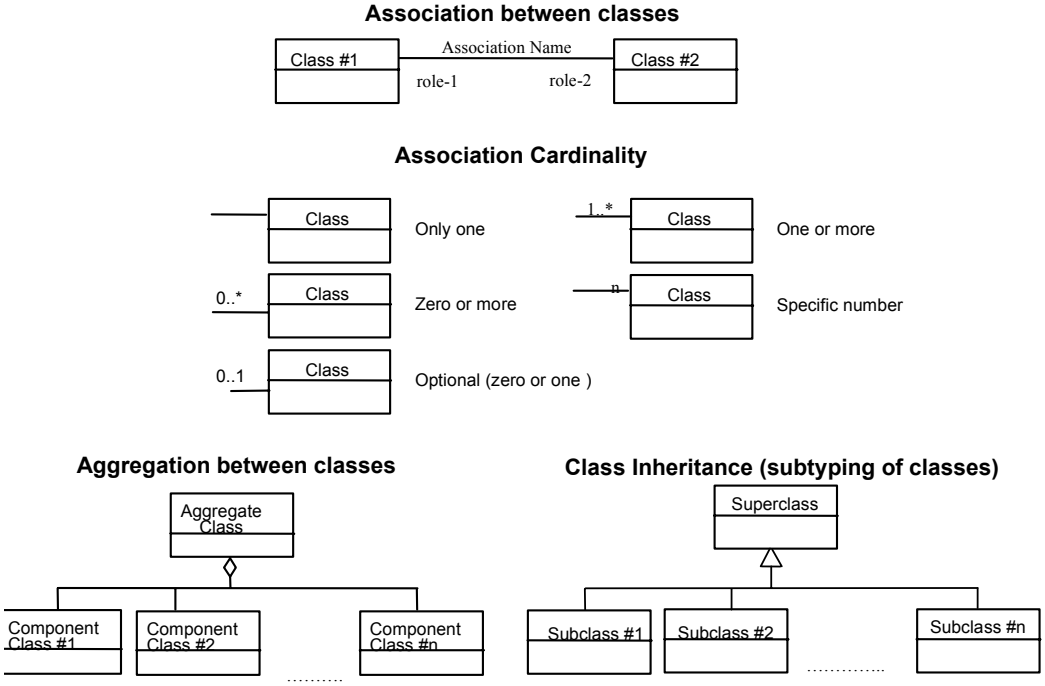


Figure 1.1 — UML notation

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

1.6. BACKGROUND

1.6.1. 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.

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

(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]*.

1.6.2. 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 OpenGIS 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).

1.6.3. 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.

1.6.4. Importance to Sensor Web Enablement

Issue Name: [INCOMPLETE. (meb, 2002-04-16)]

Issue Description: Provide text regarding:

- Sensor autonomy and communication between sensors
- Web friendly access to information required to find, process, and geolocate sensor data
- Ability to bypass time consuming process of collecting data at a data centers, geolocating and processing all data to given data level, and then aggregating data into a data product that meets general needs of data community. Instead can enable immediate access to subset of the data of interest to the user and allow processing and mapping to custom desires.
- Sensor systems can publish alerts on which other sensor systems can act

Resolution:

1.7. 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

Part 2. Design and Specification

*** SensorML is key component of the OGC Sensor Web Enablement framework, but can be used independently of the framework ***

2.1. DESIGN CRITERIA AND ASSUMPTIONS

2.1.1. 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.

2.1.2. Sensor Group Concepts

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

A sensor package 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”. Sensor package 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.

2.1.3. Relationship of the sensor to a platform

A sensor system is composed of three main elements as shown in the UML diagram in Figure 2.1. 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.

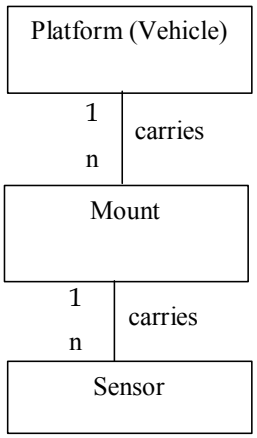


Figure 2.1 Diagram showing basic relationships between sensor, mount, and platform.

The main importance of the mount is that it defines the position and orientation of the measured quantity for each carried sensor in a mount defined frame of reference. Within the design of SensorML, a mount is considered a special case of a platform (in particular, an *AttachedPlatform*). The position and orientation of the mount are then defined with respect to another mount-defined frame of reference and ultimately to the platform-defined frame of reference. All of the frames of reference can in general be considered to

be moving. Figure 2.2 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). The concept of frames is introduced in Section 2.1.4.

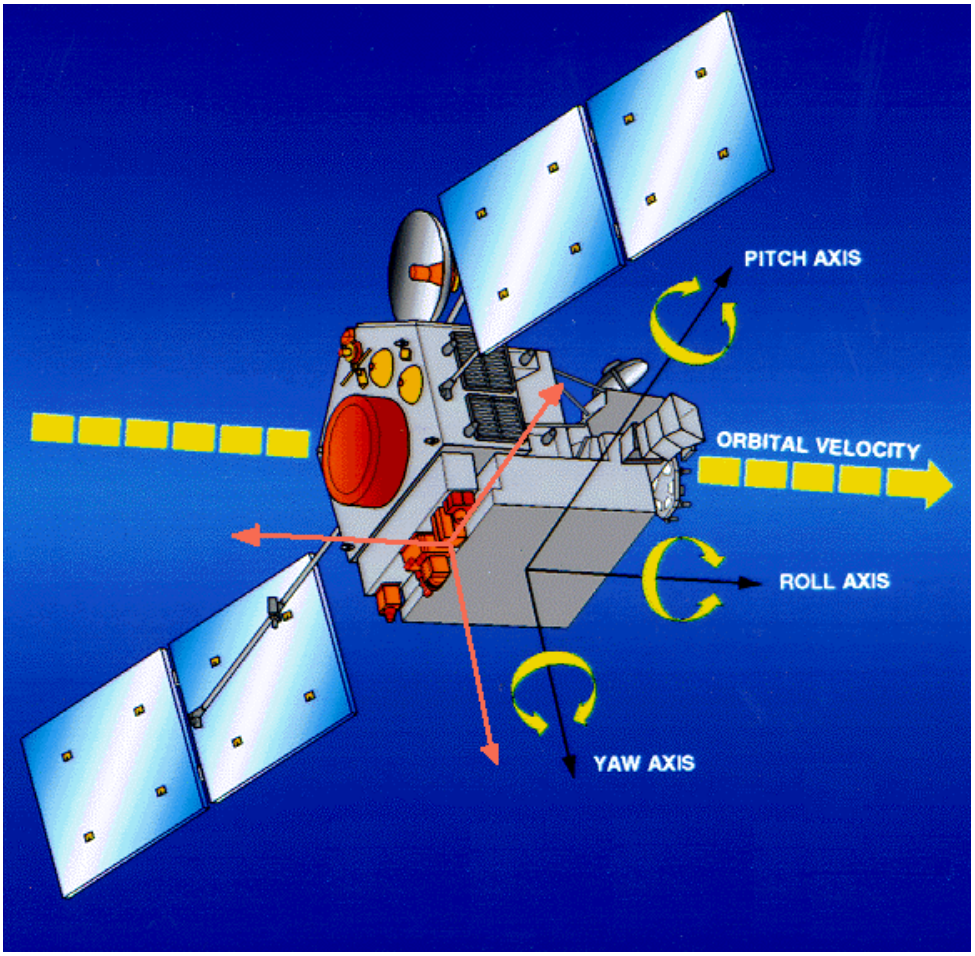


Figure 2.2 Relationship of sensor frame (pink) to the moving platform frame (black).

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 in Section 1.1, we separate the description of the sensor from that of its mount and 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 2.1. Relationships between in-situ and remote sensors and dynamic and fixed platforms.

2.1.4. Concept of 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.

2.1.5. 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. However, the SensorML document does not define the value of these observations, nor where the values are stored.

The definition of the Observation schema 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. For a definition of the Observations and Measurements schema, see the Observations and Measurements IPR [OGC 02-027].

2.1.6. 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.

2.1.7. 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. These will be discussed in more detail in Section 2.2.10.

2.2. SENSORML CORE DEFINITIONS

2.2.1. Main components of the sensor description

The root for all SensorML documents is *Sensor*, or an extension of *Sensor* such as *SensorGroup*. The *Sensor* has required attribute, *id* (of type *gml:id*) as a. This unique identifier allows one to link to this sensor's description of a sensor instance where appropriate, whether that description is internal or external. Based on the concepts above and as shown in Figure 2.3, the sensor description is divided into nine main components of information, including:

- Basic **sensor identification**, such as a unique identifier and names, as well as sensor type
- **Document constraints**, which provides the valid time range over which this instance is valid, the classification level, use restrictions, and the last modification date of the sensor document
- The **platform** to which the sensor is **attached**
- The definition of the Sensor's **coordinate reference system (CRS)**
- The **location of the sensor** relative to a geographical reference system or a local coordinate reference system
- A description of the measurement characteristics. These include:
    - the **observables** that can be **measured**,
    - the sensor's response characteristics relative to each of these observables (e.g. sensitivity, range, quality, environmental constraints, etc.)
    - the sensor model parameters used for geolocating the sensor's observations; i.e. the information required to determine the **geometric and temporal characteristics** of the samples in geodetic space and real-world time
- Information regarding the **operator**(s) of the sensor, as well as any possible links to a **Sensor Planning Service** (SPS)
- The **sensor description** or **metadata**, including the manufacturer, model name, serial number, and history of the sensor
- SensorML **documentation description** or **metadata,** such as who created this document, what modifications have been made/when, what assumptions were used, and what other references and informative web sites are available

Each of the these components will be discussed in detail in the sections below. Notice that the only required properties (those outlined with solid lines in Figure 2.3) for the *Sensor* are *identifiedAs*, *documentConstraints*, and *measures*. The other properties are

optional so that a sensor description need only contain those properties that are relevant to the particular sensor type or sensor purpose.

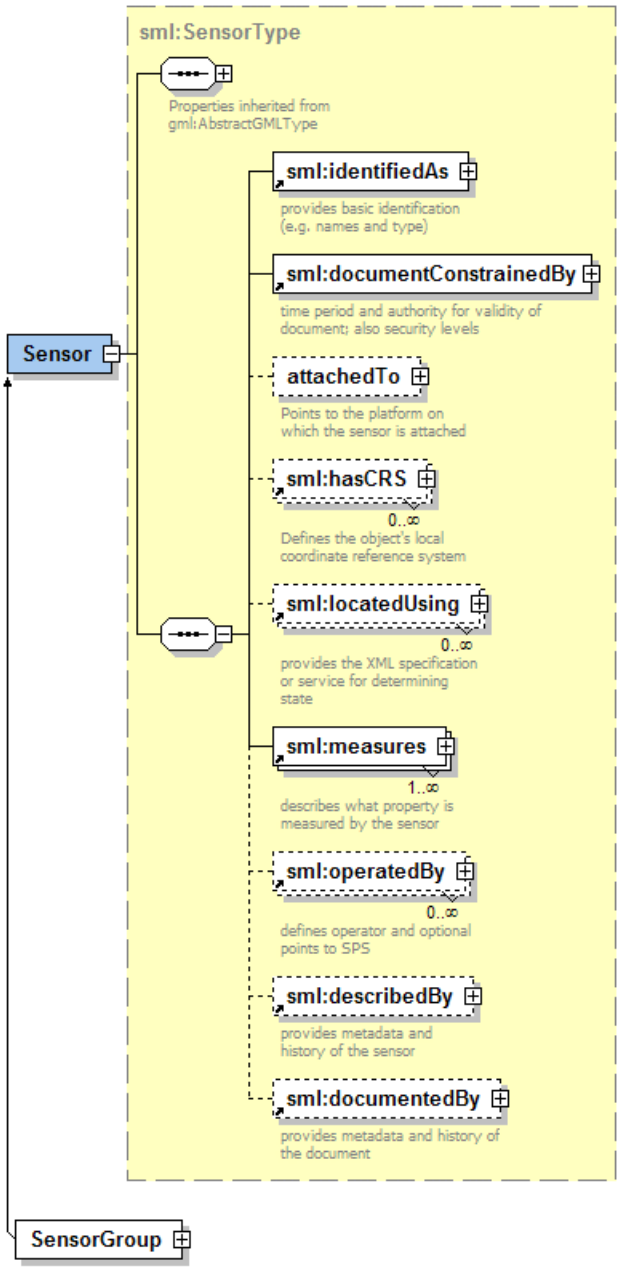


Figure 2.3 Schema diagram showing the highest level of the SensorML document structure.

A SensorML instance document utilizes the practice of self-encapsulation; that a SensorML instance document includes metadata about itself. While there is an opposing practice that considers that all metadata about an entity should only be retained outside of that entity, we have chosen to include vital metadata about the document (such as authors, history, valid times, constraints, etc) within the instance document. This is similar to a Word document that internally retains information about its creator, the data

last modified, and even its revision history. It is felt that this allows a given instance document to be more independent and self-reliant. Not only does this not preclude relevant information about the document from also being retained externally in a registry, it is perceived that this practice will greatly assist a sensor registry from mining any information of relevance directly from the SensorML instance. The two SensorML properties that contain information about the document, itself, include the *documentConstrainedBy* and *documentedBy* properties.

Issue Name: [**gml:AbstractGMLType Properties**. (meb, 2002-12-18)]

Issue Description:

Sensor is now of the type *gml:AbstractGMLType* and thus inherits the properties *gml:metaDataProperty*, *gml:description*, and *gml:name*. These properties are currently redundant to some of the properties defined for *Sensor*. There is a need to consider whether it makes sense to consolidate some of these.

Resolution:

As discussed in previous sections, a collection of sensors can itself be a sensor, as shown in the schema in Figure 2.4. The derived *SensorGroup* types include the *SensorArray*, which consist of a group of sensors which provide an array of like measurements and *SensorPackage* which can provide a derived measurement based on the collection of dissimilar sensor measurements, as discussed in Section 2.1.2.

Issue Name: [**Sensor Group Division**. (meb, 2002-03-26)]

Issue Description:

Although these divisions make conceptual sense, it is unclear at present whether they need to be defined and utilized in the *SensorGroup* cases. We may find that its better not to make a definite distinction of the two within the SensorML schema; it may complicate issues, particularly when a sensor collection may have properties of both

Resolution:

A *SensorGroup* may define its *Sensor* members immediately within the *member* property, as in:

```
<SensorGroup…>
    …
    <sensorMember>
        <Sensor gml:id=”thisSensorsUniqueID”  …>
            <identifiedAs>  …   </ identifiedAs >
            <documentConstrainedBy>  …  </ documentConstrainedBy >
            <measures>  …  </ measures >
            …
        </Sensor>
    </sensorMember>
    …
</SensorGroup>
```

Alternatively, a *SensorGroup* can use a *SensorReference* to link to either an internally defined sensor (using *xlink:href*):

```
<sensorMember  xlink:href="#sensor_1"/>
```

or to an externally-defined Sensor:

```
<sensorMember xlink:href="http://some.url/mySensor.xsd#sensor_1"/>
```

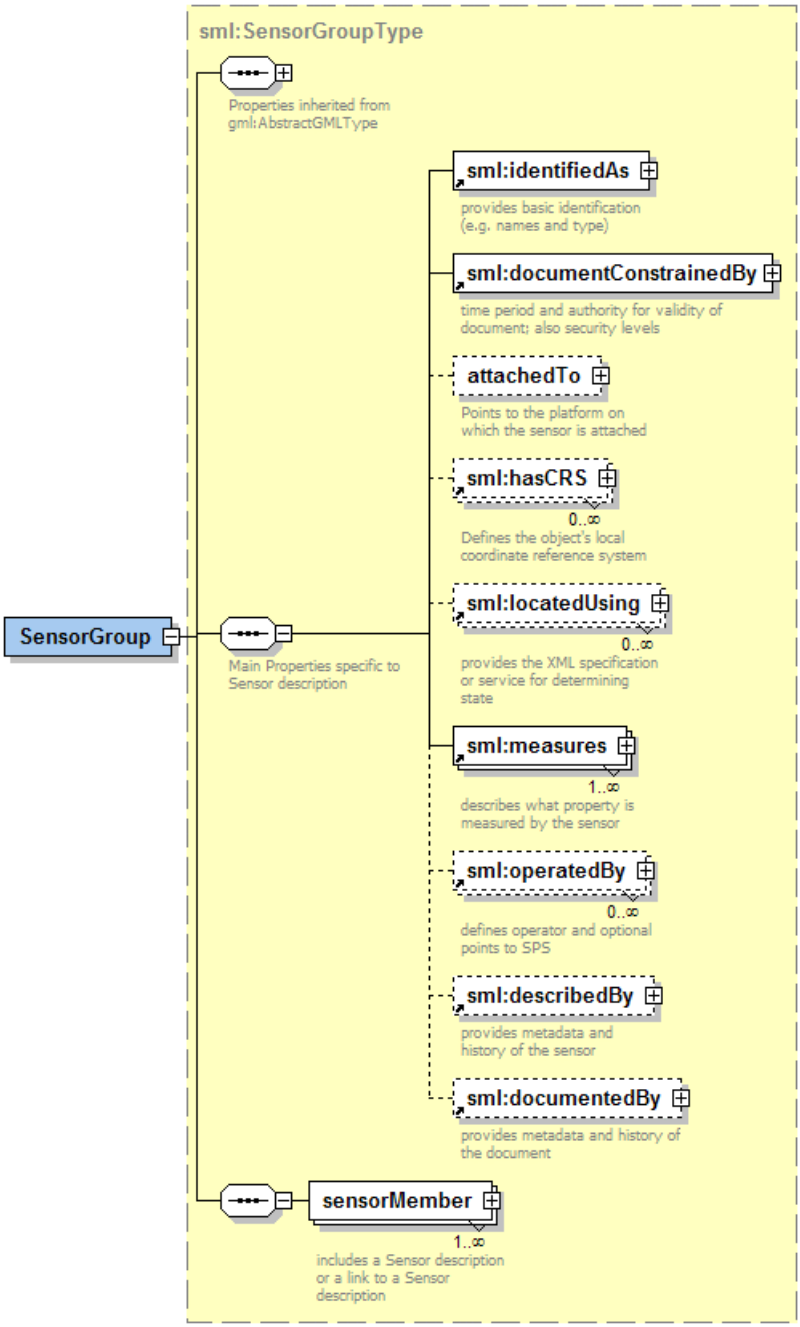


Figure 2.4 Schema defining SensorGroup.

2.2.2. Sensor Identification

The *identifiedAs* property provides fundamental high-level information regarding the sensor. As shown in Figure 2.5, these include *shortName* and *longName* properties which take xs:string as their value, and *sensorType*, which provides an xs:anyURI pointer. The *sensorType* pointer is expected to link to an entry in a sensor taxonomy dictionary.

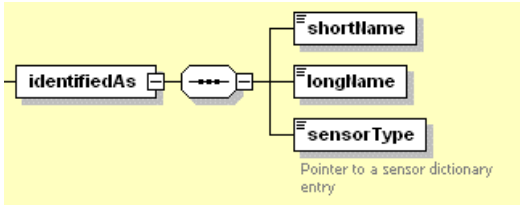


Figure 2.5. Schema diagram showing sensor identification component.

An example of the root *Sensor* and *identifiedAs* properties is given by:

```
<Sensor xmlns="http://www.opengis.net/sensorML" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sensorML sensorML.xsd" id="ssmi_F08"
documentVersion="meb_0.01" documentDate="2002-08-14T03:25:00.000">
    <identifiedAs>
        <shortName>SSM/I F08</shortName>
        <longName>
            Special Sensor Microwave/Imager on Defense Meteorological Satellite Program (DMSP)
            F08
        </longName>
        <sensorType>http:/www.opengis.net/sensors#conicScanner</sensorType>
    </identifiedAs>
    …
</Sensor>
```

2.2.3. Document Constraints

The constraints imposed on the sensor instance document are listed within the *documentConstrainedBy* property. As shown in Figure 2.6, the constraints include validTime (*gml:TimePrimitive*), securityLevel (*iso19115:MD_SecurityConstraintType*), and legalUse (*iso19115:MD_LegalConstraintType*).

The *validTime* property specifies the time of observation for which this sensor description applies, and can be either a range or an instance. For some sensor descriptions (e.g. a scanner on a satellite platform), the sensor description may not change often and the *validTime* range may be tens of years. For other sensor descriptions that are associated, perhaps, with a single image or sample, the sensor description may be valid for only an instant in time. This range over time for which a given SensorML instance document is valid is often a function of the complexity of the sensor, the design of the sensor instance, and the type of sensor model used for geolocation.

The *legalUse* property specifies any legal constraints, perhaps related to copyrights, commercial use, etc., while the *securityLevel* refers to the security classification level established by military or intelligence agencies, perhaps.

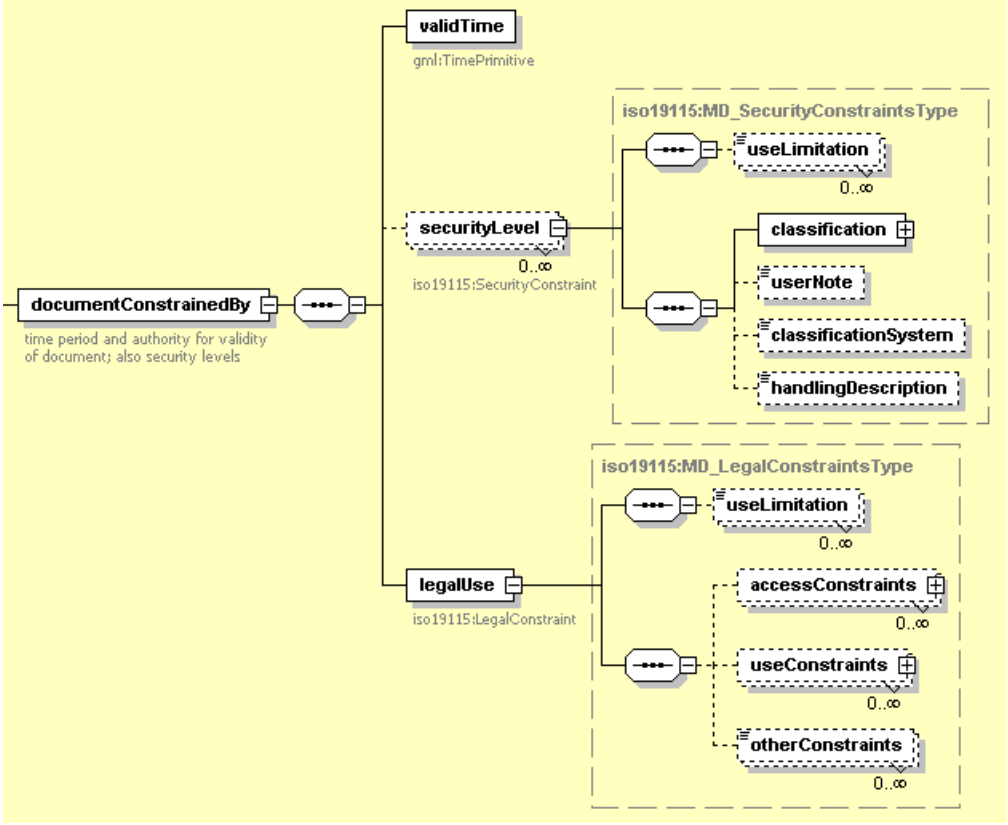


Figure 2.6. Schema diagram of the sensor constraint specifications.

2.2.4. Sensor Platform

The optional *attachedTo* property takes a *Platform* as it value. The *Platform* description can be either defined within the sensor instance document or be referenced through a URI. Typically, except in very simple cases (such as a in-situ sensor on a simple static platform), it is expected that the *Platform* definition will be external to the *Sensor* definition. As with the *Sensor* definition, there is a *id* (type xs:id) attribute that serves as a unique identifier for that platform and an optional p*latformReference* attribute that allows one to link to an internally or externally defined *Platform*:

```
<attachedTo lnik:href="http://www.opengis.net/platforms.xsd#myPlatform">
```

The *Platform* schema defined in SensorML is currently rather basic and perhaps incomplete. Its primary focus is to provide a means of defining a coordinate reference system that can assist in determining the geographic and orientation of the sensor, as well the geolocation of the sensor's observation.

As shown in Figure 2.7, *Platform* has several properties that are identical to *Sensor*, including *identifiedAs*, *documentConstrainedBy*, *hasCRS*, *locatedUsing*, *describedBy*,

and *documentedBy*. These properties have been or will be described in more detail within other sections. Additionally, the *Platform* schema contains any number of *carries* property values which either links to (using *xlink:href*) or define a *_Sensor* or *AttachedPlatform* instance. As will be discussed below, the *locatedUsing* property for determining the location and orientation of the sensor, perhaps as a function of time.

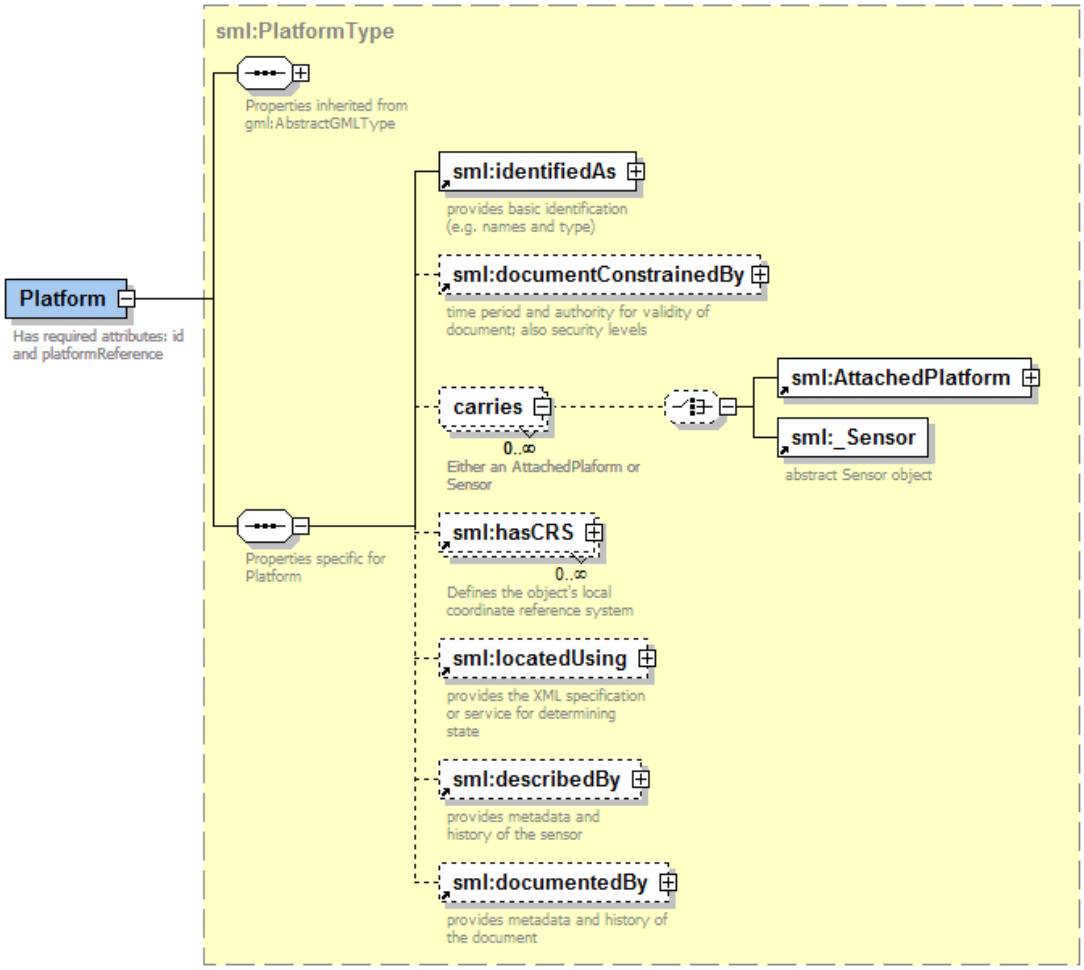


Figure 2.7. Schema for Platform.

Specific platforms can be defined through extension of *PlatformType*. It is intended that a given type of *Platform* may expect certain choices of values for the *locatedUsing* property. Currently, there are simple definitions for aircraft, satellite, land vehicle, water vehicle, and stationary platforms. In addition, there is a special case *AttachedPlatform* that will typically derive its geospatial positioning from its positional relationship with a parent *Platform*. A sensor mount would be one example of an *AttachedPlatform*.

2.2.5. Coordinate Reference Systems

In order for sensor observations to be useful, they generally need to be referenced to some useful coordinate reference system (CRS).

In some cases, such as for 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.

Issue Name: [**minor changes to gml:coordinate.xsd expected.** (meb, 2002-12-12)]

Issue Description:

Minor changes are expected in the GML 3.0 coordinates reference system and coordinate operations schema. These will be reflected in SensorML as soon as the changes are released.

Resolution:

Issue Name: [**Harmonization with GML** (meb, 2002-12-12)]

Issue Description:

NOTE: We plan to improve harmonisation of SensorML with GML for components discussed in this clause.

In future versions of SensorML it is likely that some elements that are in the SensorML namespace ("sml:element") will be moved to or substituted by GML components which offer the same functionality in a more standardised way.

Resolution:

SensorML utilizes definitions for coordinate systems, coordinate reference systems, and coordinate operations provided within 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.

Within the SensorML schema, the objects *Platform*, *Sensor*, and *Sample* (to be discussed below) all include the *hasCRS* property. As shown in Figure 2.8, this optional property either links to (through *xlink:href*) or defines an *EngineeringCRS* as its value. This and provides for the definition of multiple relevant coordinate reference systems within the Sensor instance. Every CRS has an optional *gml:id* attribute that provides a unique

identifier for this CRS. Any CRS with a unique identifier can be referenced or defined within *State* or *_CoordinateOperation* descriptions (to be described below).

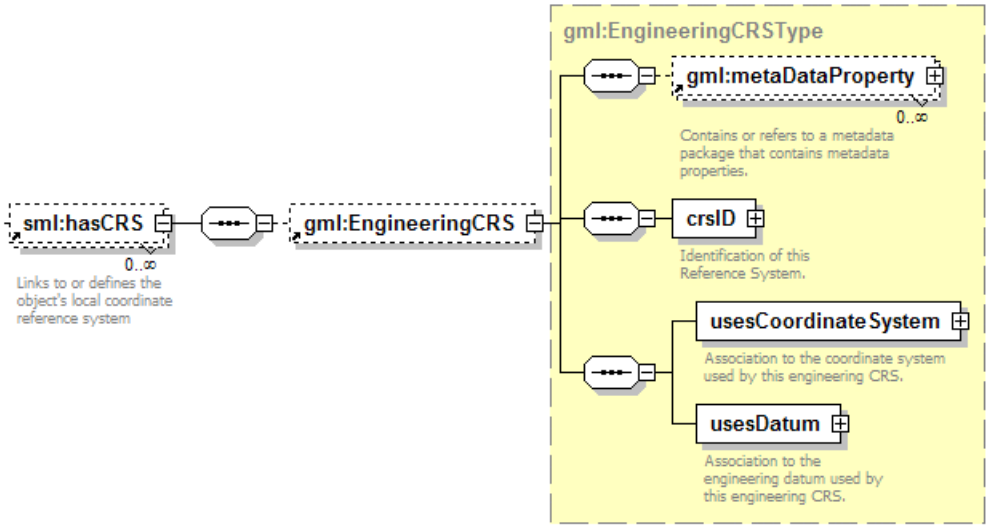


Figure 2.8. Schema for *hasCRS* property used within Platform, Sensor, and Sample.

2.2.6. Sensor and Platform State

The location of a *Platform* or a *Sensor* is provided by the *locatedUsing* property. As shown in Figure 2.9, the *locatedUsing* property either defines or links to (via xlink:href) a *sml:ObjectState*, *gml:_CoordinateOperation*, or *sml:ObjectStateProvider* value. Each of these types will be discussed individually below.

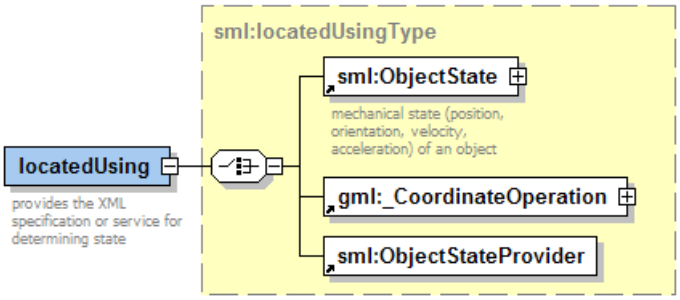


Figure 2.9. Schema for specifying *Platform* and *Sensor* state using either an *sml:ObjectState* or *gml:_CoordinateOperation*.

Issue Name: [**Harmonization with GML** (meb, 2002-12-12)]

Issue Description:

NOTE: We plan to improve harmonisation of SensorML with GML for components discussed in this clause.

In future versions of SensorML it is likely that some elements that are in the SensorML namespace ("sml:element") will be moved to or substituted by GML components which offer the same functionality in a more standardised way.

Resolution:

Issue Name: [**minor changes to gml:coordinate.xsd expected.** (meb, 2002-12-12)]

Issue Description:

Minor changes are expected in the GML 3.0 coordinates schema. These will be reflected in SensorML as soon as the changes are released.

Resolution:

ObjectState. In order to fully specify the state of a sensor or platform, one should specify not only its location, but also its orientation, velocity, acceleration, and rotational acceleration. The state of an object can be specified relative to any coordinate reference system. For a static, in-situ sensor, its state may be simply be defined as a location in a geospatial reference system, such as latitude-longitude-altitude, with orientation being unimportant. This can easily be expressed within the SensorML description using perhaps a *gml:pos* value.

```
<locatedUsing>
    <ObjectState>
        <location>
            <gml:Point>
                <gml:pos srsName="http://www.opengis.net/crs#epsg4326"  dimension="2">
                    35.4 -85.8
                </gml:pos>
            </gml:Point>
        </location>
    </ObjectState>
</locatedUsing>
```

In contrast, the state of a dynamic, remote sensor may be expressed relative to the local coordinate system of its platform or mount, or to a geographic coordinate reference system. Both of these cases can be expressed within SensorML as a *sml:ObjectState* (Figure 2.10). Typically, the sensor state can be expressed as a static (i.e. non-changing) state relative to its platform. However, the platform's state relative to a geographic coordinate system may be constantly changing with time. While both can be expressed within an XML document using the *locatedUsing* schema, one may wish to instead reference a web service for obtaining the platform state for a given time. Complete state

might be obtained through a single state provider or by obtaining location and orientation from separate providers.

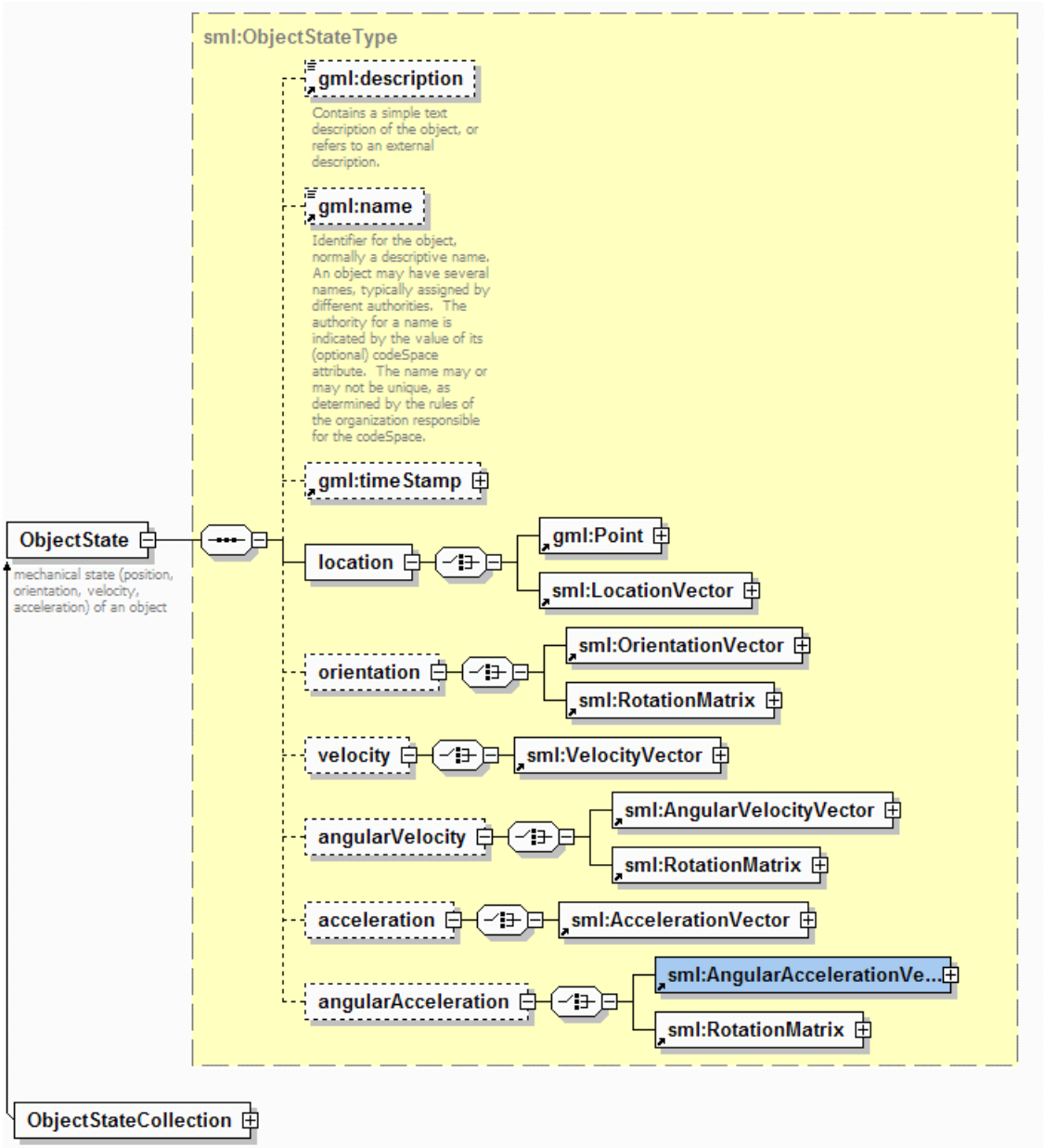


Figure 2.10. The *ObjectState* schema for describing an object's *location*, *orientation*, *velocity*, *angularVelocity*, *acceleration*, and *angularAcceleration* as a function of time.

CoordinateOperation. Another way to specify the state of a platform, sensor, or sample, is by describing one or more coordinate operations (e.g. coordinate transforms) using an object derived from *gml:_CoordinateOperation* (Figure 2.11). 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.

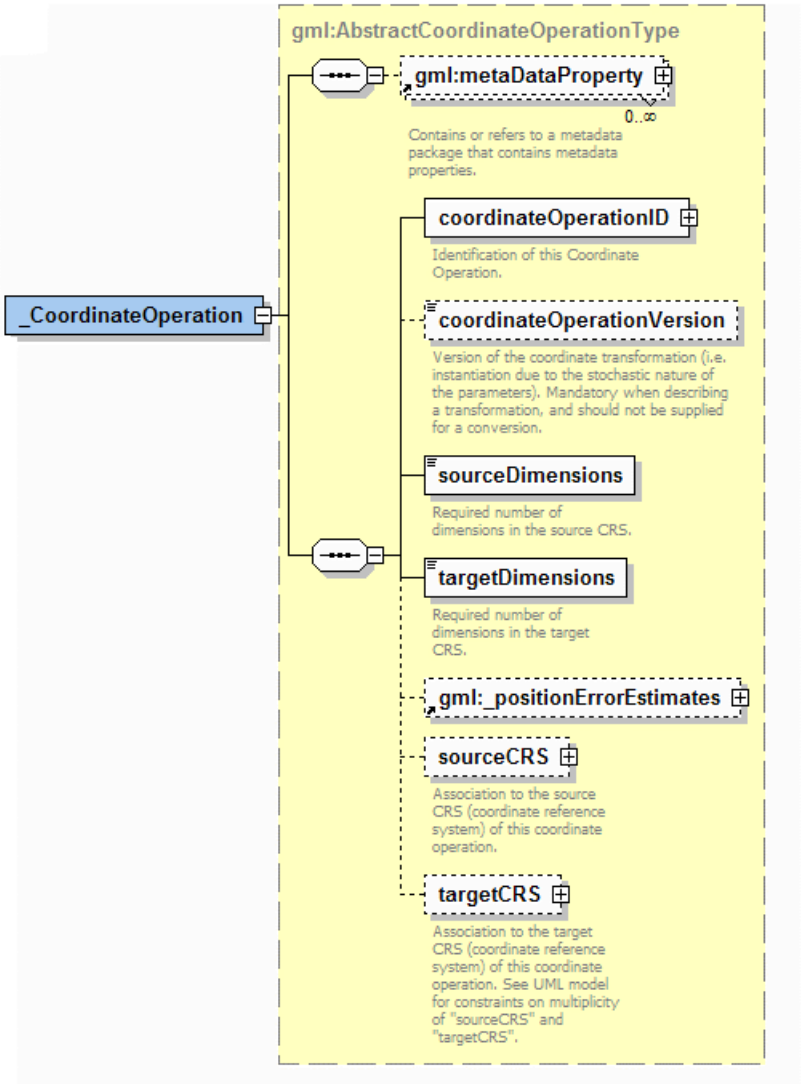


Figure 2.11. The _*CoordinateOperation* schema as defined in GML3 operations.xsd. [*note: sourceDimensions and targetDimensions have been removed within newer releases of the gml schema. That change will be reflected here as the gml schema is solidified]*.**

There are no currently no pre-defined guidelines as to when one should use an *ObjectState* versus a _*CoordinateOperation*. This is purely at the discretion of the sensor

document author or State Provider Service (described below). There are times, particularly for dynamic objects, that providing this information as a transform is more compact and flexible, but places greater burden on the software that makes use of this information. The mixing of state and coordinate operation descriptions is allowed.

Within SensorML, coordinate operations are specified using concrete implementations of the *_CoordinateOperation* property defined in the GML3.0 operations.xsd schema. Typically a *_CoordinateOperation* specifies source and target CRS and the method(s) and parameters used by the operation. Full description of this specification is beyond this initial document, but examples of its use will be shown in later sections.

SensorML defines some coordinate transformation and operations that are helpful for dynamic and static sensors. These currently include OrbitalElementsPropagation, SensorMount, DynamicAffineTransformation, and EllipsoidIntersection. Other useful coordinate operations have been and will be defined by other application schemas.

2.2.7. Sensor and Platform State Provider Services

If the platform or sensor state is obtained from a web service (**ObjectStateProvider**), the requests should follow conventions of the OGC web service framework (***reference***). The request for obtaining object state is the **GetState** request. An example http request for object state might be:

> http://www.someserver.com/server?service=stateProvider&
> request=GetState&version=0.1
> &objectType=platform&objectName=dmspF10&
> time=1999-10-22T12:30:00.0/1999-10-22T16:34:00.0P2min

Issue Name: [**Possible merger of services** (meb, 2002-12-12)]

Issue Description:

It is possible that these services might be consolidated or brought in line with other services; in particular, it should be discussed whether the GetCoordinateOperation should be considered as part of the Web Coordinate Transform Services specification.

Resolution:

Formally, the request has the parameters specified in Table 2.1. The **time** parameter can be a range as shown in the previous example or an instance in time:

> time=2002-10-22T12:30:00.000

The return value type (i.e. format) for the **GetState** request is currently expected to be an XML instance of the sml:ObjectState type.

Request Parameter	Required/ Optional	Description
version	O	Request version number
service=StateProvider	R	Service type
request=GetState	R	Request name
objectType	O	Type for the Platform or Sensor (platform, sensor, or sample)
objectName	R	Identifier of the platform, sensor, or sample
time	O	ISO conformant DateTime event or range
format=sml:objectState	O	Format for returned state (currently only SensorML ObjectState supported

Table 2.1: The parameters of a GetState request URL

An additional request type for the **ObjectStateProvider** is the **GetCoordinateOperation** request. As the name implies, this request returns a concrete implementation of a _*CoordinateOperation*, complete with appropriate parameters. The parameters are listed in Table 2.2. An example might be:

http://www.someserver.com/server?service=stateProvider&
request=GetCoordinateOperation&version=0.1
&objectType=platform&objectName=dmspF10&
transformType=OrbitalElementsPropagator&
time=2002-10-22T00:00:00.0/2002-10-26T00:00:00.0

Request Parameter	Required/ Optional	Description
version	O	Request version
service=StateProvider	R	Service type
request=GetCoordinateOperation	R	Request name
objectType	O	Type for the Platform or Sensor (platform, sensor, or sample)
objectName	R	Identifier of the platform, sensor, or sample
transformType	R	Transform identifier (associated with a Transform schema)
time	O	ISO conformant DateTime event or range
format=xml	O	Format for returned state (currently only SensorML ObjectState supported

Table 2.2: The parameters of a GetCoordinateOperation request URL

2.2.8. Sensor Measurement Characteristics (Measurand)

A sensor measures an observable such as temperature, radiation, animal population, chemical concentration, or others. These observable type entries are expected to be defined within an observable taxonomy dictionary. A sensor has certain sensitivity characteristics with regard to this observable and to the sensor's surrounding environment. Furthermore, a sensor's measurement is generally based on a sample of the environment. This sample might, for example, be a discrete physical entity (e.g. a rock specimen), a volume of some medium (e.g. a volume of air taken in by an in-situ "sniffer"), or an area of a sensor surface that receives radiation.

These measurement characteristics are described by the *measures* property which takes a *Measurand* as its value (Figure 2.12). The *Measurand*, in essence, provides most of the information required to process, analyze, and geolocate observations from the sensor.

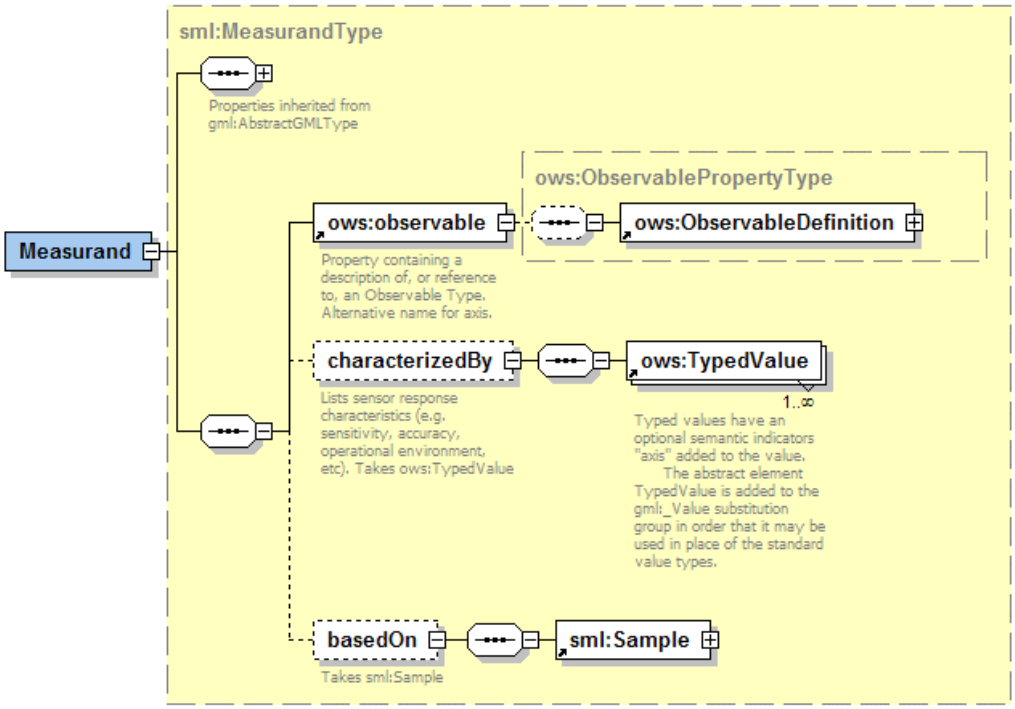


Figure 2.12. Schema diagram for sensor Measurand property. The *Measurand*, in essence, provides most of the information required to process, analyze, and geolocate observations from the sensor.

The *observable* property either links to (through xlink:href) or defines an *ows:ObservableDefinition*. An observable might link to an observables dictionary:

```
<ows:observable xlink:href="http://www.opengis.net/observables#windSpeed"/>
```

or internally be defined using, for instance, a ParameterisedObservable:

```
<ows:observable>
    <ows:ParameterisedObservable>
        <ows:baseObservable xlink:href="http://www.opengis.net/observables#radiance"/>
        <ows:constraints>
            <ows:TypedQuantity uom="http://www.opengis.net/units#GHz"
                axis="http://www.opengis.net/sensorGlossary#peakWavelength"> 19
            </ows:TypedQuantity>
        </ows:constraints>
    </ows:ParameterisedObservable>
</ows:observable>
```

The *characterizedBy* property of the Measurand takes any number of values of type equal to *ows:TypedValue*. These describe the response characteristics of the sensor, as well as its operational conditions. A collection of general response properties are provided in the generalResponse.xsd schema of SensorML. For example, the *characterizedBy* property for a wind speed sensor might include:

```
<characterizedBy>
    <dynamicRange axis="http://www.opengis.net/observables#windSpeed">
        <low uom="http://www.opengis.net/units#mph">0.0</low>
        <high uom="http://www.opengis.net/units#mph">134.0</high>
    </dynamicRange>
    <threshold uom="#mph"
            axis="http://www.opengis.net/observables#windSpeed">2.2</threshold>
    <operationalRange axis="http://www.opengis.net/observables#temperature">
        <low uom="http://www.opengis.net/units#celsius">-40.0</low>
        <high uom="http://www.opengis.net/units#celsius">40.0</high>
    </operationalRange>
    <survivableRange axis="http://www.opengis.net/observables#windSpeed">
        <low uom="http://www.opengis.net/units#mph">0.0</low>
        <high uom="http://www.opengis.net/units#mph">220.0</high>
    </survivableRange>
</characterizedBy>
```

The limited number of general characteristics that will be available in the *GeneralResponse* model may be sufficient to support a large number of simple sensors. Other general characteristics will almost certainly need to be added. 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.

One such collection of specialized response characteristics is defined in the radiationResponse.xsd schema. Newly defined properties include

peakWavelength	*(ows:TypedQuantity)*
bandwidth	*(ows:TypedQuantity)*
spectralRange	*(ows:TypedQuantityExtent)*
polarizationAngle	*(ows:TypedQuantity)*
polarizationDirection	*(ows:TypedCategory)*
spectralResponse	*(ows:TypedQuantityList)*

These parameters should be sufficient for most radiation-based remote sensors. Others may be added in future schema versions.

2.2.9. Samples and the Geolocation of Observations

As discussed above, sensors base their measurements on an individual sample or on a temporal/spatial collection of samples. The *Measurand* property *basedOn* takes a *Sample* or *SampleCollection* as its value. These objects provide information required to determine the dimensions and location of the individual sample or sample collection. Current definitions of *SampleCollection* include *Image*, *ImageSection*, and *ScanLine*. Others will be added as needed.

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.

There are two important concepts to consider regarding sensor samples. First, they can potentially have both spatial and temporal properties that are of interest. Temporal properties might include not only the time of sampling, but also the duration of sampling. 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).

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

For example, scan dynamics might be expressed relative to a scan start time or a sensor clock which has been calibrated to geodetic time. A pixel's geometry might be described as two angular dimensions defined relative to the sensor's local coordinate frame; likewise, the look direction for that given pixel might also be defined within the sensor's local coordinate system. It is then through knowing the relationship of the sensor to the platform and the state of the platform relative to the target (e.g. Earth's ellipsoid) that one determines the conic volume of atmosphere observed or the location of the sample's footprint on the Earth's surface.

As shown in Figure 2.13, *Sample* and its derivatives include either a definition or reference to its CRS (*hasCRS*), a description of its dimensions (*hasSizeOf*), and information for locating the single sample or individuals samples within its collection (*locatedUsing*). As with the location of the platform and sensor discussed previously, the *locatedUsing* property can take any number of *ObjectState* or *gml:_CoordinateOperation* descriptions. In the case of a stationary, in-situ sensor, this may simply be a location within a geographic coordinate reference system, as in:

```
<basedOn>
   <Sample>
      <locatedUsing>
         <ObjectState>
            <location>
               <gml:Point>
                  <gml:pos srsName="http://www.opengis.net/crs#epsg4326"
                           dimension="2"> 35.4 -85.8 </gml:pos>
               </gml:Point>
            </location>
         </ObjectState>
      </locatedUsing>
   </Sample>
</basedOn>
```

It is also possible that the *gml:pos* value could be described relative to the platform's CRS.

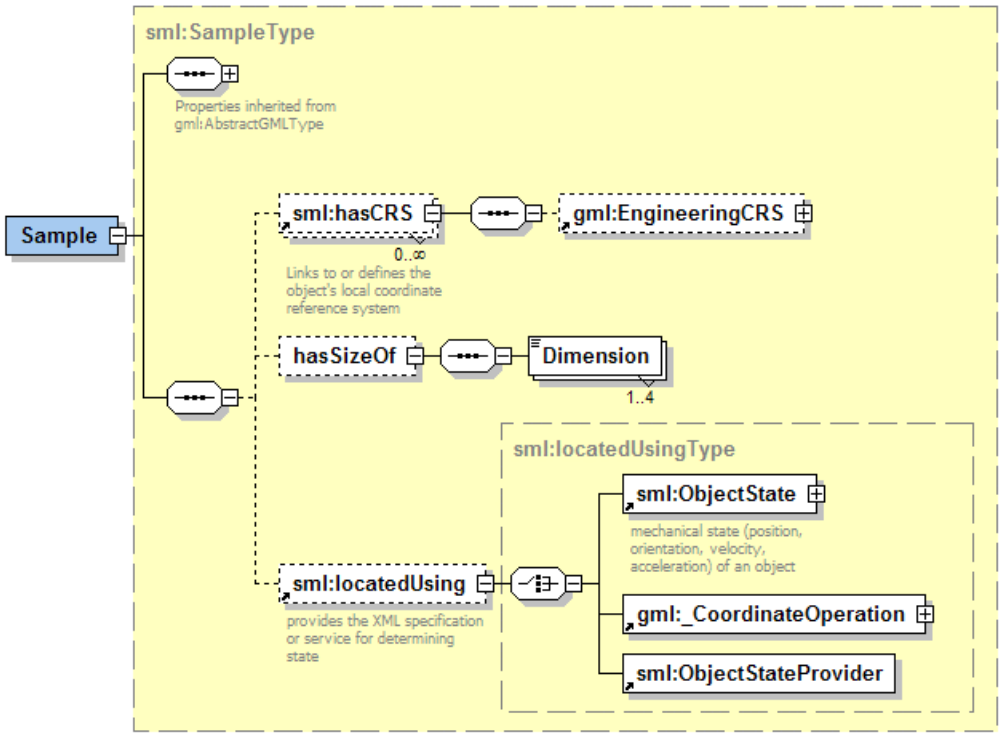


Figure 2.13. Schema diagram for the *Sample* property. Sample provides all information needed to determine the dimensions and geolocation of observations from the sensor.

More complex sensors and sensor arrays, particularly dynamic, remote sensors, may require somewhat more complex descriptions of their location. As an example, for a remote camera on a satellite platform, the *locatedUsing* property might include a collection of *ObjectState* or *gml_CoordinateOperation* descriptions that would define:

1. An *sml:SensorModel* that defines the relationship between an individual sample (i.e. pixel) CRS and the sample collection (i.e. image) CRS
2. An *sml:ObjectState* (e.g. mounting position and rotation) that relates the sample collection CRS to platform CRS
3. An *sml:ObjectState*, *gml:_CoordinateOperation*, or a *sml:ObjectStateProvider* service that relates the platform CRS to a geographic CRS (e.g. Earth-Centered-Inertial CRS) as a function of time.
4. A *gml:Transformation* that transforms the platform CRS from ECI CRS to Earth-Centered-Earth-Fixed (ECEF) CRS.
5. An *sml:IntersectionOperation* that relates the intersection of the pixel's look ray (now in ECEF-CRS) with an Earth ellipsoid CRS (also in ECEF-CRS)
6. Finally, a *gml:Transformation* that transforms intersection coordinates from ECEF-CRS to a latitude-longitude-altitude based CRS.

It should be noted that each of these transform and operation descriptions within SensorML do not directly specify the coordinates, but rather they specify the operations and parameters required to derive the transformed coordinates. The actual derivation of the coordinates depends on software components that understand how to calculate these coordinates for specific transformations and operation based on these parameters.

The *SensorModel* used within this example is part of the substitution group, *gml:_CoordinateOperation* and extends *gml:AbstractCoordinateOperationType*. As shown in Figure 2.14, *SensorModel* is in essence identical to the *gml:Transformation* with the exception that the property, *usesParameters* can either link to (using *xlink:href*) or define *_SensorModelParameters* as its value.

The value of *usesMethod* will provide a description of the method needed to perform the sensor model transformations based on the sensor parameters specified in *usesParameters*. The property *useMethod* will be of the type gml:OperationMethodRefType. The value of usesParameters will be specific for each sensor model, but will define a schema for describing the parameters necessary to perform transformations from sample coordinates to some local or external CRS.

This provides an abstract basis for defining any number of "plug-and-play" sensor models for specifying the geometry, dynamics, and location of samples. There are currently three sensor models defined within SensorML, including (1) the Scanner/Profiler Model, (2) the Optical Camera Model, and (3) the Rapid Positioning Coordinates (RPC) Model. These are discussed in detail in Section 2.4 below. More models will certainly be added in future releases, while others may be defined by specific sensor communities.

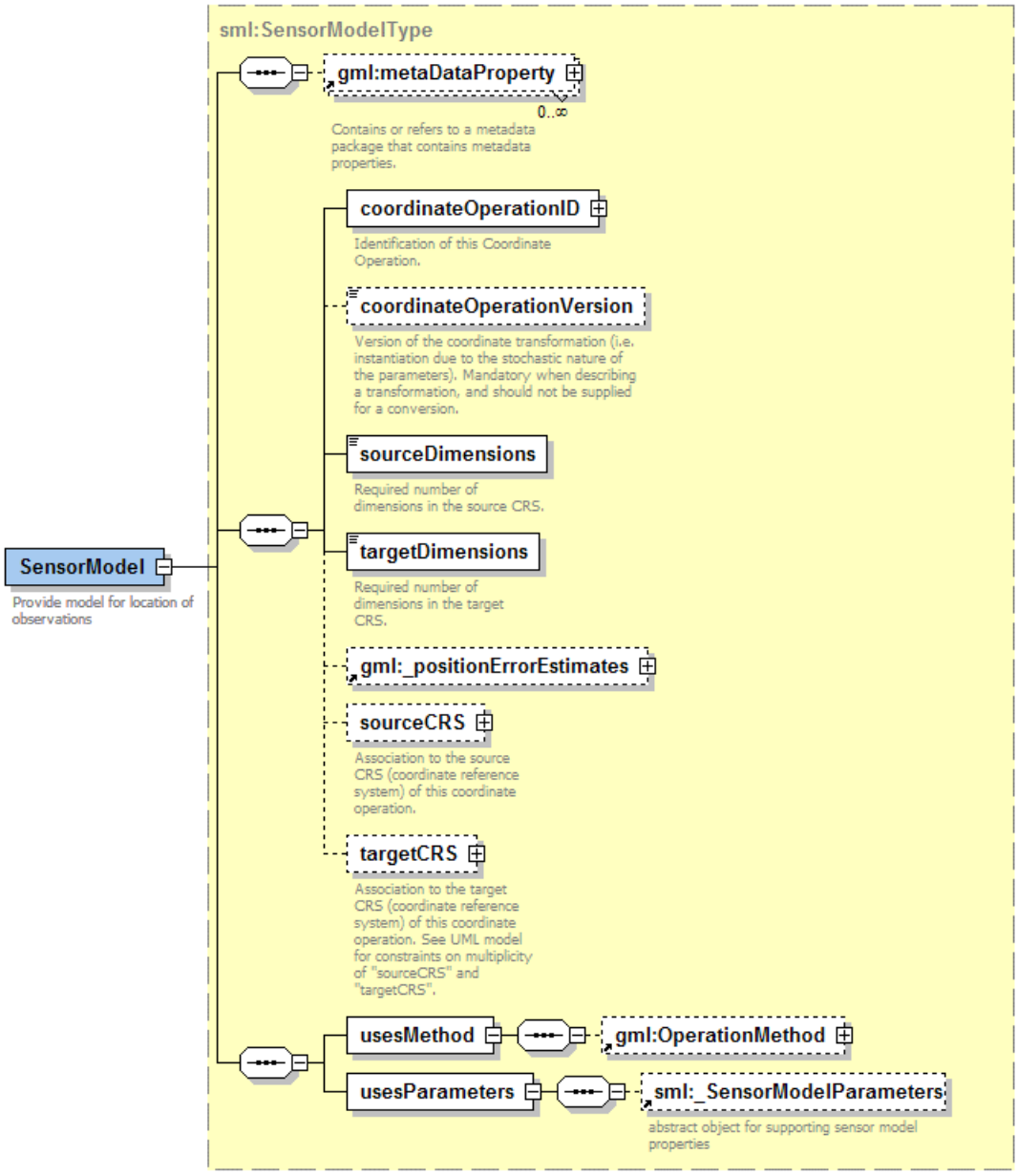


Figure 2.14. Schema diagram for the *OpticalModel*. This is the primary means for relating the CRS of individual samples to either an engineering CRS (e.g. a sample collection CRS) or to a geographic coordinate reference system. Various sensor models will typically differ in the target CRS and particularly the *usesParameters* value.

2.2.10. Sensor Operator and Sensor Planning Services

As shown in Figure 2.15, the *operatedBy* property provides information regarding the party or parties (e.g. company, agency, individual) responsible for operating the sensor and possibly managing its data. The *ResponsibleParty* is of the type *iso19115:CI_ReponsiblePartyType* and defines the role of the party, as well as providing

points of contact if additional information is needed regarding tasking of the sensor or data availability, perhaps. Optionally, an *operatedBy* property can provide a *serviceURI* link (*xs:anyURI*) to a *SensorPlanningService* for this sensor. A Sensor Planning Service (SPS) is an OGC-compliant service for tasking a sensor system. Its description is beyond the scope of this document [SPS].

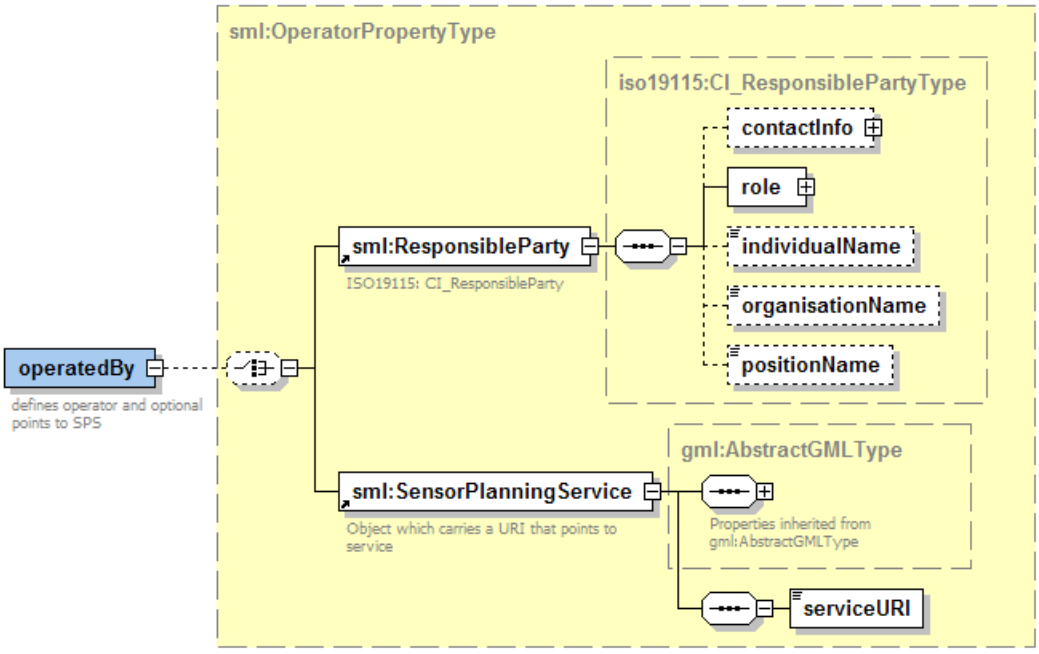


Figure 2.15. Schema for Sensor operator information and links to Sensor Planning Services.

2.2.11. History and Actions

As described earlier, a SensorML document is considered a "living" document that can accompany (or record changes to) a sensor from its design through its creation, deployment, calibration, maintenance, and ultimately its removal from service. Within SensorML, these will be recorded through a collection of *_SensorEvent* values. Similarly, the SensorML document itself undergoes changes, some of which result from actions on the sensor, and others that result from a *_SensorEvent,* document additions, or changes to the sensor model assumptions. All of these changes are recorded within the SensorML through the use of a collection of *_DocumentEvent* descriptions.

Both of the abstract classes *_SensorEvent* and *_DocumentEvent* derive from the base abstract class of *_EventType*. As shown in Figure 2.16, the base class of *_EventType* includes some basic event metadata properties, such as *responsibleParty* (iso19115:CIResponsiblePartyType), *when* (xs:dateTime), *description* (xs:string), and *ReferenceDocuments*.

From the *_SensorEvent*, we derive *SensorCreation*, *SensorDeployment*, *SensorInspection*, *SensorCalibration*, and *SensorDecommission*. From the base

_DocumentEvent, we derive *DocumentCreation*, *DocumentApproval*, and *DocumentModification*. As will be described in more detail below, *_SensorEvent* and *_DocumentEvent* will be employed as values in the *history* properties for *describedBy* and *documentedBy* sections of the SensorML document.

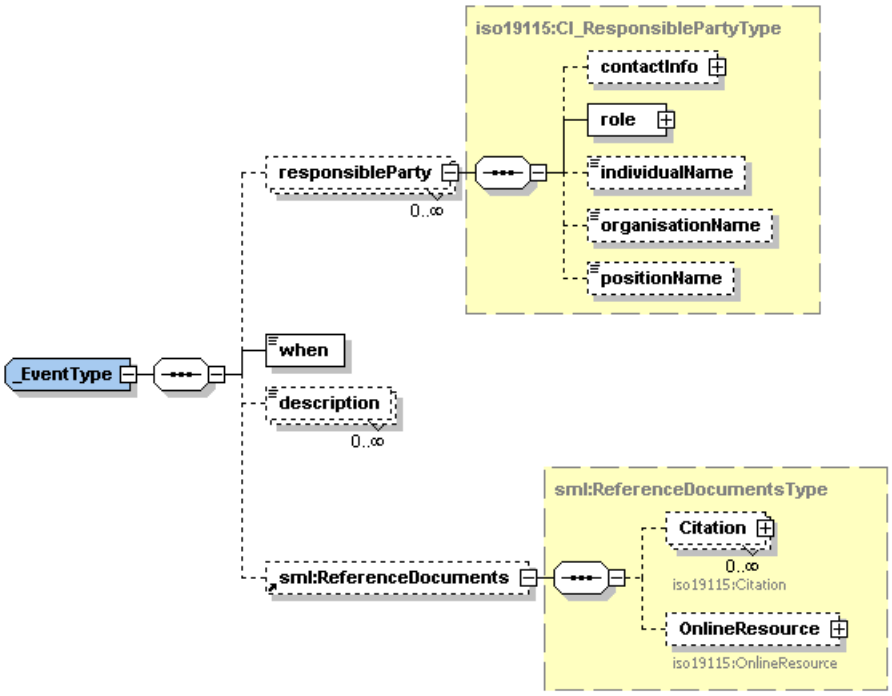


Figure 2.16. Base abstract *_EventType* from which *_SensorEvent* and *_DocumentEvent* are derived.

2.2.12. Sensor Metadata

General metadata regarding the sensor is provided by the *describedBy* property which takes an *AssetDescription* as its argument. As shown in the schema diagram of Figure 2.17, *AssetDescription* makes use of the ISO19115 schema for metadata. Optional values include *Manufacturer* (iso19115: CI_ResponsiblePartyType), *ModelNumber* (xs:string), *IdentificationNumber* (xs:string), *DeployingAgency* (iso19115:CI_ ResponsiblePartyType), *AssetHistory* (*_AssetEvent* described above), and *Description* (xs:string). The *Description* value is provided for textual discussion of various aspects of the sensor. In addition, the *describedBy* property includes a *ReferenceDocument* value which can hold any number of *Citation* (iso19115: Citation) and *OnlineResource* (iso19115:OnlineResource) values to refer to exterior sources for additional information regarding the sensor.

The final optional value within the sensor specifications section is one or more *GenericProperty* entries. *GenericProperty* allows one to include properties that might not

be directed supported by SensorML, but that might be desired by some proprietary software or manufacturer data base. Examples for the YSI water quality sensor include:

```
<GenericProperty name="sensorTechnology" dataType="xs:string"> rapid pulse </GenericProperty>
<GenericProperty name="measurementMethod" dataType="xs:string"> EPA accepted </GenericProperty>
<GenericProperty name="membraneThickness" dataType="xs:double" uom="#mil"> 1.0 </GenericProperty>
<GenericProperty name="membraneType" dataType="xs:string"> Teflon </GenericProperty>
<GenericProperty name="probeSolutionType" dataType="xs:string"> Na2SO4 </GenericProperty>
```

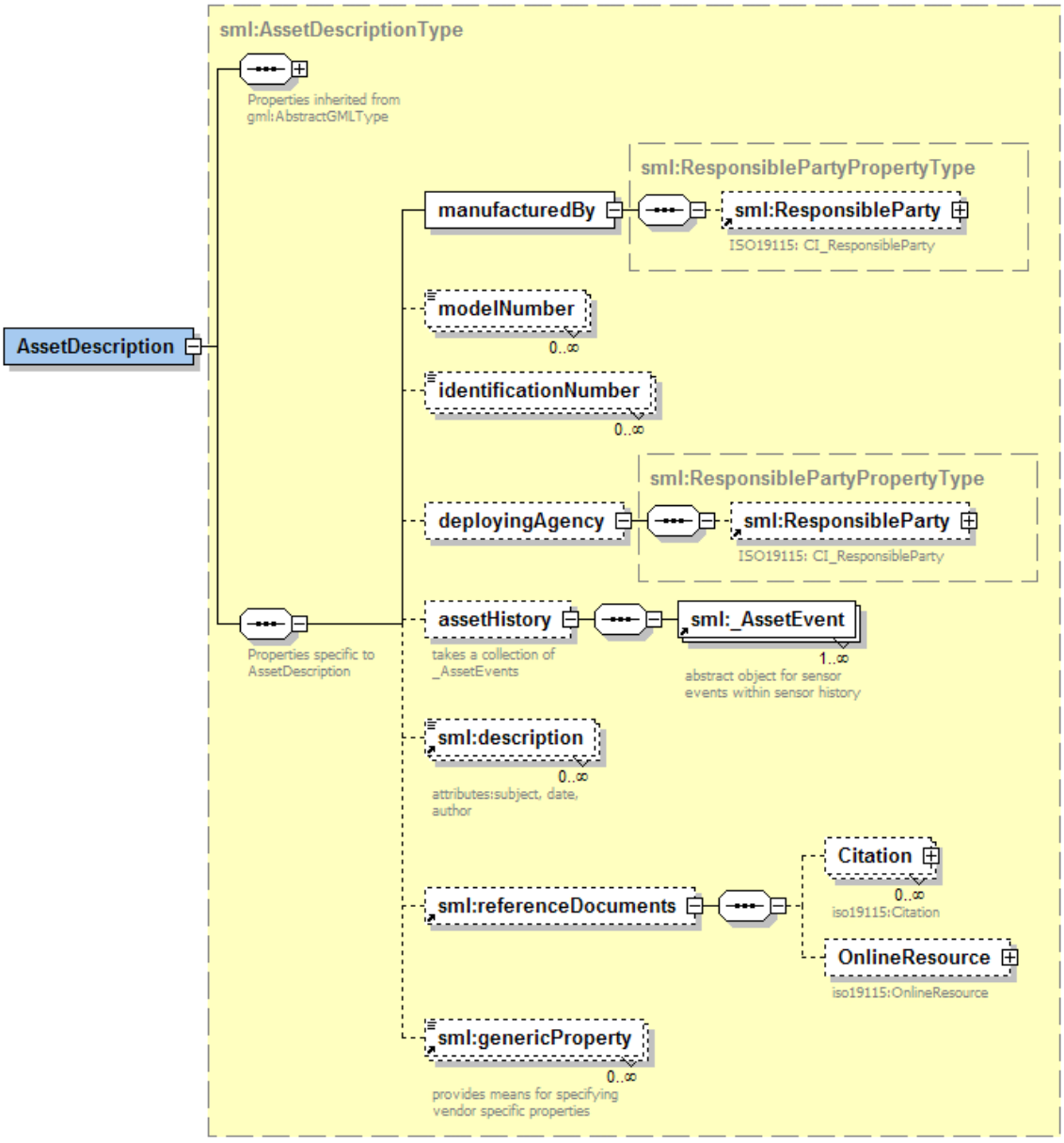


Figure 2.17. Schema for Sensor Metadata.

2.2.13. Document Description and Metadata

As discussed earlier, a SensorML document is both self-describing and a living document. Thus, assumptions regarding the design of the sensor description, as well as the history of the document are recorded in the *documentedBy* property which takes a *DocumentMetadata* as its value. As shown in Figure 2.18, *DocumentMetadata* includes a *documentHistory* (*_DocumentEvent*), zero or more *description* entries (xs:string), and r*eferenceDocuments*.

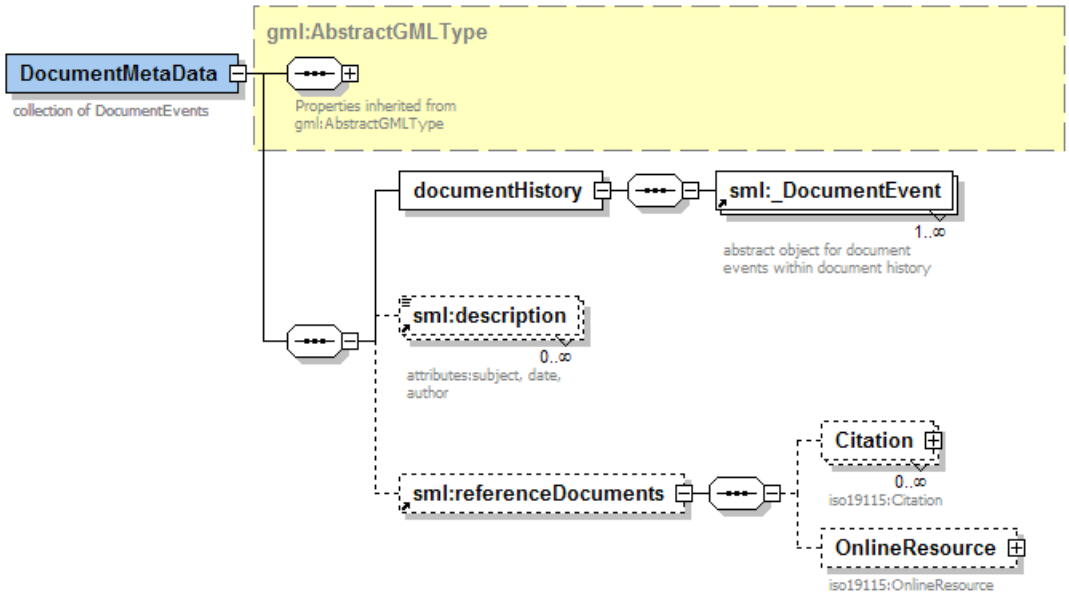


Figure 2.18. Schema diagram for document metadata.

2.3. SPECIFIC RESPONSE MODELS

Issue Name: [**placeholder**. (meb, 2002-08-17)]

Issue Description:

This is a placeholder for describing specific response models as they are developed. These will include perhaps radiation, chemical concentration, biological population, and geological models. Currently, the only specific model is the radiation response model that is adequately described in section 2.2.7.

Resolution:

Issue Name: [**may move to separate document**. (meb, 2002-12-05)]

Issue Description:

Specific response models and specific sensor models may move to a separate document in future documentation.

Resolution:

2.4. SPECIFIC SENSOR MODELS

In section 2.2.9, the *SensorModel* was introduced as means for defining "plug-and-play" components for describing observation geometry, dynamics, and geolocation. Most of the flexibility for defining various sensor models is derived from the *usesMethod* and *usesParameters* properties.

Issue Name: [**may move to separate document**. (meb, 2002-12-05)]

Issue Description:

Specific response models and specific sensor models may move to a separate document in future documentation.

Resolution:

In addition to the simple in-situ sensor case described earlier, there are currently three specific remote sensor models defined within SensorML. These three models will probably suffice for a large proportion of sensor systems. However, it is anticipated that there may be more sensor models defined as the needs arise, particularly for various functional models and NIMA's upcoming Replacement Sensor Model. The three current models include (1) the Scanner/Profiler Model, (2) the Optical Camera Model, and (3) the Rapid Positioning Coordinate, or RPC, Model. These will be discussed in detail below.

All of these models are based on the SensorModel schema pattern shown in Figure 2.14 and repeated in Figure 2.19. The schema is derived from the base *gml:AbstractCoordinateOperationType* and serves as part of the substitution group, *gml:_CoordinateOperation*. The *SensorModel* has an optional *gml:ID* attribute that can uniquely identify the particular instance of the model. In addition, the sensor model includes optional *gml:metaDataProperty*, a required *coordinateOperationID*, and an optional *coordinateOperationVersion*. The *coordinateOperationID* would be common for all instances of the same sensor model type and would thus be fixed for each model. The required *sourceDimensions* and *targetDimensions* properties describe the number of dimensions in the source and target CRSes. The *gml:_positionErrorEstimates* property provides a means for describing the estimated error expected from the model.

The *sourceCRS* and *targetCRS* either define or reference through a URI the coordinate reference systems involved in the transformation. While the *sourceCRS* is typically the CRS of an individual sample, the *targetCRS* could be a sample collection CRS, the sensor CRS, the platform CRS, or a geographic CRS. For example, typically, a *scannerModel* might relate the individual sample CRS to a local engineering CRS, while a *rpcModel* tends to support coordinate transforms directly from a sample CRS to a geographic CRS.

However, the main difference in particular sensor model types will be in the method defined in *usesMethod* and in the model parameter schema provided for *usesParameters*. Most of the following discussions will focus on these two properties.

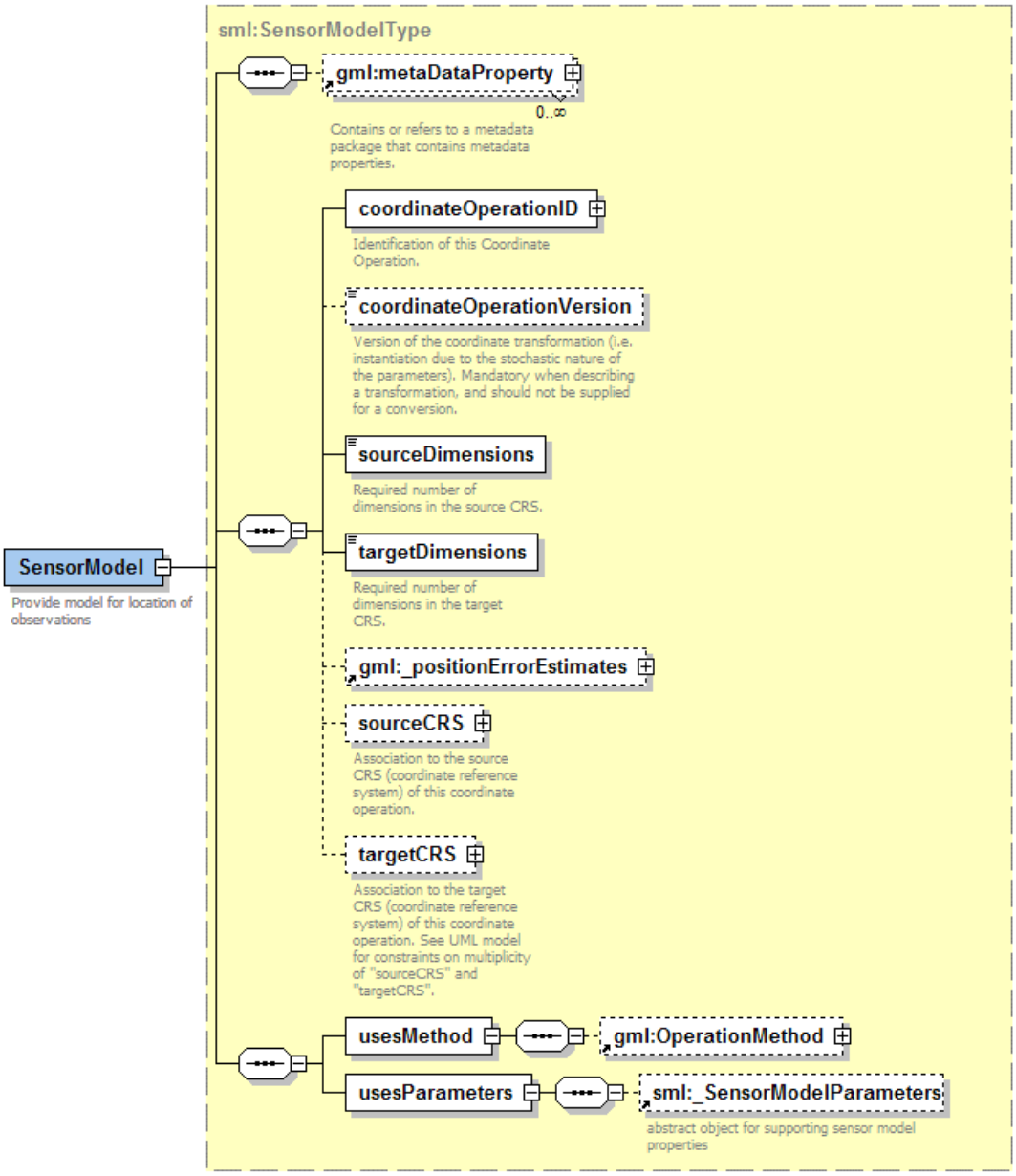


Figure 2.19. The general schema for *SensorModel* used in SensorML. The schema is derived from the *gml:AbstractCoordinateOperationType* and is part of the *_CoordinateOperation* substitution group.

2.4.1. Scanner / Profiler Model

The sensor model for scanners and profilers is designed to support a wide range of remote sensors. This would include sensors such as a simple line-of-sight range detector, a static or dynamic 1D profiler, 2D linear or conical scanners (either stationary or attached to dynamic platforms), and a static or dynamic 3D scanner/profiler (e.g. a

Doppler weather radar). Within SensorML is a general definition for *ScannerProperties* which can be used "as-is" or as a basis for defining very specific scanner/profiler models. Additionally within SensorML, specific scanner/profiler models have also been derived from *ScannerProperties*, including *FixedView*, *Profiler1D*, *Profiler2D*, *Profiler3D*, *Scanner1D*, and *Scanner2D* scanner types. These should be sufficient for supporting most scanners and profilers, although it is possible to define others. The existing scanner types will be discussed in Section 2.4.1.2, following the description of the general scanner model.

2.4.1.1. General Scanner/Profiler Model.

Scanning Coordinates. Most rigorous sensor models for scanners and profilers can best be defined within a spherical coordinate reference system. Within SensorML, a scanner/profiler can sample within three directions, referred to within the spherical CRS as sweep, elevation, and profile directions. For visualization purposes, these directions can be viewed as spherical coordinates within the sensor's right-handed coordinate frame, with the sensor's radiation detection occurring at the origin. As shown in Figure 2.20, the sweep angle always specifies rotation about the polar axis, Z, elevation measures the angle from the sensor's equatorial plane, and profile can be considered as a radial

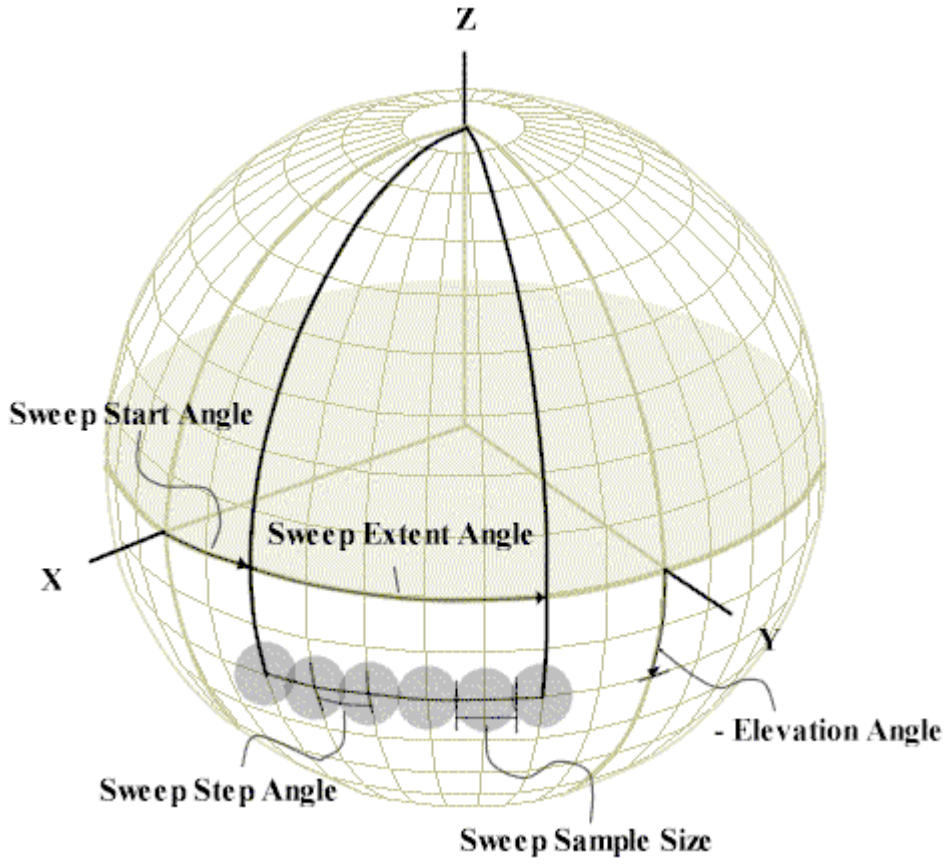


Figure 2.20. Spherical coordinate reference system used for the scanner/profiler sensor model.

dimension along the line-of-sight. In the proposed SensorML, Z is always expected to designate the polar axis. Sweep and elevation angles can be specified as either positive or negative values, but they must follow the right-handed coordinate rule.

By changing the orientation of this spherical CRS, one can support various scanning patterns. For example, using this general model, a conical scanner (scanner in which the scan line on the earth approximates an conic arc) can be considered as a sensor in which the polar axis, Z, is roughly parallel to the main direction of the target, while the polar axis of a line scanner is roughly perpendicular to the main target direction (Figure 2.21). Thus, a line scanner and conic scanner are mathematically equivalent, and are distinguished only by the general direction of the polar axis relative to the target.

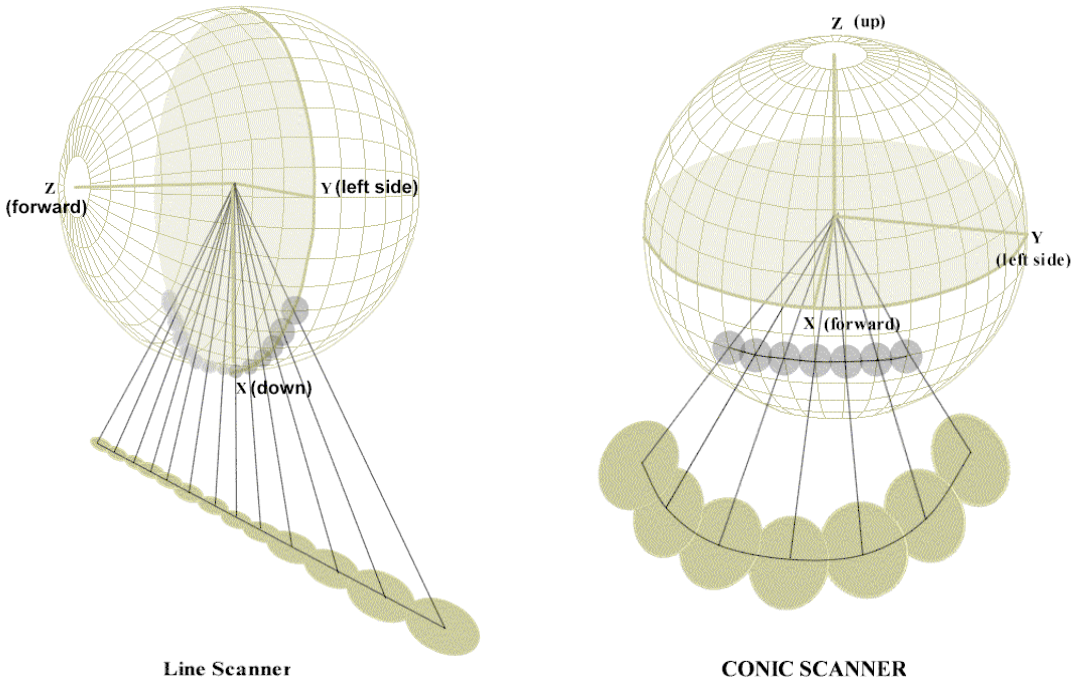


Figure 2.21. Schematics comparing the line scanner orientation to that of a conic scanner. The orientation of a line scanner has the polar axis, Z, roughly tangential to the target surface and roughly in the velocity direction. In contrast, a conic scanner is oriented with the polar axis, Z, roughly perpendicular to the target surface, thereby creating a conic pattern of pixels.

Scanner Properties. The *scannerProperties* provides the primary information required to determine look ray directions (in sensor coordinate space) for any given pixel and to determine which pixel is being sampled at any given time. As shown in Figure 2.22, the specification is based on describing one or more *_scanAction* values within a *do* property. The collection of *_scanAction* values can be sequentially listed indicating a temporal sequence of scan actions, of can be nested within one another. As in computer programming using "Do" or "For" loops, the nesting of scan actions within the SensorML specifies the iteration order in which these actions occur. This allows virtually any scanner or profiler to be described using a single sensor model.

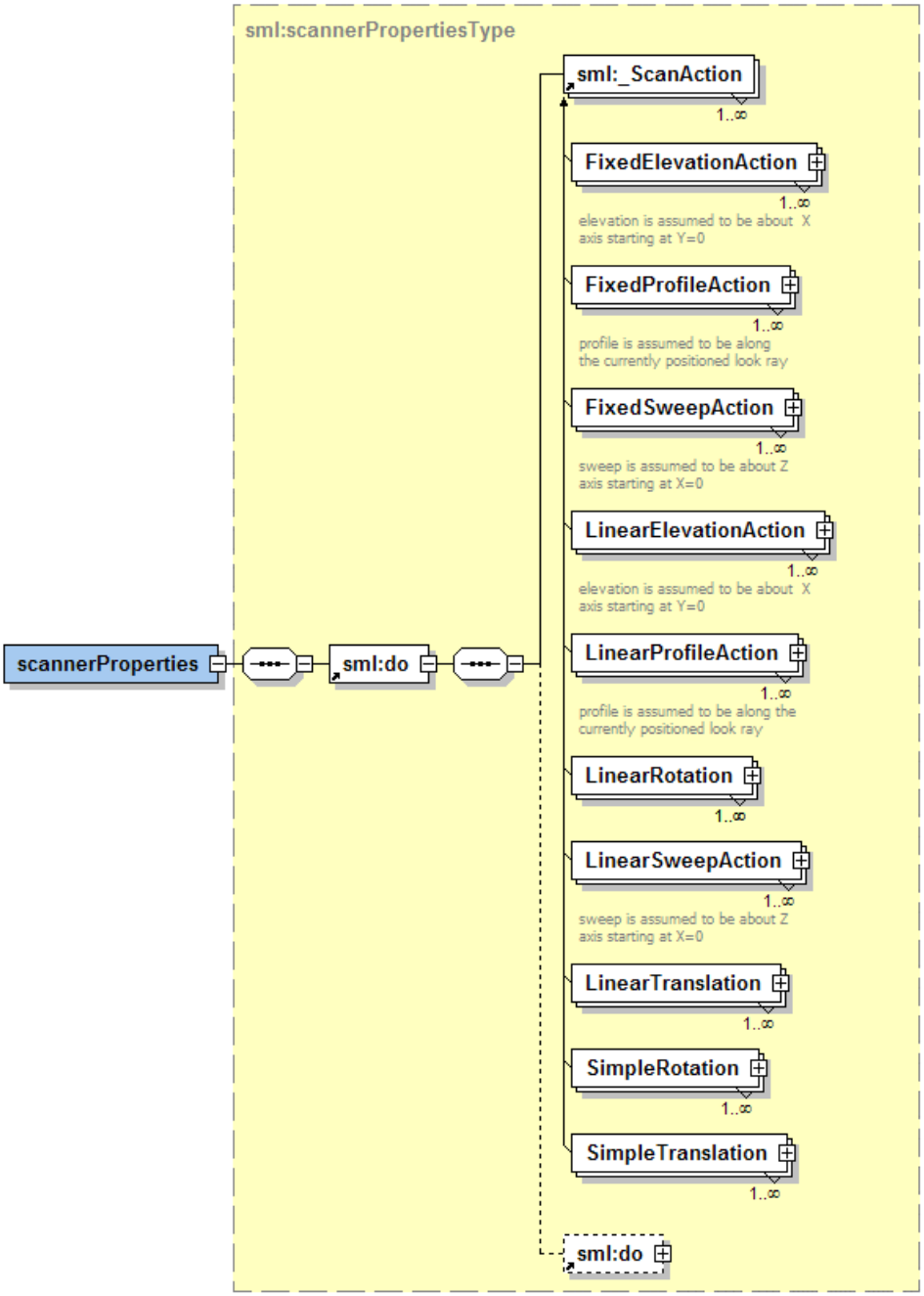


Figure 2.22. The schema diagram for the base-level scannerProperties. The *sml*:do property takes a *_ScanAction* as its argument and can be nested to signify the order of scan actions. Several scan action types have been defined within SensorML.

Currently, SensorML supports base type definitions for simple (i.e. fixed) rotations and translations and for linear (i.e. constant step) translations and rotations. These schema are illustrated in Figure 2.22 as being of the type *_ScanAction* and thus able to be substituted within the *do* property of *scannerProperties*. It is anticipated that future releases will

include support for non-linear rotations and translations. While non-linear scanning could be supported in the current release by explicitly listing a collection of angles or distances using *SimpleRotation* and *SimpleTranslation*, respectively, these might be more compactly defined using definitions of polynomial functions.

The four basic *_ScanAction* types currently supported include *SimpleTranslation*, *SimpleRotation*, *LinearTranslation*, and *LinearRotation*. As shown in Figure 2.23, the *SimpleTranslation* and *SimpleRotation* scan actions take single fixed values for distance and angle, respectively. The *fixedDistance* property is of type *DistanceValueType* which extends *xs:double* and has required attributes of *distanceUnit* (*xs:anyURI*) and basis (*xs:string*). The basis attribute is an enumeration with the current accepted values of "sampleCenter", "sampleOuterEdge", or "sampleInnerEdge". Similarly, *fixedAngle* is of type *AngleValueType* with the attributes of *angleUnit* and *basis*.

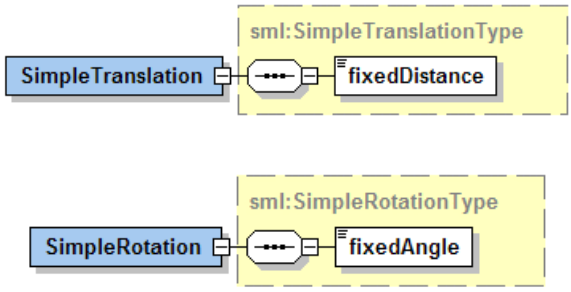


Figure 2.23. *SimpleTranslation* and *SimpleRotation* scan actions.

The scan action, *LinearRotation*, is shown in Figure 2.24, and supports a step-wise rotation with fixed step angle. The *numberOfSamples* property specifies the number of sample observations taken during this complete scan action. The properties, *startAngle* and *startTime*, indicate the starting position (within the target CRS) and starting delta time (relative to the temporal CRS) for the first pixel sample of the scan action. To specify the extent over which sampling occurs, one may use either the *extentAngle* or the *stopAngle* property. Either of these can be combined with the *numberOfSamples* element to derive the other.

All of the rotation properties are of the type *AngleValueType* and thus include the attributes of *angleUnit* and *basis*. The time properties are of the type, *TimeValueType* with similar attributes for *timeUnit* and *basis*. Although the *stepAngle* could similarly be derived from other properties, it is required here in order to specify the direction of rotation. Thus, a positive step value indicates rotation consistent with the right-hand rule, while a negative step value indicates rotation in the opposite direction.

The *extentTime* specifies the amount of time required to traverse the *extentAngle*, while *stepTime* does the same for the *stepAngle*. Like the *extentAngle* and *stepAngle*, it is only necessary to specify one of the parameters, since the other will be derived using the *numberOfSamples* value. If *extentTime* or *stepTime* are not specified or are set to zero, then the entire sweep is assumed to be sampled instantaneously, as would be the case in the imager or push broom sensor type. The *repeatTime* specifies the amount of time that

passes before the next sweep scan begins; in essence, this determines how often a new sweep event will occur.

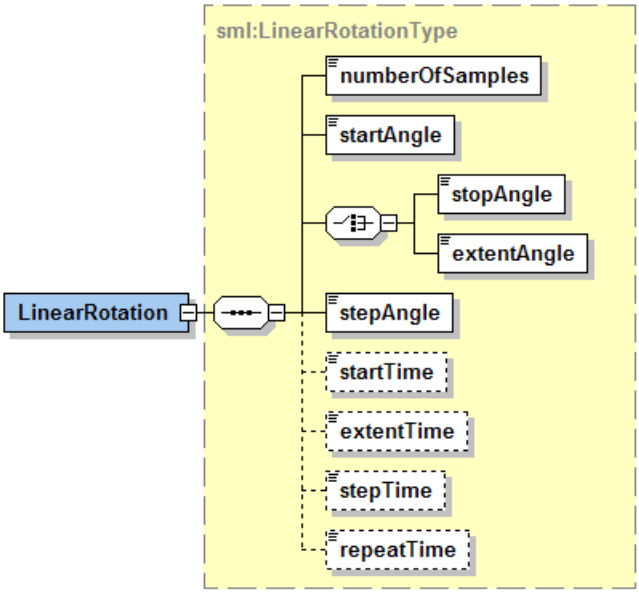


Figure 2.24. *LinearRotation* scan action supporting an iterative stepping model with constant angular steps.

The properties within the *LinearTranslation* define sampling characteristics based on translation (e.g. distance between sensors in a sensor grid) or along the line of sight direction (i.e. a profile measurement). As illustrated in Figure 2.25, the LinearTranslation scan action has properties similar to the LinearRotation with the exception of the use of distance measures instead of angles. Translation distances are measured as distances relative to the *targetCRS* origin.

It is important to stress that the time properties in these definitions can be relative to a geographic-based gml:*temporalCRS* or to a local gml:*temporalCRS.* As an example, the *startTime* and *stopTime* for a scanner will often be specified relative to a local time for each scan line or each scan volume.

Issue Name: [**temporal and spatial CRS needed in sensor model** (meb, 2002-12-05)]

Issue Description:

The current gml:AbstractCoordinateOperationType from which the sensor models are derived, only accepts a single CRS for target and source CRS properties. Thus, either (1) the time element must also be included as part of the CRS definition, (2) the gml: AbstractCoordinateOperationType needs to allow more than one CRS for target and source CRSes, or (3) we need to derive sensor models from a less restrictive base.

Resolution:

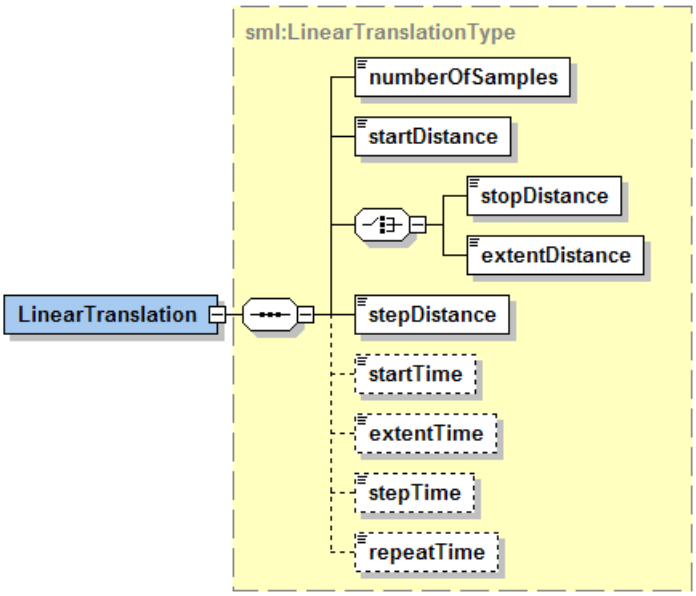


Figure 2.25. *LinearTranslation* scan action supporting an iterative stepping model with constant distance steps.

Spherical Scanner Model. While scanners and profilers could be defined simply using these base translation and rotation types, the scanner/profiler schema derives several elements based on defining angular, spatial, and temporal measurements within the *sweep*, *elevation*, and *profile* coordinates in the spheroid model shown in Figure 2.20. These include:

Base Types	**Spherical CRS specific**
SimpleTranslation	*FixedProfileAction*
LinearTranslation	*LinearProfileAction*
SimpleRotation	*FixedSweepAction*
	FixedElevationAction
LinearRotation	*LinearSweepAction*
	LinearElevationAction

These derived types have been derived from their base class by restriction with their axes attributes fixed according to the convention in Figure 2.20. They primarily serve as convenience elements.

The *sweep* coordinate is an angle measurement about the Z axis, starting at the X axis and with positive direction being from the X axis to the Y axis ("right-handed rule"). The *elevation* coordinate is an angle measurement about the X axis, starting at the Y axis and with positive direction being from the Y axis to the Z axis ("right-handed rule"). The *profile* coordinate is a distance measurement from the center of the sphere and outward

along a radial. It is always positive. It is explicitly noted that these angles and distances are defined relative to the target CRS, which in this case is the scanner's spherical CRS.

Order of Scan Operations. A sequential execution of *_ScanAction*(s) is specified by including any number of *_ScanAction* definitions within a single do property, in order of occurrence.

```
<usesParameters>
    <Scanner2D>
        <do>
            < FixedSweepAction >  <!— first action -->
                …
            </ FixedSweepAction >
            < FixedSweepAction > <!— second action -->
                …
            </ FixedSweepAction >
            < FixedSweepAction > <!— third action -->
                …
            </ FixedSweepAction >
        </do>
    </Scanner2D>
</usesParameters>
```

The hierarchical nesting of *_ScanAction*(s) provides essential information for describing the order, direction, and timing of sampling. Similar to "Do-loops" or "For-loops" within programming languages, SensorML is capable of describing which *_ScanActions* (e.g. *LinearRotation* or *LinearTranslation*) vary the fastest. As in programming loops, sampling steps within the innermost loops will progress first, followed by the *_ScanAction*(s) in the outer loops.

For example, this nesting describes a the scan pattern consisting of one step in the elevation direction, followed by a complete sweep in the positive direction, followed by another elevation step, and so on. This pattern would be repeated until all elevation steps had been completed.

```
<usesParameters>
    <Scanner2D>
        <do>
            < LinearElevationAction >
                …
                <stepAngle angleUnit="#degrees"
                           basis="sampleCenter"> 10.0 </startAngle>
                …
            </ LinearElevationAction >
            <do>
                <LinearSweepAction>
                    …
                    <stepAngle angleUnit="#degrees"
                               basis=" sampleCenter "> 3.0 </startAngle>
                    …
                </LinearSweepAction>
            </do>
    </Scanner2D>
</usesParameters>
```

This example shows instead a case where elevation steps would occur first followed by a step in the horizontal sweep direction. This pattern would be repeated until all sweep steps had been completed.

```
<usesParameters>
    <Scanner2D>
        <do>
            < LinearSweepAction >
                …
                <stepAngle angleUnit="#degrees"
                                basis="sampleCenter"> 10.0 </startAngle>
                …
            </ LinearSweepAction >
            <do>
                < LinearElevationAction >
                    …
                    <stepAngle angleUnit="#degrees"
                                basis=" sampleCenter "> 3.0 </startAngle>
                    …
                </ LinearElevationAction >
            </do>
        </do>
    </Scanner2D>
</usesParameters>
```

Finally, the pattern below illustrates a scan pattern where there would be a elevation step followed by a complete sweep in the positive direction, followed by a complete sweep in the negative direction, followed a repeating of the pattern until all elevation steps are completed.

```
<usesParameters>
    <Scanner2D>
        <do>
            < LinearElevationAction >
                …
                <stepAngle angleUnit="#degrees"
                                basis="sampleCenter"> 10.0 </startAngle>
                …
            </ LinearElevationAction >
            <do>
                <LinearSweepAction>
                    …
                    <stepAngle angleUnit="#degrees"
                                basis=" sampleCenter "> 3.0 </startAngle>
                    …
                </LinearSweepAction>
            </do>
            <do>
                <LinearSweepAction>
                    …
                    <stepAngle angleUnit="#degrees"
                                basis=" sampleCenter "> -3.0 </startAngle>
                    …
                </LinearSweepAction>
```

```
            </do>
        </do>
    </Scanner2D>
</usesParameters>
```

The following example for the SSM/I conical sensor specifies that the elevation angle is first set to a fixed angle of –45 degrees, and then a sweeping pattern from +51 to –51 degrees repeats constantly, sampling for 64 steps. These pixels are sampled over a time of 1.9 seconds and the complete scan repeats every 3.8. If the elevation angle had not been fixed, as in the case of Doppler radar, then each complete sweep would have been followed by an elevation step.

```
<usesParameters>
    <Scanner2D>
        <do>
            <FixedElevationAction>
                <fixedAngle angleUnit="#degrees" basis="sampleCenter"> -45.0 </fixedAngle>
            </FixedElevationAction>
            <do>
                <LinearSweepAction>
                    <numberOfSamples>64</numberOfSamples>
                    <startAngle angleUnit="#degrees"
                                basis="sampleOuterEdge"> 51.0 </startAngle>
                    <extentAngle angleUnit="#degrees"
                                basis="sampleOuterEdge"> -102.4 </extentAngle>
                    <startTime timeUnit="#seconds"
                                basis="sampleOuterEdge"> 0.0 </startTime>
                    <extentTime timeUnit="#seconds"
                                basis="sampleOuterEdge"> 1.9 </extentTime>
                    <repeatTime timeUnit="#seconds"
                                basis="sampleOuterEdge"> 3.8 </repeatTime>
                </LinearSweepAction>
            </do>
        </do>
    </Scanner2D>
</usesParameters>
```

2.4.1.2. Scanner/Profiler types.

A wide variety of scanners and profilers can be supported using the ScannerModel with various derived *scannerProperties*. This section looks as some sensor types and suggest the appropriate *scannerProperties* types to utilize for each.

Line-of-sight sensor. The line-of-sight sensor is the simplest profiler, consisting of essentially a single look direction. It will thus have constant sweep and elevation angles, sampling either a single point or multiple points along this line-of-sight. Examples of line-of-sight sensors include altimeters and range finders, ground-based atmospheric sounders, and non-sweeping LIDAR. For these types of sensors, one could utilize the Profiler1D scanner properties (Figure 2.26).

Conic-scanner. Conic scanners include those sensors that scan by sweeping around a rotation axis producing a cone pattern relative to the target body (see Figure 2.21). Conic scanners can be one to three dimensions, possibly including sweep, elevation, and/or profile components. Examples of conic scanners include SSM/I (1D), which sweeps

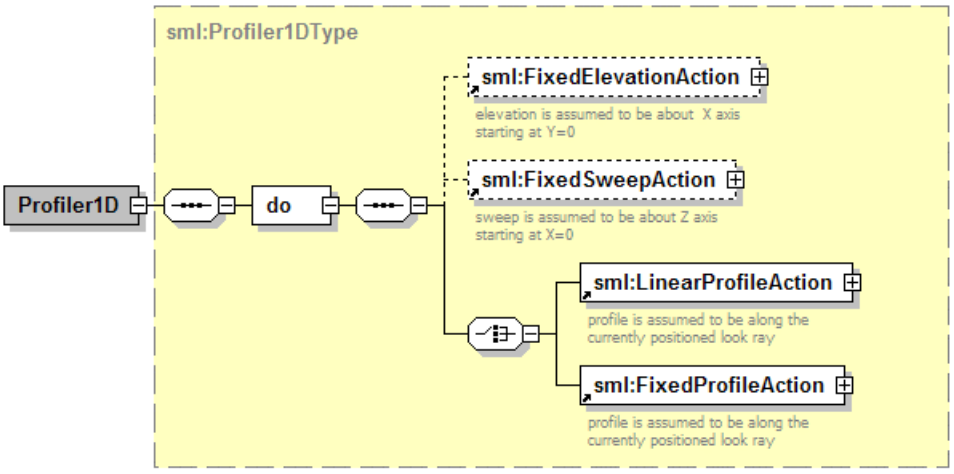


Figure 2.26. ***Profiler1D scannerProperties*.**

around a single axis at a constant elevation angle, and ground-based Doppler Radar (3D), which sweeps around a vertical axis in a conic pattern at varying elevation angles, and measures atmospheric properties at multiple distances along the instantaneous line-of-sight.

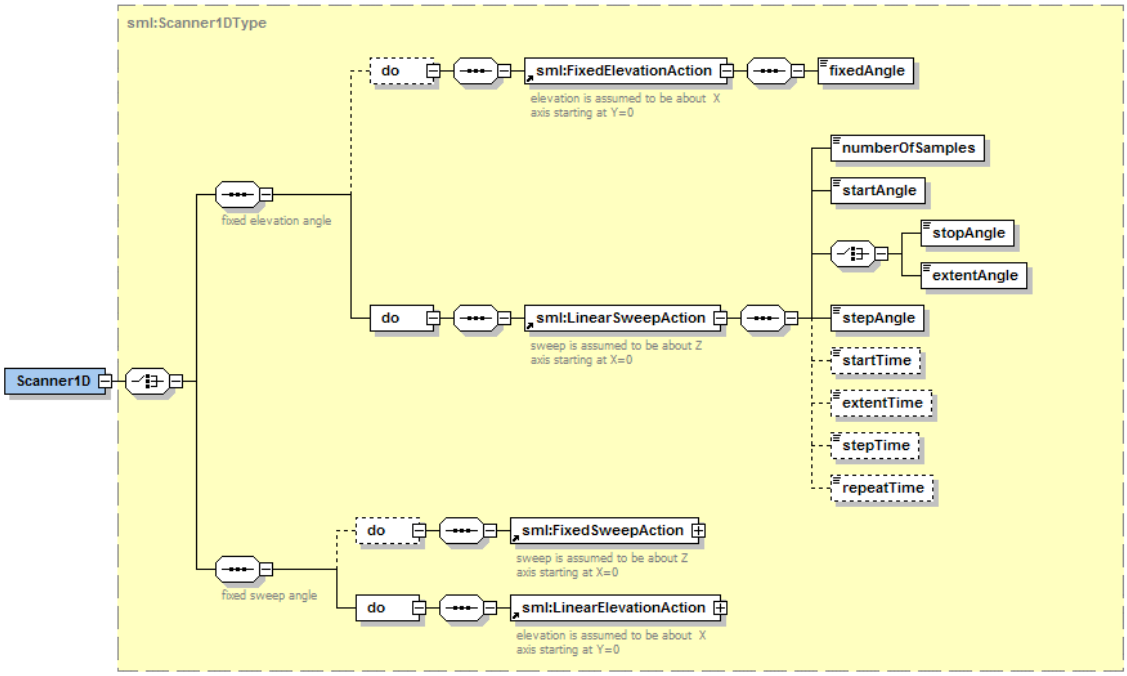


Figure 2.26. ***Scanner1D scannerProperties*.**

The 1D conic scanners could be described using the Scanner1D (Figure 2.27) with the Z axis of the scannerCRS oriented perpendicular to the observed surface. The elevation angle would typically be fixed.

For a 3D scanner/profiler, such as the volumetric Doppler radar used for weather observation, one should use the Profiler3D (Figure 2.28). It the case of the typical scanning mode for Doppler radar, one would utilize the most general case with no fixed angles or distances; in other words, one would use Scanner3D with the nesting:

```
<usesParameters>
   <Scanner3D>
      <do>
         < LinearElevationAction >
            …
         </ LinearElevationAction >
         <do>
            <LinearSweepAction>
               …
            </LinearSweepAction>
            <do>
               <LinearProfileAction>
                  …
               </ LinearProfileAction >
            </do>
         </do>
      </do>
   </Scanner3D>
</usesParameters>
```

Line-Scanner. Line-scanners sample by sweeping around a rotation axis such that they sample a linear array relative to the target (see Figure 2.21). Sweeping at either a constant or varying elevation angle, scan dimensions can be one to two dimensions, possibly include sweep, elevation, or profile components. As discussed above, a line scanner is mathematically equivalent to a conic scanner rotated approximately 90 degrees. This is perhaps the most common sensor system used for lower resolution Earth observation.

For 1D line scanners, such as AVHRR and MODIS, one could use a *Scanner1D* (Figure 2.26) with the Z axis oriented rough tangential to the observed surface. For sensor that have both the elevation and sweep components, such as the GOES imager, then one would use the *Scanner2D* properties (Figure 2.29). Finally, for a sensor that scans in one direction while measuring a profile (e.g. the GOES Sounder and TRMM PR), then one could utilize the Profiler2D properties (Figure 2.30).

Imagers. The imager class of sensors can be considered a special case of line or conic-scanners in which the entire array of pixels is sampled at an instant in time. The imager type sensor can be handled within the SensorML definition by simply setting step times for sweep and elevation to be zero (or by not defining these parameters). However, one may want to consider whether the *OpticalModel* might be more appropriate for defining the characteristics of an imager.

If one chooses to use the *ScannerModel* to define imager characteristics, then like the line scanner, they might utilize the Scanner1D (Figure 2.27) properties for single line imagers (i.e. “push-broom” cameras) and the Scanner2D (Figure 2.29) for 2D imagers (e.g. CCD array cameras).

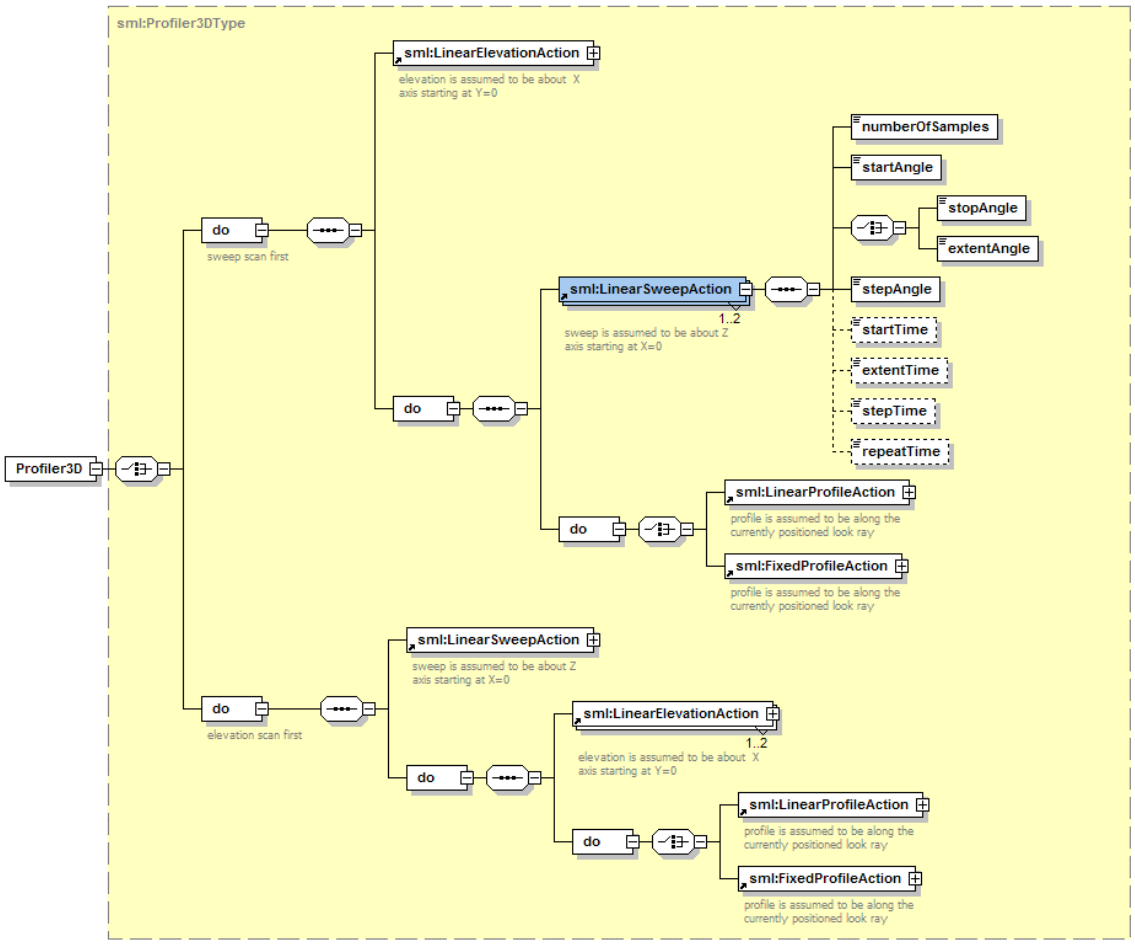


Figure 2.28. ***Profiler3D scannerProperties*.**

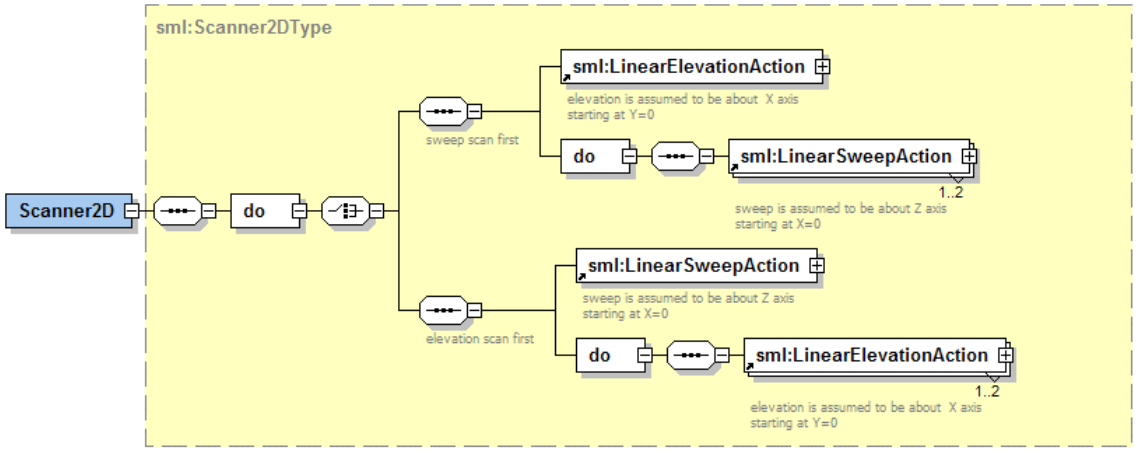


Figure 2.29. ***Scanner2D scannerProperties*.**

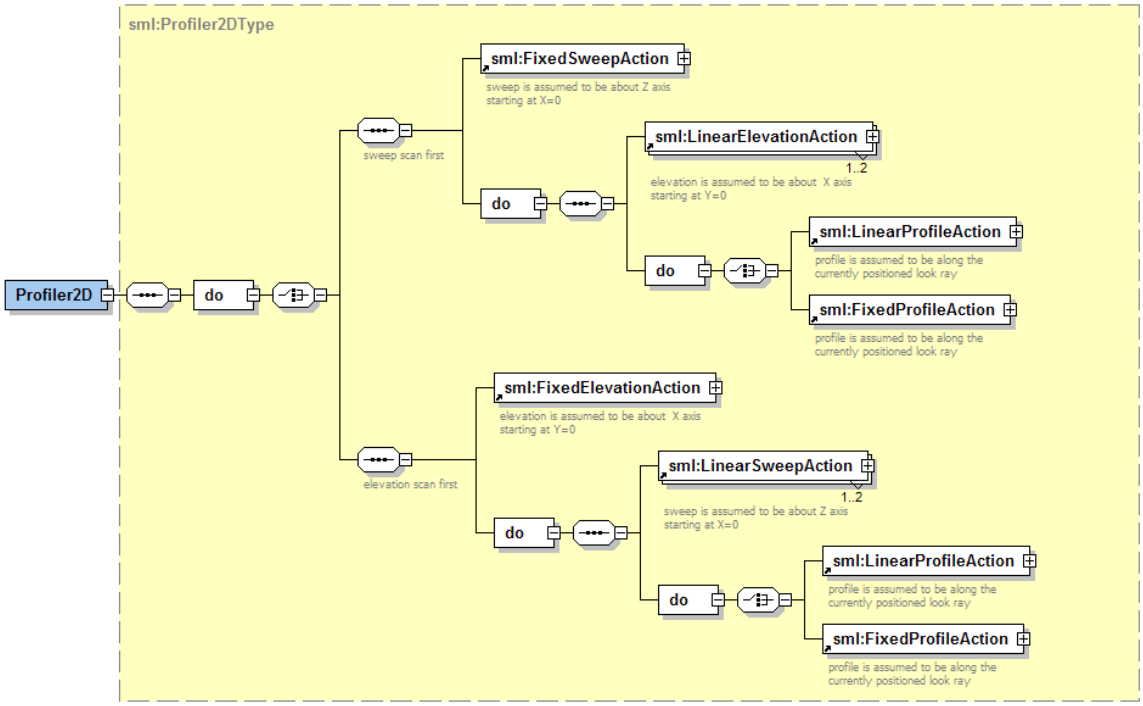


Figure 2.30. ***Scanner2D scannerProperties.***

Virtual Sensors. As discussed previously, some complex sensors could be described by using a sensor model based not on the actual sensor design, but instead based on a design that could in essence have produced the processed and distributed data set. Sensors such as SAR could possibly fall under this category. Such sensors would NOT be of the type virtual, but would instead utilize appropriate scanner properties to define these properties.

2.4.2. Optical Model

Issue Name: [**Optical Sensor Model being redesigned**. (meb, 2002-12-05)]

Issue Description:

The Optical Sensor Model is being revised through interaction with NIMA and US DOD sensor model groups. The existing schema diagram is presented here strictly as an illustration of initial concepts.

Resolution:

*** Will support Frame Cameras, Video Cameras, SLRs, etc. ****

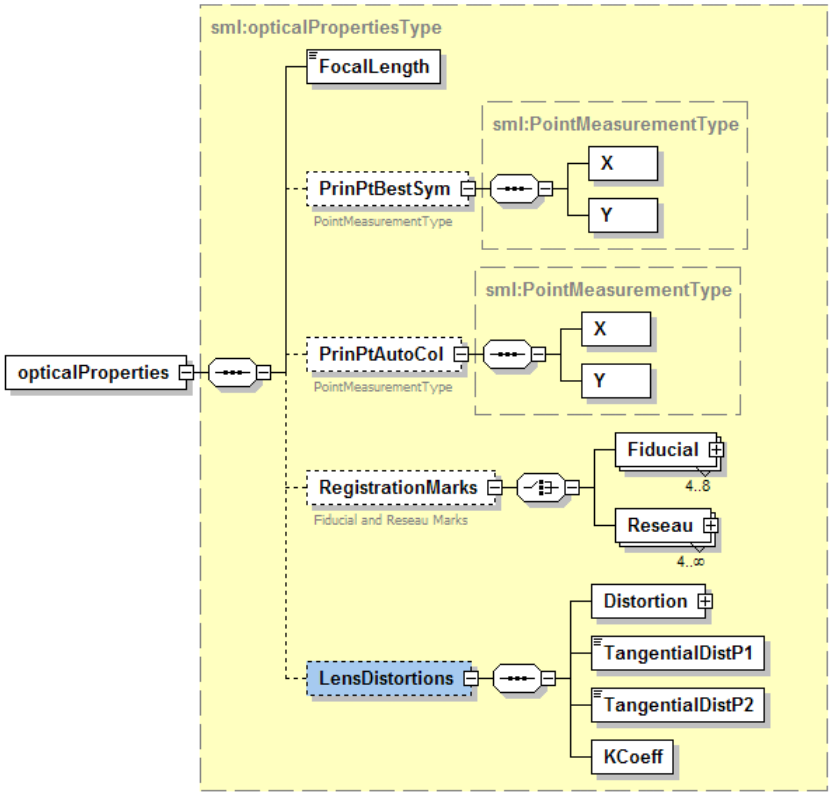


Figure 2.31. Initial schema design for *opticalProperties* used in the *OpticalModel*. This will be redesign in future release.

2.4.3. Rapid Positioning Coordinates Model (rpcModel)

Issue Name: [More complete description needed. (meb, 2002-12-05)]

Issue Description:

The Rapid Positioning Coordinates method is fully described within the OGC Abstract Specification Topic 7: The Earth Imagery Case (under 6.4. Ratios of Polynomials and 6.5. Universal Real-time Image Geometry Model). However, much of that discussion needs to be brought in here and shown how the *rpcProperties* relate to the equations.

Resolution:

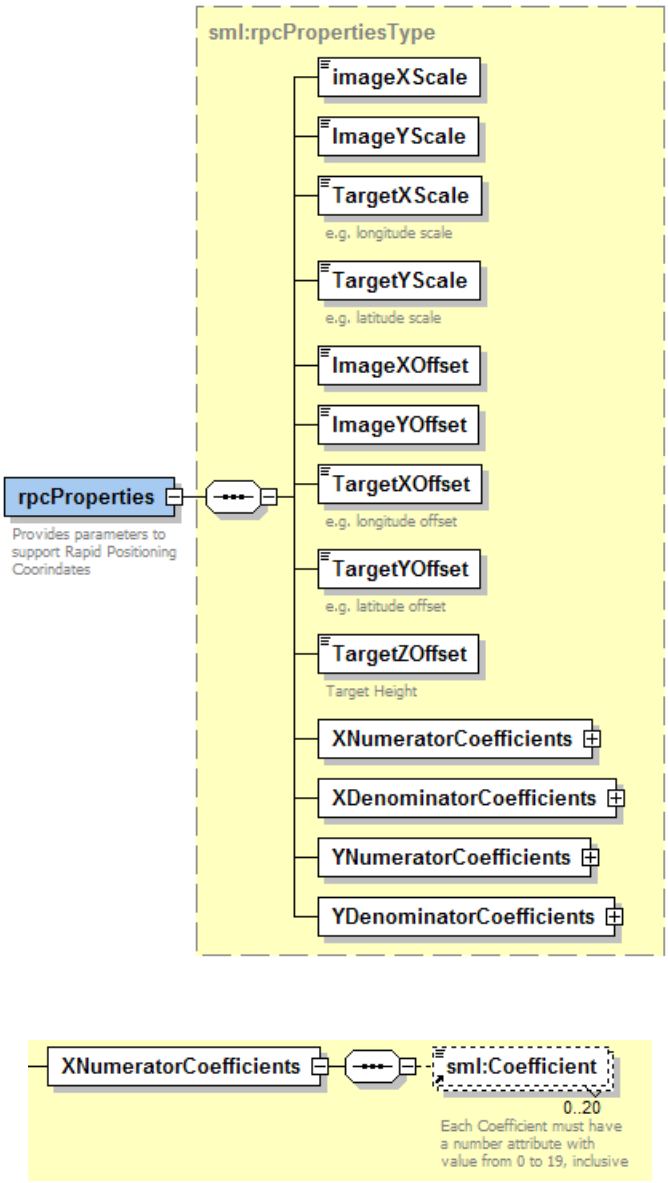
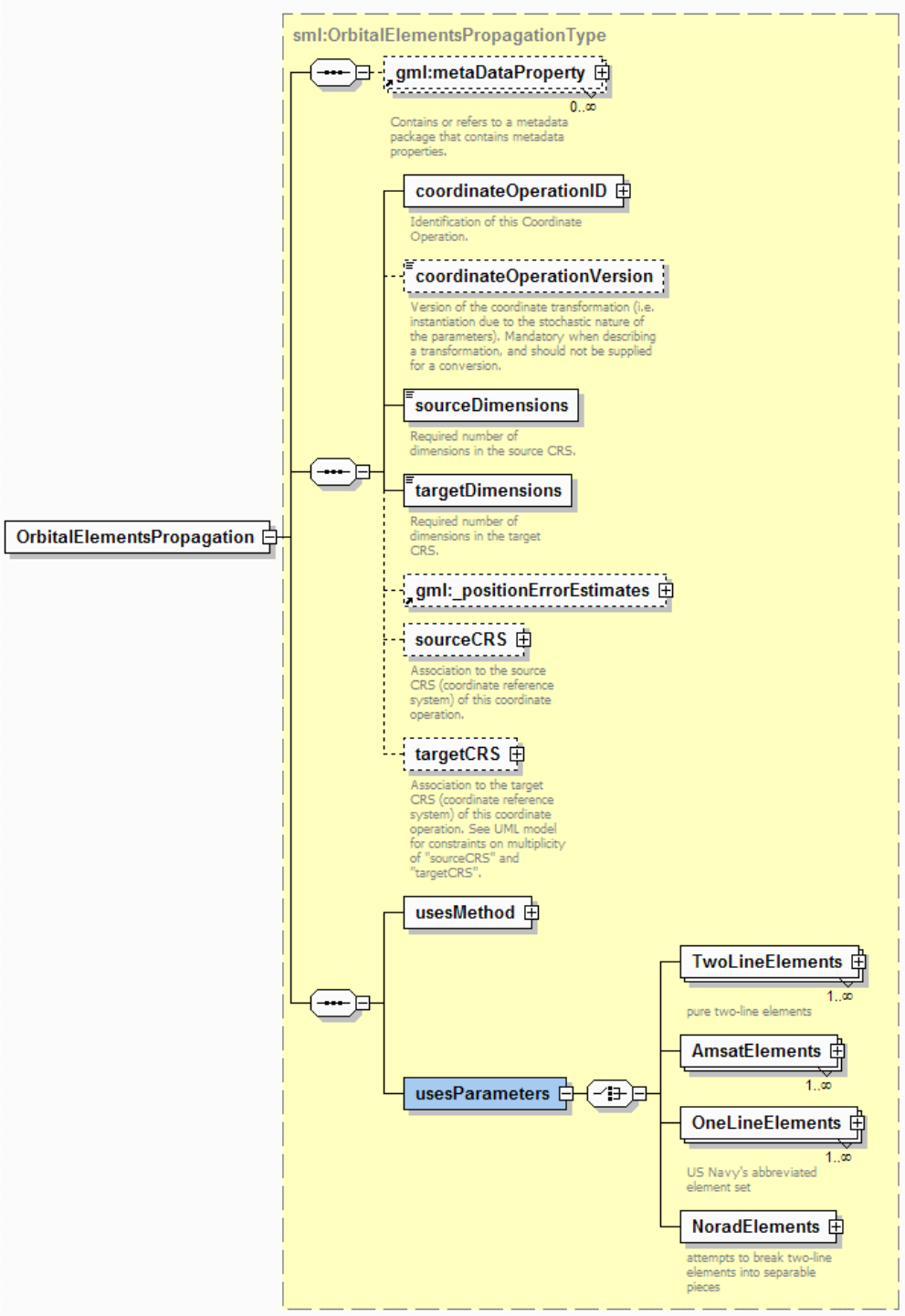
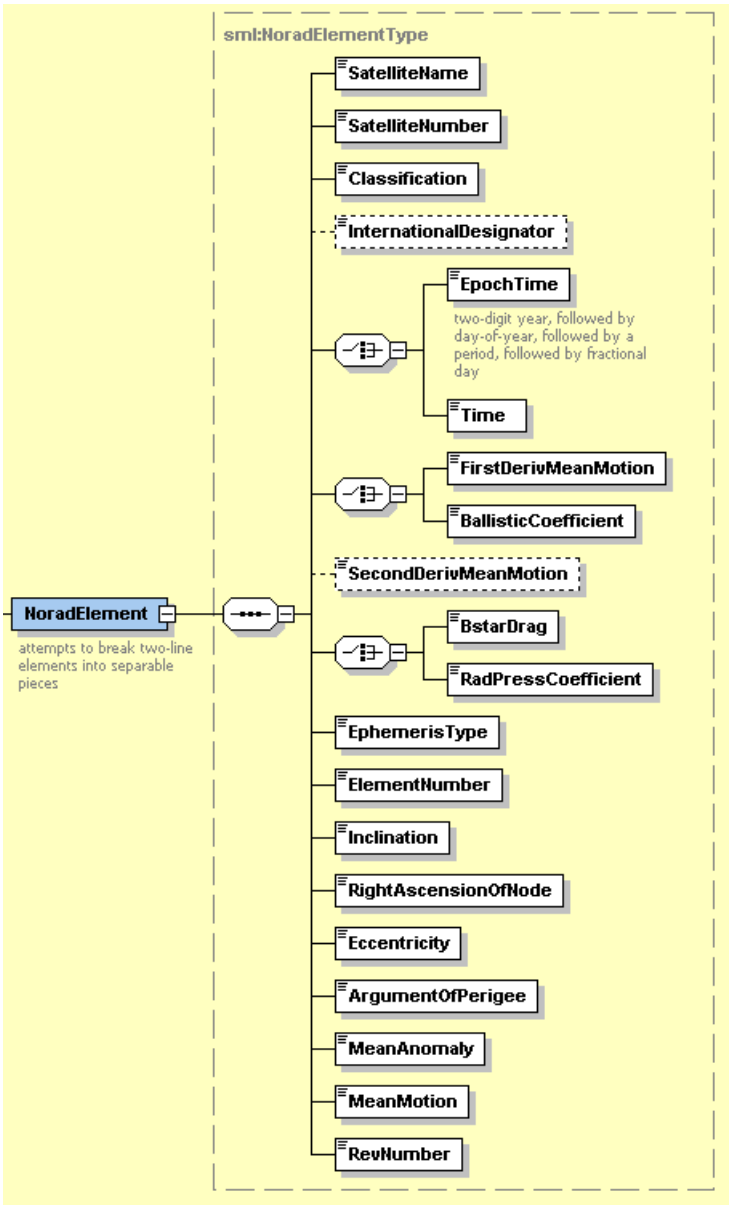


Figure 2.32. Schema diagram for *rpcProperties* used in the *RpcModel*. Full discussion will accompany Figure in future documentation release.

2.5. SensorML related Utilities, and Coordinate Transformations and Operations

2.5.1. Orbital Element Propagation





2.5.2. DynamicAffineTransformation

*** describe ***

2.6. FUTURE DIRECTIONS AND REMAINING ISSUES

This is the first release of the "complete" schema and documentation for SensorML. It is anticipated that there will be both minor and major refinements and extensions to the existing schema. There are currently plans to incorporate expertise and sensor model designs from several existing activities within NIMA, NASA, and EPA, as well as recommendations that will be received from the general sensor communities. 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 certainly provide lessons learned that will be used to refine the schemas of SensorML.

Some of the refinements and extensions will have been anticipated and some may not. Below is a limited list of recognized desires and needs in future releases:

1. The establishment of authoritative glossaries and schema repositories for sensor related terms, coordinate reference systems, and coordinate operations
2. greater conformance to schema design concepts utilized in other relevant schema, such as the use of *xlink:href* in GML3, etc.
3. more complete definitions of utility and mathematical schemas within mathUtility.xsd and transformations.xsd
4. possible transfer of some SensorML elements to GML3 (such as *sml:ObjectState* and several coordinate operation definitions
5. extension or replacement of the existing minimal opticalModel for frame cameras, video, etc.
6. sensor model for a simple sensor grid (e.g. example, a sampling pattern for ecological observations on the ground or within a microscope)
7. sensor models for other mathematical models, e.g. polynomials, homogeneous matrices, rubber sheeting using tie points
8. addition of NIMA's Replacement Sensor Model based on extension of the RPC model
9. more response characteristics, particularly for measures of accuracy and confidence
10. additional response model characteristics for chemical sampling, population measurement, etc.
11. inclusion of a new high-level sensor property for listing Nominal Design Properties; this would provide information that might be mined for use in sensor catalogs (such as expected ground resolution), but may not be robust enough for actual processing or geolocation of sensor observations.
12. Definitions for spectral radiation curves, to be used for defining both the source radiation and a remote sensor's sensitivity to various wavelengths

Annex A: XML Schemas for SensorML (normative)

A.1 SensorMLBase.xsd.

SensorMLBase.xsd provides the basic definitions and abstract elements used throughout SensorML schema.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (NSSTC) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Michael E Botts (University of Alabama in Huntsville) -
->
<schema targetNamespace="http://www.opengis.net/sensorML" xmlns:sml="http://www.opengis.net/sensorML"
xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://www.opengis.net/ows"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:xlink="http://www.w3.org/1999/xlink" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <annotation>
        <appinfo>sensorML.xsd v0.7 2002-12-20</appinfo>
        <documentation>base class definitions for elements common to several classes</documentation>
    </annotation>
    <!-- ================================================================================
       includes and imports
    ================================================================================ -->
    <include schemaLocation="./objectState.xsd"/>
    <import namespace="http://www.isotc211.org/iso19115/"
schemaLocation="../../iso19115/0.1.21/iso19115.xsd"/>
    <import namespace="http://www.opengis.net/gml" schemaLocation="./gmlStub.xsd"/>
    <import namespace="http://www.opengis.net/ows" schemaLocation="./owsStub.xsd"/>
    <!-- ================================================================================
       global abstract elements
    ================================================================================ -->
    <element name="_Platform" abstract="true">
        <annotation>
            <documentation>abstract Platform object </documentation>
        </annotation>
    </element>
    <element name="_Sensor" abstract="true">
        <annotation>
            <documentation>abstract Sensor object</documentation>
        </annotation>
    </element>
    <element name="_SensorModel" abstract="true">
        <annotation>
            <documentation>abstract object for providing geolocation models</documentation>
        </annotation>
    </element>
    <element name="_DocumentEvent" abstract="true">
        <annotation>
            <documentation>abstract object for document events within document history</documentation>
        </annotation>
    </element>
    <element name="_AssetEvent" abstract="true">
        <annotation>
            <documentation>abstract object for sensor events within sensor history</documentation>
        </annotation>
    </element>
    <!-- ================================================================================
      Sensor Model
    ================================================================================ -->
    <element name="_SensorModelParameters" type="sml:_SensorModelParametersType" abstract="true">
        <annotation>
            <documentation>abstract object for supporting sensor model properties</documentation>
```

```
        </annotation>
    </element>
    <complexType name="_SensorModelParametersType">
        <attribute ref="gml:id" use="optional"/>
    </complexType>
    <element name="SensorModelMethod" type="gml:OperationMethodType" abstract="false">
        <annotation>
            <documentation>object for supporting sensor model properties</documentation>
        </annotation>
    </element>
    <!-- ======================================================================
      Response Model  (??? no longer used ???)
    ====================================================================== -->
    <element name="ResponseModel" abstract="false">
        <annotation>
            <documentation>describes sensor's response chracteristics to the observable
stimulus</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="ows:TypedValue" maxOccurs="unbounded"/>
            </sequence>
        </complexType>
    </element>
    <!-- ======================================================================
      GenericProperty Element
    ====================================================================== -->
    <element name="genericProperty" type="sml:GenericPropertyType">
        <annotation>
            <documentation>provides means for specifying vendor specific properties</documentation>
        </annotation>
    </element>
    <complexType name="GenericPropertyType">
        <simpleContent>
            <extension base="string">
                <attribute name="name" type="string" use="required"/>
                <attribute name="dataType" type="anySimpleType" use="optional"/>
                <attribute name="uom" type="anyURI" use="optional"/>
            </extension>
        </simpleContent>
    </complexType>
    <!-- ======================================================================
      locatedUsing Element
    ====================================================================== -->
    <element name="locatedUsing" type="sml:locatedUsingType">
        <annotation>
            <documentation>provides the XML specification or service for determining state</documentation>
        </annotation>
    </element>
    <complexType name="locatedUsingType">
        <choice>
            <element ref="sml:ObjectState"/>
            <element ref="gml:_CoordinateOperation"/>
            <element ref="sml:ObjectStateProvider"/>
        </choice>
    </complexType>
    <element name="ObjectStateProvider">
        <complexType>
            <attribute name="serviceURI" type="anyURI" use="required"/>
        </complexType>
    </element>
    <!-- ======================================================================
      measures Element
    ====================================================================== -->
    <element name="measures">
        <annotation>
```

```
            <documentation>describes what property is measured by the sensor</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="sml:Measurand" minOccurs="0"/>
            </sequence>
            <attributeGroup ref="gml:AssociationAttributeGroup"/>
        </complexType>
    </element>
    <!-- ==========================================================================
      documentConstrainedBy Element
    ========================================================================== -->
    <element name="documentConstrainedBy">
        <annotation>
            <documentation>time period and authority for validity of document; also security
levels</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="sml:Constraints"/>
            </sequence>
        </complexType>
    </element>
    <element name="Constraints">
        <complexType>
            <sequence>
                <element name="validTime" type="gml:TimeStampType"/>
                <element name="securityLevel" type="iso19115:MD_SecurityConstraintsType" minOccurs="0"
maxOccurs="unbounded">
                    <annotation>
                        <documentation>iso19115:SecurityConstraint</documentation>
                    </annotation>
                </element>
                <element name="legalUse" type="iso19115:MD_LegalConstraintsType" minOccurs="0">
                    <annotation>
                        <documentation>iso19115:LegalConstraint</documentation>
                    </annotation>
                </element>
            </sequence>
        </complexType>
    </element>
    <!-- ==========================================================================
      operatedBy Element
    ========================================================================== -->
    <element name="operatedBy" type="sml:OperatorPropertyType">
        <annotation>
            <documentation>defines operator and optional points to SPS</documentation>
        </annotation>
    </element>
    <complexType name="OperatorPropertyType">
        <choice minOccurs="0">
            <element ref="sml:ResponsibleParty"/>
            <element ref="sml:SensorPlanningService"/>
        </choice>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
    <!-- ==========================================================================
      Responsible Party Element
    ========================================================================== -->
    <element name="ResponsibleParty">
        <annotation>
            <documentation>ISO19115: CI_ResponsibleParty</documentation>
        </annotation>
        <complexType>
            <complexContent>
                <extension base="iso19115:CI_ResponsiblePartyType">
```

```
                    <attribute name="id" type="anyURI" use="optional"/>
                </extension>
            </complexContent>
        </complexType>
    </element>
    <complexType name="ResponsiblePartyPropertyType">
        <sequence>
            <element ref="sml:ResponsibleParty" minOccurs="0">
                <annotation>
                    <documentation>iso19115: CI_ReponsibleParty</documentation>
                </annotation>
            </element>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
    <!-- ==========================================================================
      Sensor Planning Service Element
    ========================================================================== -->
    <element name="SensorPlanningService">
        <annotation>
            <documentation>Object which carries a URI that points to service</documentation>
        </annotation>
        <complexType>
            <complexContent>
                <extension base="gml:AbstractGMLType">
                    <sequence>
                        <element name="serviceURI" type="anyURI"/>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>
    <!-- ==========================================================================
      describedBy Element
    ========================================================================== -->
    <element name="describedBy">
        <annotation>
            <documentation>provides metadata and history of the sensor; can accept xlink:href to external
description</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="sml:AssetDescription" minOccurs="0"/>
            </sequence>
            <attributeGroup ref="gml:AssociationAttributeGroup"/>
        </complexType>
    </element>
    <!-- ==========================================================================
      AssetDescription Element
    ========================================================================== -->
    <element name="AssetDescription" type="sml:AssetDescriptionType"/>
    <complexType name="AssetDescriptionType">
        <complexContent>
            <extension base="gml:AbstractGMLType">
                <sequence>
                    <annotation>
                        <documentation>Properties specific to AssetDescription</documentation>
                    </annotation>
                    <element name="manufacturedBy" type="sml:ResponsiblePartyPropertyType"/>
                    <element name="modelNumber" type="string" minOccurs="0" maxOccurs="unbounded"/>
                    <element name="identificationNumber" type="string" minOccurs="0"
maxOccurs="unbounded"/>
                    <element name="deployingAgency" type="sml:ResponsiblePartyPropertyType"
minOccurs="0"/>
                    <element name="assetHistory" minOccurs="0">
                        <annotation>
```

```
                        <documentation>takes a collection of _AssetEvents</documentation>
                    </annotation>
                    <complexType>
                        <sequence>
                            <element ref="sml:_AssetEvent" maxOccurs="unbounded"/>
                        </sequence>
                    </complexType>
                </element>
                <element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="sml:referenceDocuments" minOccurs="0"/>
                <element ref="sml:genericProperty" minOccurs="0" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ==========================================================================
  documentedBy Element
========================================================================== -->
<element name="documentedBy">
    <annotation>
        <documentation>provides metadata and history of the document</documentation>
    </annotation>
    <complexType>
        <sequence>
            <element ref="sml:DocumentMetaData" minOccurs="0"/>
        </sequence>
        <attributeGroup ref="gml:AssociationAttributeGroup"/>
    </complexType>
</element>
<element name="DocumentMetaData">
    <annotation>
        <documentation>collection of DocumentEvents</documentation>
    </annotation>
    <complexType>
        <complexContent>
            <extension base="gml:AbstractGMLType">
                <sequence>
                    <element name="documentHistory">
                        <complexType>
                            <sequence>
                                <element ref="sml:_DocumentEvent" maxOccurs="unbounded"/>
                            </sequence>
                        </complexType>
                    </element>
                    <element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                    <element ref="sml:referenceDocuments" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</element>
<!-- ==========================================================================
  identifiedAs property Element
========================================================================== -->
<element name="identifiedAs" type="sml:identifiedAsType">
    <annotation>
        <documentation>provides basic identification (e.g. names and type)</documentation>
    </annotation>
</element>
<complexType name="identifiedAsType">
    <sequence>
        <element name="shortName" type="string"/>
        <element name="longName" type="string"/>
        <element name="type" type="anyURI">
            <annotation>
                <documentation>Pointer to a type dictionary entry</documentation>
```

```
                </annotation>
            </element>
        </sequence>
    </complexType>
    <!-- ==========================================================================
         Sample Object Element
    ========================================================================== -->
    <element name="Sample" type="sml:SampleType"/>
    <complexType name="SampleType">
        <complexContent>
            <extension base="gml:AbstractGMLType">
                <sequence>
                    <element ref="sml:hasCRS" minOccurs="0" maxOccurs="unbounded"/>
                    <element name="hasSizeOf" minOccurs="0">
                        <complexType>
                            <sequence>
                                <element name="Dimension" maxOccurs="4">
                                    <complexType>
                                        <simpleContent>
                                            <extension base="gml:MeasureType">
                                                <attribute name="axis" type="sml:DimensionAxisType"
use="required"/>
                                                <attribute name="basis" type="anyURI" use="optional"/>
                                            </extension>
                                        </simpleContent>
                                    </complexType>
                                </element>
                            </sequence>
                        </complexType>
                    </element>
                    <element ref="sml:locatedUsing" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <simpleType name="DimensionAxisType">
        <restriction base="string">
            <enumeration value="x"/>
            <enumeration value="y"/>
            <enumeration value="z"/>
            <enumeration value="time"/>
            <enumeration value="sweep"/>
            <enumeration value="elevation"/>
            <enumeration value="profile"/>
            <enumeration value="volume"/>
        </restriction>
    </simpleType>
    <!-- ==========================================================================
         sampleCollection Element
    ========================================================================== -->
    <element name="SampleCollection" type="sml:SampleCollectionType" substitutionGroup="sml:Sample"/>
    <complexType name="SampleCollectionType">
        <complexContent>
            <extension base="sml:SampleType">
                <sequence>
                    <element name="sampleMember" maxOccurs="unbounded">
                        <complexType>
                            <sequence>
                                <element ref="sml:Sample"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

```
    <!-- ==========================================================================
         Specific Sample Collection Elements
         ========================================================================== -->
    <element name="Image" type="sml:SampleCollectionType" substitutionGroup="sml:Sample"/>
    <element name="ImageSection" substitutionGroup="sml:Sample">
        <complexType>
            <complexContent>
                <extension base="sml:SampleCollectionType">
                    <sequence>
                        <element name="coordinates">
                            <annotation>
                                <documentation>These should be relative to image space</documentation>
                            </annotation>
                            <complexType>
                                <sequence>
                                    <element name="xMin"/>
                                    <element name="xMax"/>
                                    <element name="yMin"/>
                                    <element name="yMax"/>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </extension>
            </complexContent>
        </complexType>
    </element>
    <element name="ScanLine" type="sml:SampleCollectionType" substitutionGroup="sml:Sample"/>
    <!-- ==========================================================================
         CRS support
         ========================================================================== -->
    <element name="hasCRS">
        <annotation>
            <documentation>Links to or defines the object's local coordinate reference system</documentation>
        </annotation>
        <complexType>
            <sequence>
                <element ref="gml:EngineeringCRS" minOccurs="0"/>
            </sequence>
            <attributeGroup ref="gml:AssociationAttributeGroup"/>
        </complexType>
    </element>
    <!-- ==========================================================================
         Observation  Element
         ========================================================================== -->
    <element name="Measurand" type="sml:MeasurandType"/>
    <complexType name="MeasurandType">
        <complexContent>
            <extension base="gml:AbstractGMLType">
                <sequence>
                    <element ref="ows:observable"/>
                    <element name="characterizedBy" minOccurs="0">
                        <annotation>
                            <documentation>Lists sensor response characteristics (e.g. sensitivity, accuracy,
operational environment, etc). Takes ows:TypedValue</documentation>
                        </annotation>
                        <complexType>
                            <sequence>
                                <element ref="ows:TypedValue" maxOccurs="unbounded"/>
                            </sequence>
                        </complexType>
                    </element>
                    <element name="basedOn" minOccurs="0">
                        <annotation>
                            <documentation>Takes sml:Sample</documentation>
                        </annotation>
```

```
                        <complexType>
                            <sequence>
                                <element ref="sml:Sample"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ==========================================================================
      description Element
    ========================================================================== -->
    <element name="description">
        <annotation>
            <documentation>attributes:subject, date, author</documentation>
        </annotation>
        <complexType>
            <simpleContent>
                <extension base="string">
                    <attribute name="subject" type="string" use="optional"/>
                    <attribute name="dateTime" type="dateTime" use="optional"/>
                    <attribute name="author" type="string" use="optional"/>
                </extension>
            </simpleContent>
        </complexType>
    </element>
    <!-- ==========================================================================
      additionalInformation Element
    ========================================================================== -->
    <element name="referenceDocuments">
        <complexType>
            <sequence>
                <element name="Citation" type="iso19115:CI_CitationType" minOccurs="0"
maxOccurs="unbounded">
                    <annotation>
                        <documentation>iso19115:Citation</documentation>
                    </annotation>
                </element>
                <element name="OnlineResource" type="iso19115:CI_OnlineResourceType" minOccurs="0">
                    <annotation>
                        <documentation>iso19115:OnlineResource</documentation>
                    </annotation>
                </element>
            </sequence>
        </complexType>
    </element>
    <!-- ==========================================================================
       SensorModel Transformation
    ========================================================================== -->
    <element name="SensorModel" type="sml:SensorModelType"
substitutionGroup="gml:_CoordinateOperation">
        <annotation>
            <documentation>Provide model for location of observations</documentation>
        </annotation>
    </element>
    <complexType name="SensorModelType">
        <complexContent>
            <extension base="gml:AbstractCoordinateOperationType">
                <sequence>
                    <element name="usesMethod">
                        <complexType>
                            <sequence>
                                <element ref="gml:OperationMethod" minOccurs="0"/>
                            </sequence>
                            <attributeGroup ref="gml:AssociationAttributeGroup"/>
```

```
                    </complexType>
                </element>
                <element name="usesParameters">
                    <complexType>
                        <sequence>
                            <element ref="sml:_SensorModelParameters" minOccurs="0"/>
                        </sequence>
                        <attributeGroup ref="gml:AssociationAttributeGroup"/>
                    </complexType>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
</schema>
```

A.2 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 (NSSTC) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Michael E Botts (University of Alabama in Huntsville) -
->
<schema targetNamespace="http://www.opengis.net/sensorML"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML" xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <annotation>
        <appinfo>sensor.xsd v0.7 2002-12-20</appinfo>
        <documentation>Ongoing xs:schema definition for SensorML</documentation>
    </annotation>
    <!-- ================================================================================
        includes and imports
    ================================================================================ -->
    <include schemaLocation="sensorMLBase.xsd"/>
    <!-- ================================================================================
        global elements
    ================================================================================ -->
    <element name="Sensor" type="sml:SensorType" substitutionGroup="sml:_Sensor"/>
    <element name="SensorGroup" type="sml:SensorGroupType" substitutionGroup="sml:Sensor"/>
    <!--
    <element name="SensorArray" type="sml:SensorArrayType" substitutionGroup="sml:Sensor"/>
    <element name="SensorPackage" type="sml:SensorPackageType" substitutionGroup="sml:Sensor"/>
    -->
    <!-- ================================================================================
            sensor components
     ================================================================================ -->
    <complexType name="SensorType">
        <complexContent>
            <extension base="gml:AbstractGMLType">
                <sequence>
                    <annotation>
                        <documentation>Main Properties specific to Sensor description</documentation>
                    </annotation>
                    <element ref="sml:identifiedAs"/>
                    <element ref="sml:documentConstrainedBy">
                        <annotation>
                            <documentation>time period and authority for validity of document; also security
levels</documentation>
                        </annotation>
                    </element>
                    <element name="attachedTo" minOccurs="0">
                        <annotation>
```

```
                        <documentation>Points to the platform on which the sensor is
attached</documentation>
                    </annotation>
                    <complexType>
                        <sequence>
                            <element ref="sml:_Platform"/>
                        </sequence>
                    </complexType>
                </element>
                <element ref="sml:hasCRS" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="sml:locatedUsing" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="sml:measures" maxOccurs="unbounded"/>
                <element ref="sml:operatedBy" minOccurs="0" maxOccurs="unbounded"/>
                <element ref="sml:describedBy" minOccurs="0"/>
                <element ref="sml:documentedBy" minOccurs="0">
                    <annotation>
                        <documentation>Provides metadata and history of the document</documentation>
                    </annotation>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ========================================================================
     sensor collections
     ======================================================================== -->
<complexType name="SensorGroupType">
    <complexContent>
        <extension base="sml:SensorType">
            <sequence>
                <element name="sensorMember" maxOccurs="unbounded">
                    <annotation>
                        <documentation>includes a Sensor description or a link to a Sensor
description</documentation>
                    </annotation>
                    <complexType>
                        <sequence>
                            <element ref="sml:Sensor" minOccurs="0"/>
                        </sequence>
                        <attributeGroup ref="gml:AssociationAttributeGroup"/>
                    </complexType>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
</schema>
```

A.3 PLATFORM.XSD

Provides initial 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 (NSSTC) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Michael E Botts (University of Alabama in Huntsville) -
->
<schema targetNamespace="http://www.opengis.net/sensorML" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:sml="http://www.opengis.net/sensorML"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <annotation>
        <appinfo>platform.xsd v0.7 2002-12-20</appinfo>
        <documentation>basic definitions for platforms used in conjunction with SensorML</documentation>
    </annotation>
    <!-- ========================================================================
```

```
    includes and imports
  ========================================================================== -->
  <include schemaLocation="sensorMLBase.xsd"/>
  <!-- ========================================================================
    global elements
  ========================================================================== -->
  <element name="Platform" type="sml:PlatformType" substitutionGroup="sml:_Platform">
    <annotation>
      <documentation>Has required attributes: id and platformReference</documentation>
    </annotation>
  </element>
  <element name="StationaryPlatform" type="sml:StationaryPlatformType" substitutionGroup="sml:_Platform"/>
  <element name="AttachedPlatform" type="sml:AttachedPlatform" substitutionGroup="sml:_Platform"/>
  <element name="SatellitePlatform" type="sml:SatellitePlatformType" substitutionGroup="sml:_Platform"/>
  <element name="AircraftPlatform" type="sml:AircraftPlatformType" substitutionGroup="sml:_Platform"/>
  <element name="LandVehiclePlatform" type="sml:LandVehiclePlatformType"
substitutionGroup="sml:_Platform"/>
  <element name="WaterVehiclePlatform" type="sml:WaterVehiclePlatformType"
substitutionGroup="sml:_Platform"/>
  <!-- ========================================================================
    Fundamental types
  ========================================================================== -->
  <complexType name="PlatformType" abstract="true">
    <annotation>
      <documentation>Has required attribute: id</documentation>
    </annotation>
    <complexContent>
      <extension base="gml:AbstractGMLType">
        <sequence>
          <annotation>
            <documentation>Properties specific for Platform</documentation>
          </annotation>
          <element ref="sml:identifiedAs"/>
          <element ref="sml:documentConstrainedBy" minOccurs="0"/>
          <element name="carries" minOccurs="0" maxOccurs="unbounded">
            <annotation>
              <documentation>Either an AttachedPlaform or Sensor</documentation>
            </annotation>
            <complexType>
              <choice minOccurs="0">
                <element ref="sml:AttachedPlatform"/>
                <element ref="sml:_Sensor"/>
              </choice>
              <attributeGroup ref="gml:AssociationAttributeGroup"/>
            </complexType>
          </element>
          <element ref="sml:hasCRS" minOccurs="0" maxOccurs="unbounded"/>
          <element ref="sml:locatedUsing" minOccurs="0"/>
          <element ref="sml:describedBy" minOccurs="0"/>
          <element ref="sml:documentedBy" minOccurs="0"/>
        </sequence>
      </extension>
    </complexContent>
  </complexType>
  <!--========================================================================
      Platform Types
   ========================================================================-->
  <complexType name="StationaryPlatformType">
    <complexContent>
      <extension base="sml:PlatformType"/>
    </complexContent>
  </complexType>
  <complexType name="SatellitePlatformType">
    <complexContent>
      <extension base="sml:PlatformType"/>
    </complexContent>
```

```
    </complexType>
    <complexType name="AircraftPlatformType">
        <complexContent>
            <extension base="sml:PlatformType"/>
        </complexContent>
    </complexType>
    <complexType name="LandVehiclePlatformType">
        <complexContent>
            <extension base="sml:PlatformType"/>
        </complexContent>
    </complexType>
    <complexType name="WaterVehiclePlatformType">
        <complexContent>
            <extension base="sml:PlatformType"/>
        </complexContent>
    </complexType>
    <complexType name="AttachedPlatform">
        <complexContent>
            <extension base="sml:PlatformType">
                <sequence>
                    <element name="attachedTo">
                        <annotation>
                            <documentation>Either a Platform or PlatformReference</documentation>
                        </annotation>
                        <complexType>
                            <sequence>
                                <element ref="sml:_Platform"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

A.4 EVENT.XSD.

event.xsd provides definitions for events used in *history* property. These include *_SensorEvent* and *_DocumentEvent* and their derivatives.

```
<?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) -
->
<!--Generated by XML Authority. Conforms to w3c http://www.w3.org/2001/XMLSchema-->
<schema targetNamespace="http://www.opengis.net/sensorML" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:sml="http://www.opengis.net/sensorML"
xmlns:iso19115="http://www.isotc211.org/iso19115/" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <annotation>
        <documentation>events.xsd v0.7 2002-12-12</documentation>
        <documentation>Defines sensor and document events to be used in sensorML histories</documentation>
    </annotation>
    <!-- ==========================================================================
       Imports and includes
     ========================================================================== -->
    <include schemaLocation="sensorMLBase.xsd"/>
    <import namespace="http://www.opengis.net/gml" schemaLocation="../../gml/3.0/base/gmlBase.xsd"/>
    <!-- ==========================================================================
       global elements
     ========================================================================== -->
    <element name="DocumentCreation" type="sml:DocumentCreationType"
substitutionGroup="sml:_DocumentEvent"/>
```

```
    <element name="DocumentModification" type="sml:DocumentModificationType"
substitutionGroup="sml:_DocumentEvent"/>
    <element name="DocumentApproval" type="sml:DocumentApprovalType"
substitutionGroup="sml:_DocumentEvent"/>
    <element name="SensorCreation" type="sml:AssetCreationType" substitutionGroup="sml:_AssetEvent"/>
    <element name="SensorDeployment" type="sml:AssetDeploymentType"
substitutionGroup="sml:_AssetEvent"/>
    <element name="SensorInspection" type="sml:AssetInspectionType" substitutionGroup="sml:_AssetEvent"/>
    <element name="SensorDecommission" type="sml:AssetDecommissionedType"
substitutionGroup="sml:_AssetEvent"/>
    <element name="SensorCalibration" type="sml:SensorCalibrationType"
substitutionGroup="sml:_AssetEvent"/>
    <!-- ============================================================================
      base abstract eventType
    ============================================================================== -->
    <complexType name="_EventType" abstract="true">
        <complexContent>
            <extension base="gml:AbstractGMLType">
                <sequence>
                    <element name="byWhom" type="sml:ResponsiblePartyPropertyType" minOccurs="0"
maxOccurs="unbounded"/>
                    <element ref="gml:timeStamp"/>
                    <element ref="sml:description" minOccurs="0" maxOccurs="unbounded"/>
                    <element ref="sml:referenceDocuments" minOccurs="0"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ============================================================================
       document events
    ============================================================================== -->
    <complexType name="DocumentCreationType">
        <complexContent>
            <extension base="sml:_EventType">
                <sequence>
                    <element name="versionNumber" type="string"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ============================================================================== -->
    <complexType name="DocumentModificationType">
        <complexContent>
            <extension base="sml:DocumentCreationType">
                <sequence>
                    <element name="modification" maxOccurs="unbounded">
                        <complexType>
                            <simpleContent>
                                <extension base="string">
                                    <attribute name="subject" type="string" use="optional"/>
                                </extension>
                            </simpleContent>
                        </complexType>
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <!-- ============================================================================== -->
    <complexType name="DocumentApprovalType">
        <complexContent>
            <extension base="sml:_EventType">
                <sequence>
                    <element name="approvedBy" type="sml:ResponsiblePartyPropertyType"/>
                </sequence>
            </extension>
```

```
    </complexContent>
</complexType>
<!-- ================================================================================
    sensor and platform events
================================================================================ -->
<complexType name="AssetCreationType">
    <complexContent>
        <extension base="sml:_EventType"/>
    </complexContent>
</complexType>
<!-- ================================================================================ -->
<complexType name="AssetDeploymentType">
    <complexContent>
        <extension base="sml:_EventType">
            <sequence>
                <element name="where" type="string"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ================================================================================ -->
<complexType name="AssetInspectionType">
    <complexContent>
        <extension base="sml:_EventType">
            <sequence>
                <element name="status">
                    <simpleType>
                        <restriction base="string">
                            <enumeration value="OK"/>
                            <enumeration value="DEFECTIVE"/>
                            <enumeration value="NEEDS ATTENTION"/>
                        </restriction>
                    </simpleType>
                </element>
                <element name="issue">
                    <complexType>
                        <sequence>
                            <element name="description" type="string"/>
                            <element name="actionTaken" type="string"/>
                        </sequence>
                        <attribute name="subject" type="string" use="optional"/>
                    </complexType>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ================================================================================ -->
<complexType name="AssetDecommissionedType">
    <complexContent>
        <extension base="sml:_EventType">
            <sequence>
                <element name="reason" type="string" maxOccurs="unbounded"/>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ================================================================================ -->
<complexType name="SensorCalibrationType">
    <complexContent>
        <extension base="sml:_EventType">
            <sequence>
                <element name="calibrationResult" maxOccurs="unbounded">
                    <complexType>
                        <simpleContent>
                            <extension base="string">
```

```
                                    <attribute name="subject" type="string" use="optional"/>
                                </extension>
                            </simpleContent>
                        </complexType>
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

A.5 GENERALRESPONSE.XSD

Defines schema for a general response characteristics, such as sensitivity, accuracy, thresholds, operating conditions, etc.

```
<?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) -
->
<schema targetNamespace="http://www.opengis.net/sensorML" xmlns:sml="http://www.opengis.net/sensorML"
xmlns:ows="http://www.opengis.net/ows" xmlns:gml="http://www.opengis.net/gml"
xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <annotation>
        <documentation>sensorML.xsd v0.7 2002-12-20</documentation>
        <documentation>Defines basic ResponseType definition and commonly used sensor
characteristics</documentation>
    </annotation>
    <!-- ================================================================================
       includes and xs:imports
    ================================================================================ -->
    <!--<xs:include schemaLocation="../extSchema/value.xsd"/>-->
    <include schemaLocation="../sensorMLBase.xsd"/>
    <!-- ================================================================================
       global  elements
    ================================================================================ -->
    <!--
    <xs:element name="GeneralResponse" type="sml:GeneralResponseType"
substitutionGroup="sml:_ResponseModel">
        <xs:annotation>
            <xs:documentation>optional attribute: id</xs:documentation>
        </xs:annotation>
    </xs:element>
    -->
    <!-- ================================================================================
       common characteristics measures
    ================================================================================ -->
    <element name="relativeAccuracy" substitutionGroup="ows:TypedValue">
        <complexType>
            <complexContent>
                <restriction base="ows:TypedValueExtentType">
                    <sequence>
                        <element ref="sml:condition" minOccurs="0" maxOccurs="unbounded"/>
                        <element name="low" type="ows:TypedQuantityType" minOccurs="0"/>
                        <element name="high" type="ows:TypedQuantityType" minOccurs="0"/>
                    </sequence>
                </restriction>
            </complexContent>
        </complexType>
    </element>
    <element name="resolution" type="ows:TypedQuantityType" substitutionGroup="ows:TypedValue"/>
    <element name="threshold" type="ows:TypedQuantityType" substitutionGroup="ows:TypedValue"/>
    <element name="capacity" type="ows:TypedQuantityType" substitutionGroup="ows:TypedValue"/>
    <element name="survivableRange" substitutionGroup="ows:TypedValue">
```

```
        <complexType>
            <complexContent>
                <restriction base="ows:TypedValueExtentType">
                    <sequence>
                        <element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
                        <element name="low" type="ows:TypedQuantityType" minOccurs="0"/>
                        <element name="high" type="ows:TypedQuantityType" minOccurs="0"/>
                    </sequence>
                </restriction>
            </complexContent>
        </complexType>
    </element>
    <element name="operationalRange" substitutionGroup="ows:TypedValue">
        <complexType>
            <complexContent>
                <restriction base="ows:TypedValueExtentType">
                    <sequence>
                        <element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
                        <element name="low" type="ows:TypedQuantityType" minOccurs="0"/>
                        <element name="high" type="ows:TypedQuantityType" minOccurs="0"/>
                    </sequence>
                </restriction>
            </complexContent>
        </complexType>
    </element>
    <element name="dynamicRange" substitutionGroup="ows:TypedValue">
        <complexType>
            <complexContent>
                <restriction base="ows:TypedValueExtentType">
                    <sequence>
                        <element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
                        <element name="low" type="ows:TypedQuantityType" minOccurs="0"/>
                        <element name="high" type="ows:TypedQuantityType" minOccurs="0"/>
                    </sequence>
                </restriction>
            </complexContent>
        </complexType>
    </element>
    <element name="measurementMethod" type="ows:TypedCategoryType"
substitutionGroup="ows:TypedValue"/>
    <!-- ==========================================================================
       base Response types
    ========================================================================== -->
    <!--
    <xs:complexType name="GeneralResponseType">
        <xs:annotation>
            <xs:documentation>Defines various response and quality characteristics for the sensor taking value of
_responseProperty</xs:documentation>
        </xs:annotation>
        <xs:sequence>
            <xs:element ref="sml:_responseProperty" minOccurs="0" maxOccurs="unbounded"/>
        </xs:sequence>
        <xs:attribute name="id" type="xs:ID" use="optional"/>
    </xs:complexType>
    -->
    <element name="condition" type="sml:conditionType" substitutionGroup="gml:_MetaData"/>
    <complexType name="conditionType">
        <complexContent>
            <extension base="gml:AbstractMetaDataType">
                <sequence>
                    <element name="description" type="string" minOccurs="0" maxOccurs="unbounded"/>
                    <element name="between" type="ows:TypedValueExtentType" maxOccurs="unbounded"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
```

```
</schema>
```

A.6 RADIATIONRESPONSE.XSD

An example of a specific response model, *RadiationResponse* provides schema for defining the response of a radiometer.

```
<?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/sensorML"
xmlns:xs="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
xmlns:sml="http://www.opengis.net/sensorML" xmlns:ows="http://www.opengis.net/ows"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:annotation>
        <xs:documentation>sensorML.xsd v0.7 2002-12-20</xs:documentation>
        <xs:documentation>Defines specific responseType for radiation sensors</xs:documentation>
    </xs:annotation>
    <!-- ========================================================================
       includes and xs:imports
    ========================================================================== -->
    <xs:include schemaLocation="../sensorMLBase.xsd"/>
    <xs:include schemaLocation="generalResponse.xsd"/>
    <!-- ========================================================================
       elements
    ========================================================================== -->
    <xs:element name="RadiationResponse" type="sml:RadiationResponseType"
substitutionGroup="sml:ResponseModel">
        <xs:annotation>
            <xs:documentation>susbstGrp: _ResponseModel</xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="peakWavelength" substitutionGroup="ows:TypedValue">
        <xs:annotation>
            <xs:documentation>Quantity</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:simpleContent>
                <xs:restriction base="ows:TypedQuantityType">
                    <xs:attribute name="axis" default="http://www.opengis.net/sensorML#wavelength"/>
                </xs:restriction>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="bandWidth" substitutionGroup="ows:TypedValue">
        <xs:annotation>
            <xs:documentation>Quantity</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:simpleContent>
                <xs:restriction base="ows:TypedQuantityType">
                    <xs:attribute name="axis" default="http://www.opengis.net/sensorML#bandWidth"/>
                </xs:restriction>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="spectralResponse" type="ows:TypedValueArrayType"
substitutionGroup="ows:TypedValue">
        <xs:annotation>
            <xs:documentation>WORK ON THIS ... waiting for OM changes </xs:documentation>
        </xs:annotation>
    </xs:element>
    <xs:element name="polarizationDirection" substitutionGroup="ows:TypedValue">
```

```
        <xs:annotation>
            <xs:documentation>CategoryType</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:simpleContent>
                <xs:restriction base="ows:TypedCategoryType">
                    <xs:attribute name="axis" default="http://www.opengis.net/sensorML#polarizationDirection"/>
                </xs:restriction>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="polarizationAngle" substitutionGroup="ows:TypedValue">
        <xs:annotation>
            <xs:documentation>QuantityType</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:simpleContent>
                <xs:restriction base="ows:TypedQuantityType">
                    <xs:attribute name="axis" default="http://www.opengis.net/sensorML#polarizationAngle"/>
                </xs:restriction>
            </xs:simpleContent>
        </xs:complexType>
    </xs:element>
    <xs:element name="spectralRange" substitutionGroup="ows:TypedValue">
        <xs:annotation>
            <xs:documentation>ValueExtent</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:complexContent>
                <xs:restriction base="ows:TypedValueExtentType">
                    <xs:sequence>
                        <xs:element ref="gml:metaDataProperty" minOccurs="0" maxOccurs="unbounded"/>
                        <xs:element name="low" type="ows:TypedQuantityType" minOccurs="0"/>
                        <xs:element name="high" type="ows:TypedQuantityType" minOccurs="0"/>
                    </xs:sequence>
                </xs:restriction>
            </xs:complexContent>
        </xs:complexType>
    </xs:element>
    <!-- ========================================================================
       complex types
    ======================================================================== -->
    <xs:complexType name="RadiationResponseType">
        <xs:complexContent>
            <xs:extension base="gml:AbstractGMLType">
                <xs:sequence>
                    <xs:element ref="sml:peakWavelength" minOccurs="0"/>
                    <xs:element ref="sml:bandWidth" minOccurs="0"/>
                    <xs:element ref="sml:spectralRange" minOccurs="0"/>
                    <xs:choice minOccurs="0">
                        <xs:element ref="sml:polarizationAngle"/>
                        <xs:element ref="sml:polarizationDirection"/>
                    </xs:choice>
                    <xs:element ref="sml:spectralResponse" minOccurs="0"/>
                    <xs:element ref="sml:threshold" minOccurs="0"/>
                    <xs:element ref="sml:resolution" minOccurs="0"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SpectralResponseType"/>
</xs:schema>
```

A.7 SCANNERMODEL.XSD

Provides base and derived properties for the scanner sensor model, as well as defines *ScannerModel*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (NSSTC) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Michael E Botts (University of Alabama in Huntsville) -
->
<schema targetNamespace="http://www.opengis.net/sensorML" xmlns:sml="http://www.opengis.net/sensorML"
xmlns="http://www.w3.org/2001/XMLSchema" xmlns:gml="http://www.opengis.net/gml"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <annotation>
        <appinfo>scannerModel.xsd v0.7 2002-12-20</appinfo>
        <documentation>SensorML sensor model for supporting all scanners and profilers</documentation>
    </annotation>
    <!-- ================================================================================
        includes and imports
    ================================================================================ -->
    <include schemaLocation="../sensorMLBase.xsd"/>
    <import namespace="http://www.opengis.net/gml" schemaLocation="../../../gml/3.0/base/operations.xsd"/>
    <!-- ================================================================================
        global elements
    ================================================================================ -->
    <!-- ==============   Fundamental Types
=================================================================-->
    <element name="scannerProperties" type="sml:scannerPropertiesType"
substitutionGroup="sml:_SensorModelParameters"/>
    <complexType name="scannerPropertiesType">
        <complexContent>
            <extension base="sml:_SensorModelParametersType">
                <sequence>
                    <element ref="sml:do"/>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="do">
        <complexType>
            <sequence>
                <element ref="sml:_ScanAction" maxOccurs="unbounded"/>
                <element ref="sml:do" minOccurs="0"/>
            </sequence>
        </complexType>
    </element>
    <element name="_ScanAction" type="sml:_ScanActionType" abstract="true"/>
    <!--
    <xs:element name="_SampleAction" abstract="true" substitutionGroup="sml:_ScanAction"/>
    <xs:element name="_TranslationAction" abstract="true" substitutionGroup="sml:_ScanAction"/>
    <xs:element name="_RotationAction" abstract="true" substitutionGroup="sml:_ScanAction"/>
    <xs:element name="_CalibrationAction" abstract="true" substitutionGroup="sml:_ScanAction"/>
    -->
    <!-- ==============   Fundamental Action Types
=============================================================-->
    <element name="SimpleRotation" type="sml:SimpleRotationType" substitutionGroup="sml:_ScanAction"/>
    <element name="LinearRotation" type="sml:LinearRotationType" substitutionGroup="sml:_ScanAction"/>
    <element name="SimpleTranslation" type="sml:SimpleTranslationType"
substitutionGroup="sml:_ScanAction"/>
    <element name="LinearTranslation" type="sml:LinearTranslationType" substitutionGroup="sml:_ScanAction"/>
    <!-- ==============   Spherical Scanner Actions
=============================================================-->
    <element name="LinearSweepAction" substitutionGroup="sml:_ScanAction">
        <annotation>
            <documentation>sweep is assumed to be about Z axis starting at X=0</documentation>
        </annotation>
```

```
        <complexType>
            <complexContent>
                <restriction base="sml:LinearRotationType">
                    <sequence>
                        <element name="numberOfSamples" type="positiveInteger"/>
                        <element name="startAngle" type="sml:RotationValueType"/>
                        <choice>
                            <element name="stopAngle" type="sml:RotationValueType"/>
                            <element name="extentAngle" type="sml:RotationValueType"/>
                        </choice>
                        <element name="stepAngle" type="sml:RotationValueType" minOccurs="0">
                            <annotation>
                                <documentation>can be derived from numberOfSamples and direction attribute;
option to explicitly define here</documentation>
                            </annotation>
                        </element>
                        <element name="startTime" type="sml:TimeValueType" minOccurs="0">
                            <annotation>
                                <documentation>can be relative to local temporalCRS, such as
scanStartTime</documentation>
                            </annotation>
                        </element>
                        <element name="extentTime" type="sml:TimeValueType" minOccurs="0"/>
                        <element name="stepTime" type="sml:TimeValueType" minOccurs="0"/>
                        <element name="repeatTime" type="sml:TimeValueType" minOccurs="0"/>
                    </sequence>
                    <attribute name="aboutAxis" fixed="z"/>
                </restriction>
            </complexContent>
        </complexType>
    </element>
    <element name="FixedSweepAction" substitutionGroup="sml:_ScanAction">
        <annotation>
            <documentation>sweep is assumed to be about Z axis starting at X=0</documentation>
        </annotation>
        <complexType>
            <complexContent>
                <restriction base="sml:SimpleRotationType">
                    <sequence>
                        <element name="fixedAngle" type="sml:RotationValueType"/>
                    </sequence>
                    <attribute name="aboutAxis" fixed="z"/>
                </restriction>
            </complexContent>
        </complexType>
    </element>
    <element name="LinearElevationAction" substitutionGroup="sml:_ScanAction">
        <annotation>
            <documentation>elevation is assumed to be about  X axis starting at Y=0</documentation>
        </annotation>
        <complexType>
            <complexContent>
                <restriction base="sml:LinearRotationType">
                    <sequence>
                        <element name="numberOfSamples" type="positiveInteger"/>
                        <element name="startAngle" type="sml:RotationValueType"/>
                        <choice>
                            <element name="stopAngle" type="sml:RotationValueType"/>
                            <element name="extentAngle" type="sml:RotationValueType"/>
                        </choice>
                        <element name="stepAngle" type="sml:RotationValueType" minOccurs="0">
                            <annotation>
                                <documentation>can be derived from numberOfSamples and direction attribute;
option to explicitly define here</documentation>
                            </annotation>
                        </element>
```

```
                        <element name="startTime" type="sml:TimeValueType" minOccurs="0">
                            <annotation>
                                <documentation>can be relative to local temporalCRS, such as
scanStartTime</documentation>
                            </annotation>
                        </element>
                        <element name="extentTime" type="sml:TimeValueType" minOccurs="0"/>
                        <element name="stepTime" type="sml:TimeValueType" minOccurs="0"/>
                        <element name="repeatTime" type="sml:TimeValueType" minOccurs="0"/>
                    </sequence>
                    <attribute name="aboutAxis" fixed="x"/>
                </restriction>
            </complexContent>
        </complexType>
    </element>
    <element name="FixedElevationAction" substitutionGroup="sml:_ScanAction">
        <annotation>
            <documentation>elevation is assumed to be about  X axis starting at Y=0</documentation>
        </annotation>
        <complexType>
            <complexContent>
                <restriction base="sml:SimpleRotationType">
                    <sequence>
                        <element name="fixedAngle" type="sml:RotationValueType"/>
                    </sequence>
                    <attribute name="aboutAxis" fixed="x"/>
                </restriction>
            </complexContent>
        </complexType>
    </element>
    <element name="LinearProfileAction" substitutionGroup="sml:_ScanAction">
        <annotation>
            <documentation>profile is assumed to be along the currently positioned look ray</documentation>
        </annotation>
        <complexType>
            <complexContent>
                <restriction base="sml:LinearTranslationType">
                    <sequence>
                        <element name="numberOfSamples" type="positiveInteger"/>
                        <element name="startDistance" type="sml:DistanceValueType"/>
                        <choice>
                            <element name="stopDistance" type="sml:DistanceValueType"/>
                            <element name="extentDistance" type="sml:DistanceValueType"/>
                        </choice>
                        <element name="stepDistance" type="sml:DistanceValueType" minOccurs="0">
                            <annotation>
                                <documentation>can be derived from numberOfSamples and direction attribute;
option to explicitly define here</documentation>
                            </annotation>
                        </element>
                        <element name="startTime" type="sml:TimeValueType" minOccurs="0">
                            <annotation>
                                <documentation>can be relative to local temporalCRS</documentation>
                            </annotation>
                        </element>
                        <element name="extentTime" type="sml:TimeValueType" minOccurs="0"/>
                        <element name="stepTime" type="sml:TimeValueType" minOccurs="0"/>
                        <element name="repeatTime" type="sml:TimeValueType" minOccurs="0"/>
                    </sequence>
                    <attribute name="alongAxis" fixed="radial"/>
                </restriction>
            </complexContent>
        </complexType>
    </element>
    <element name="FixedProfileAction" substitutionGroup="sml:_ScanAction">
        <annotation>
```

```
            <documentation>profile is assumed to be along the currently positioned look ray</documentation>
        </annotation>
        <complexType>
            <complexContent>
                <restriction base="sml:SimpleTranslationType">
                    <sequence>
                        <element name="fixedDistance" type="sml:DistanceValueType"/>
                    </sequence>
                    <attribute name="alongAxis" fixed="radial"/>
                </restriction>
            </complexContent>
        </complexType>
    </element>
    <!--
================================================================================================
======
                            Global complex types definitions
    ================================================================================================
============-->
    <!-- ==============   Scan Action Types
=================================================================-->
    <complexType name="_ScanActionType" abstract="true"/>
    <!-- ==============   SimpleRotation
=================================================================-->
    <complexType name="SimpleRotationType">
        <complexContent>
            <extension base="sml:_ScanActionType">
                <sequence>
                    <element name="fixedAngle" type="sml:RotationValueType"/>
                </sequence>
                <attribute name="aboutAxis" use="required">
                    <simpleType>
                        <restriction base="string">
                            <enumeration value="x"/>
                            <enumeration value="y"/>
                            <enumeration value="z"/>
                        </restriction>
                    </simpleType>
                </attribute>
                <attribute name="direction" use="optional" default="1">
                    <simpleType>
                        <restriction base="integer">
                            <enumeration value="-1"/>
                            <enumeration value="1"/>
                        </restriction>
                    </simpleType>
                </attribute>
            </extension>
        </complexContent>
    </complexType>
    <!-- =============   LinearRotation
=================================================================-->
    <complexType name="LinearRotationType">
        <complexContent>
            <extension base="sml:_ScanActionType">
                <sequence>
                    <element name="numberOfSamples" type="positiveInteger"/>
                    <element name="startAngle" type="sml:RotationValueType"/>
                    <choice>
                        <element name="stopAngle" type="sml:RotationValueType"/>
                        <element name="extentAngle" type="sml:RotationValueType"/>
                    </choice>
                    <element name="stepAngle" type="sml:RotationValueType"/>
                    <element name="startTime" type="sml:TimeValueType" minOccurs="0"/>
                    <element name="extentTime" type="sml:TimeValueType" minOccurs="0"/>
                    <element name="stepTime" type="sml:TimeValueType" minOccurs="0"/>
```

```
                <element name="repeatTime" type="sml:TimeValueType" minOccurs="0"/>
            </sequence>
            <attribute name="aboutAxis" use="required">
                <simpleType>
                    <restriction base="anyURI">
                        <enumeration value="x"/>
                        <enumeration value="y"/>
                        <enumeration value="z"/>
                    </restriction>
                </simpleType>
            </attribute>
            <attribute name="direction" use="optional" default="1">
                <simpleType>
                    <restriction base="integer">
                        <enumeration value="1"/>
                        <enumeration value="-1"/>
                    </restriction>
                </simpleType>
            </attribute>
        </extension>
    </complexContent>
</complexType>
<!-- ==============   SimpleTranslation
================================================================-->
<complexType name="SimpleTranslationType">
    <complexContent>
        <extension base="sml:_ScanActionType">
            <sequence>
                <element name="fixedDistance" type="sml:DistanceValueType"/>
            </sequence>
            <attribute name="alongAxis" use="required">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="x"/>
                        <enumeration value="y"/>
                        <enumeration value="z"/>
                        <enumeration value="radial"/>
                    </restriction>
                </simpleType>
            </attribute>
        </extension>
    </complexContent>
</complexType>
<!-- ==============   LinearTranslation
================================================================-->
<complexType name="LinearTranslationType">
    <complexContent>
        <extension base="sml:_ScanActionType">
            <sequence>
                <element name="numberOfSamples" type="positiveInteger"/>
                <element name="startDistance" type="sml:DistanceValueType"/>
                <choice>
                    <element name="stopDistance" type="sml:DistanceValueType"/>
                    <element name="extentDistance" type="sml:DistanceValueType"/>
                </choice>
                <element name="stepDistance" type="sml:DistanceValueType"/>
                <element name="startTime" type="sml:TimeValueType" minOccurs="0"/>
                <element name="extentTime" type="sml:TimeValueType" minOccurs="0"/>
                <element name="stepTime" type="sml:TimeValueType" minOccurs="0"/>
                <element name="repeatTime" type="sml:TimeValueType" minOccurs="0"/>
            </sequence>
            <attribute name="alongAxis" use="required">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="x"/>
                        <enumeration value="y"/>
```

```
                        <enumeration value="z"/>
                        <enumeration value="radial"/>
                    </restriction>
                </simpleType>
            </attribute>
        </extension>
    </complexContent>
</complexType>
<!-- ==============  Fundamental Value Types
=======================================================================-->
<!-- ==============  RotationValue
=======================================================================-->
<complexType name="RotationValueType">
    <simpleContent>
        <extension base="double">
            <attribute name="angleUnit" type="anyURI" use="required"/>
            <attribute name="basis" use="required">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="sampleCenter"/>
                        <enumeration value="sampleOuterEdge"/>
                        <enumeration value="sampleInnerEdge"/>
                    </restriction>
                </simpleType>
            </attribute>
        </extension>
    </simpleContent>
</complexType>
<!-- =============  TimeValue
=======================================================================-->
<complexType name="TimeValueType">
    <simpleContent>
        <extension base="double">
            <attribute name="timeUnit" type="anyURI" use="required"/>
            <attribute name="basis" use="required">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="sampleCenter"/>
                        <enumeration value="sampleOuterEdge"/>
                        <enumeration value="sampleInnerEdge"/>
                    </restriction>
                </simpleType>
            </attribute>
        </extension>
    </simpleContent>
</complexType>
<!-- =============  DistanceValue
=======================================================================-->
<complexType name="DistanceValueType">
    <simpleContent>
        <extension base="double">
            <attribute name="distanceUnit" type="anyURI" use="required"/>
            <attribute name="basis" use="required">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="sampleCenter"/>
                        <enumeration value="sampleOuterEdge"/>
                        <enumeration value="sampleInnerEdge"/>
                    </restriction>
                </simpleType>
            </attribute>
        </extension>
    </simpleContent>
</complexType>
<!-- ========================================================================
    Main scanner types
```

```
    ============================================================================== -->
    <!-- ==============   Fixed View
=================================================================-->
    <element name="FixedView" type="sml:FixedViewType" substitutionGroup="sml:_SensorModelParameters"/>
    <complexType name="FixedViewType">
        <sequence>
            <element name="do">
                <complexType>
                    <sequence>
                        <element ref="sml:FixedElevationAction"/>
                        <element ref="sml:FixedSweepAction"/>
                        <element ref="sml:FixedProfileAction"/>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>
    <!-- ==============   Profiler 1D
=================================================================-->
    <element name="Profiler1D" type="sml:Profiler1DType" substitutionGroup="sml:_SensorModelParameters"/>
    <complexType name="Profiler1DType">
        <sequence>
            <element name="do">
                <complexType>
                    <sequence>
                        <element ref="sml:FixedElevationAction" minOccurs="0"/>
                        <element ref="sml:FixedSweepAction" minOccurs="0"/>
                        <choice>
                            <element ref="sml:LinearProfileAction"/>
                            <element ref="sml:FixedProfileAction"/>
                        </choice>
                    </sequence>
                </complexType>
            </element>
        </sequence>
    </complexType>
    <!-- ==============   Profiler 2D
=================================================================-->
    <element name="Profiler2D" type="sml:Profiler2DType" substitutionGroup="sml:_SensorModelParameters"/>
    <complexType name="Profiler2DType">
        <sequence>
            <element name="do">
                <complexType>
                    <choice>
                        <sequence>
                            <annotation>
                                <documentation>sweep scan first</documentation>
                            </annotation>
                            <element ref="sml:FixedSweepAction"/>
                            <element name="do">
                                <complexType>
                                    <sequence>
                                        <element ref="sml:LinearElevationAction" maxOccurs="2"/>
                                        <element name="do">
                                            <complexType>
                                                <choice>
                                                    <element ref="sml:LinearProfileAction"/>
                                                    <element ref="sml:FixedProfileAction"/>
                                                </choice>
                                            </complexType>
                                        </element>
                                    </sequence>
                                </complexType>
                            </element>
                        </sequence>
                        <sequence>
```

```
                        <annotation>
                            <documentation>elevation scan first</documentation>
                        </annotation>
                        <element ref="sml:FixedElevationAction"/>
                        <element name="do">
                            <complexType>
                                <sequence>
                                    <element ref="sml:LinearSweepAction" maxOccurs="2"/>
                                    <element name="do">
                                        <complexType>
                                            <choice>
                                                <element ref="sml:LinearProfileAction"/>
                                                <element ref="sml:FixedProfileAction"/>
                                            </choice>
                                        </complexType>
                                    </element>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                </choice>
            </complexType>
        </element>
    </sequence>
</complexType>
<!-- ==============   Profiler 3D
============================================================-->
<element name="Profiler3D" type="sml:Profiler3DType" substitutionGroup="sml:_SensorModelParameters"/>
<complexType name="Profiler3DType">
    <sequence>
        <element name="do">
            <complexType>
                <choice>
                    <sequence>
                        <annotation>
                            <documentation>sweep scan first</documentation>
                        </annotation>
                        <element ref="sml:LinearSweepAction"/>
                        <element name="do">
                            <complexType>
                                <sequence>
                                    <element ref="sml:LinearElevationAction" maxOccurs="2"/>
                                    <element name="do">
                                        <complexType>
                                            <choice>
                                                <element ref="sml:LinearProfileAction"/>
                                                <element ref="sml:FixedProfileAction"/>
                                            </choice>
                                        </complexType>
                                    </element>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                    <sequence>
                        <annotation>
                            <documentation>elevation scan first</documentation>
                        </annotation>
                        <element ref="sml:LinearElevationAction"/>
                        <element name="do">
                            <complexType>
                                <sequence>
                                    <element ref="sml:LinearSweepAction" maxOccurs="2"/>
                                    <element name="do">
                                        <complexType>
                                            <choice>
```

```
                                        <element ref="sml:LinearProfileAction"/>
                                        <element ref="sml:FixedProfileAction"/>
                                    </choice>
                                </complexType>
                            </element>
                        </sequence>
                    </complexType>
                </element>
            </sequence>
        </choice>
    </complexType>
</element>
</sequence>
</complexType>
<!-- ==============  Scanner 1D
============================================================-->
<element name="Scanner1D" type="sml:Scanner1DType"
substitutionGroup="sml:_SensorModelParameters"/>
<complexType name="Scanner1DType">
    <sequence>
        <element name="do">
            <complexType>
                <choice>
                    <sequence>
                        <annotation>
                            <documentation>fixed elevation angle</documentation>
                        </annotation>
                        <element ref="sml:FixedElevationAction" minOccurs="0"/>
                        <element ref="sml:LinearSweepAction"/>
                    </sequence>
                    <sequence>
                        <annotation>
                            <documentation>fixed sweep angle</documentation>
                        </annotation>
                        <element ref="sml:FixedSweepAction" minOccurs="0"/>
                        <element ref="sml:LinearElevationAction"/>
                    </sequence>
                </choice>
            </complexType>
        </element>
    </sequence>
</complexType>
<!-- ==============  Scanner 2D
============================================================-->
<element name="Scanner2D" type="sml:Scanner2DType"
substitutionGroup="sml:_SensorModelParameters"/>
<complexType name="Scanner2DType">
    <sequence>
        <element name="do">
            <complexType>
                <choice>
                    <sequence>
                        <annotation>
                            <documentation>sweep scan first</documentation>
                        </annotation>
                        <element ref="sml:LinearElevationAction"/>
                        <element name="do">
                            <complexType>
                                <sequence>
                                    <element ref="sml:LinearSweepAction" maxOccurs="2"/>
                                </sequence>
                            </complexType>
                        </element>
                    </sequence>
                    <sequence>
                        <annotation>
```

```
                        <documentation>elevation scan first</documentation>
                    </annotation>
                    <element ref="sml:LinearSweepAction"/>
                    <element name="do">
                        <complexType>
                            <sequence>
                                <element ref="sml:LinearElevationAction" maxOccurs="2"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </choice>
        </complexType>
    </element>
  </sequence>
 </complexType>
 <!-- ========================================================================
    scannerTransformation
 ======================================================================== -->
 <element name="scannerModel" type="sml:scannerModelType"
substitutionGroup="gml:_CoordinateOperation"/>
 <complexType name="scannerModelType">
  <complexContent>
   <extension base="gml:AbstractCoordinateOperationType">
    <sequence>
     <element name="usesMethod">
      <complexType>
       <sequence>
        <element name="scannerMethod" type="gml:OperationMethodType"/>
       </sequence>
      </complexType>
     </element>
     <element name="usesParameters">
      <complexType>
       <choice>
        <element ref="sml:scannerProperties"/>
        <element ref="sml:FixedView"/>
        <element ref="sml:Profiler1D"/>
        <element ref="sml:Profiler2D"/>
        <element ref="sml:Profiler3D"/>
        <element ref="sml:Scanner1D"/>
        <element ref="sml:Scanner2D"/>
       </choice>
      </complexType>
     </element>
    </sequence>
   </extension>
  </complexContent>
 </complexType>
</schema>
```

A.8 RPCMODEL.XSD

Provides base and derived properties for the Rapid Positioning Coordinates (RPC) sensor model, as well as defines *RpcModel*.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (NSSTC) -->
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Michael E Botts (University of Alabama in Huntsville) -
->
<schema targetNamespace="http://www.opengis.net/sensorML" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:sml="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
elementFormDefault="qualified" attributeFormDefault="unqualified">
 <annotation>
```

```
        <appinfo>rpcModel.xsd v0.7 2002-12-20</appinfo>
        <documentation>SensorML schema for supporting a sensor model based on Rapid Positioning
Coefficients (RPC)</documentation>
    </annotation>
    <!-- ================================================================================
       includes and imports
    ================================================================================ -->
    <include schemaLocation="../sensorMLBase.xsd"/>
    <import namespace="http://www.opengis.net/gml" schemaLocation="../../../gml/3.0/base/operations.xsd"/>
    <!--
==========================================================================================
                                Coefficient
    ==========================================================================================-->
    <element name="Coefficient">
        <annotation>
            <documentation>Each Coefficient must have a number attribute with value from 0 to 19,
inclusive</documentation>
        </annotation>
        <complexType>
            <simpleContent>
                <extension base="double">
                    <attribute name="number" use="required">
                        <simpleType>
                            <restriction base="integer"/>
                        </simpleType>
                    </attribute>
                </extension>
            </simpleContent>
        </complexType>
    </element>
    <!-- ================================================================================
       RPC Properties
       Supports Rapid Positioning Coordinates Model
    ================================================================================ -->
    <element name="rpcProperties" type="sml:rpcPropertiesType"
substitutionGroup="sml:_SensorModelParameters">
        <annotation>
            <documentation>Provides parameters to support Rapid Positioning Coordinates</documentation>
        </annotation>
    </element>
    <complexType name="rpcPropertiesType">
        <complexContent>
            <extension base="sml:_SensorModelParametersType">
                <sequence>
                    <element name="imageXScale">
                        <complexType>
                            <simpleContent>
                                <extension base="double"/>
                            </simpleContent>
                        </complexType>
                    </element>
                    <element name="ImageYScale">
                        <complexType>
                            <simpleContent>
                                <extension base="double"/>
                            </simpleContent>
                        </complexType>
                    </element>
                    <element name="TargetXScale" type="double">
                        <annotation>
                            <documentation>e.g. longitude scale</documentation>
                        </annotation>
                    </element>
                    <element name="TargetYScale" type="double">
                        <annotation>
                            <documentation>e.g. latitude scale</documentation>
```

```
                    </annotation>
                </element>
                <element name="ImageXOffset" type="double"/>
                <element name="ImageYOffset" type="double"/>
                <element name="TargetXOffset" type="double">
                    <annotation>
                        <documentation>e.g. longitude offset</documentation>
                    </annotation>
                </element>
                <element name="TargetYOffset" type="double">
                    <annotation>
                        <documentation>e.g. latitude offset</documentation>
                    </annotation>
                </element>
                <element name="TargetZOffset" type="double">
                    <annotation>
                        <documentation>Target Height</documentation>
                    </annotation>
                </element>
                <element name="XNumeratorCoefficients">
                    <complexType>
                        <sequence>
                            <element ref="sml:Coefficient" minOccurs="0" maxOccurs="20"/>
                        </sequence>
                    </complexType>
                </element>
                <element name="XDenominatorCoefficients">
                    <complexType>
                        <sequence>
                            <element ref="sml:Coefficient" minOccurs="0" maxOccurs="20"/>
                        </sequence>
                    </complexType>
                </element>
                <element name="YNumeratorCoefficients">
                    <complexType>
                        <sequence>
                            <element ref="sml:Coefficient" minOccurs="0" maxOccurs="20"/>
                        </sequence>
                    </complexType>
                </element>
                <element name="YDenominatorCoefficients">
                    <complexType>
                        <sequence>
                            <element ref="sml:Coefficient" minOccurs="0" maxOccurs="20"/>
                        </sequence>
                    </complexType>
                </element>
            </sequence>
        </extension>
    </complexContent>
</complexType>
<!-- ================================================================
     rpcTransformation
     ================================================================ -->
<element name="rpcMethod" type="gml:OperationMethodType" substitutionGroup="gml:OperationMethod"/>
</schema>
```

A.9 OPTICALMODEL.XSD

Provides base and derived properties for the optical sensor model, as well as defines *OpticalModel*. This schema will be greatly revised in future releases.

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

```
<!-- edited with XML Spy v4.3 U (http://www.xmlspy.com) by Michael E Botts (University of Alabama in Huntsville) -->
<schema targetNamespace="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml" xmlns:sml="http://www.opengis.net/sensorML" xmlns="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <annotation>
        <appinfo>opticalModel.xsd v0.7 2002-12-20</appinfo>
        <documentation>SensorML schema for describing the sensor model for optical cameras</documentation>
    </annotation>
    <!-- ================================================================================
        includes and imports
    ================================================================================ -->
    <include schemaLocation="../sensorMLBase.xsd"/>
    <import namespace="http://www.opengis.net/gml" schemaLocation="../../../gml/3.0/base/operations.xsd"/>
    <!-- ================================================================================
        global elements
    ================================================================================ -->
    <element name="opticalProperties" type="sml:opticalPropertiesType" substitutionGroup="sml:_SensorModelParameters"/>
    <!-- replace type with specific opticalMethod definition -->
    <element name="opticalMethod" type="gml:OperationMethodType" substitutionGroup="gml:OperationMethod"/>
    <!-- ================================================================================
        Registration Mark
    ================================================================================ -->
    <complexType name="RegistrationMarkType">
        <annotation>
            <documentation>RegistrationMark can describe a fidicial or a reseau mark</documentation>
        </annotation>
        <sequence>
            <element name="description" type="string" minOccurs="0"/>
            <element name="location">
                <complexType>
                    <sequence>
                        <element name="X" type="decimal"/>
                        <element name="Y" type="decimal"/>
                    </sequence>
                    <attribute name="crs" type="anyURI" use="required"/>
                </complexType>
            </element>
            <element name="stdDevX" type="decimal" minOccurs="0"/>
            <element name="stdDevY" type="decimal" minOccurs="0"/>
        </sequence>
        <attribute name="id" type="ID" use="required"/>
        <attribute name="uom" type="string" default="mm"/>
    </complexType>
    <!-- ================================================================================
        Point Measurement Type
    ================================================================================ -->
    <complexType name="PointMeasurementType">
        <sequence>
            <element name="X"/>
            <element name="Y"/>
        </sequence>
        <attribute name="uom"/>
    </complexType>
    <!-- ================================================================================
        Opical Properties
    ================================================================================ -->
    <complexType name="opticalPropertiesType">
        <complexContent>
            <extension base="sml:_SensorModelParametersType">
                <sequence>
                    <element name="FocalLength">
                        <complexType>
                            <simpleContent>
```

```
                                <extension base="decimal">
                                    <attribute name="unit"/>
                                </extension>
                            </simpleContent>
                        </complexType>
                    </element>
                    <element name="PrinPtBestSym" type="sml:PointMeasurementType" minOccurs="0">
                        <annotation>
                            <documentation>PointMeasurementType</documentation>
                        </annotation>
                    </element>
                    <element name="PrinPtAutoCol" type="sml:PointMeasurementType" minOccurs="0">
                        <annotation>
                            <documentation>PointMeasurementType</documentation>
                        </annotation>
                    </element>
                    <element name="RegistrationMarks" minOccurs="0">
                        <annotation>
                            <documentation>Fiducial and Reseau Marks</documentation>
                        </annotation>
                        <complexType>
                            <choice>
                                <element name="Fiducial" type="sml:RegistrationMarkType" minOccurs="4"
maxOccurs="8"/>
                                <element name="Reseau" type="sml:RegistrationMarkType" minOccurs="4"
maxOccurs="unbounded"/>
                            </choice>
                        </complexType>
                    </element>
                    <element name="LensDistortions" minOccurs="0">
                        <complexType>
                            <sequence>
                                <element name="Distortion">
                                    <complexType>
                                        <sequence>
                                            <element name="Quadrant">
                                                <simpleType>
                                                    <restriction base="int">
                                                        <minInclusive value="1"/>
                                                        <maxInclusive value="4"/>
                                                    </restriction>
                                                </simpleType>
                                            </element>
                                            <element name="distance" type="decimal"/>
                                            <element name="distorsion" type="decimal"/>
                                        </sequence>
                                    </complexType>
                                </element>
                                <element name="TangentialDistP1" type="decimal"/>
                                <element name="TangentialDistP2" type="decimal"/>
                                <element name="KCoeff"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
                <attribute name="isLensDistortionEnabled" type="boolean" use="optional" default="false"/>
                <attribute name="isInverted" type="boolean" use="optional" default="false"/>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

A.10 OBJECTSTATE.XSD

Provides more complete descriptions for the state of a dynamic object than is currently supported in GML3. Supports position, orientation, velocity, acceleration, and the 2nd derivatives of these. This schema may be moved over into GML in future releases.

```
<?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) -
->
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (NSSTC) -->
<schema targetNamespace="http://www.opengis.net/sensorML" xmlns:sml="http://www.opengis.net/sensorML"
xmlns:gml="http://www.opengis.net/gml" xmlns="http://www.w3.org/2001/XMLSchema"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <annotation>
        <appinfo>objectState.xsd v0.7 2002-12-20</appinfo>
        <documentation>SensorML schema for describing complete state (location, orientation, etc.) of an
object</documentation>
    </annotation>
    <!-- ================================================================================
       includes and imports
    ================================================================================ -->
    <include schemaLocation="./mathUtility.xsd"/>
    <import namespace="http://www.opengis.net/gml"
schemaLocation="../../gml/3.0/base/coordinateReferenceSystems.xsd"/>
    <!--================================================================================
      ObjectState Element
    ================================================================================ -->
    <element name="ObjectState">
        <annotation>
            <documentation>mechanical state (position, orientation, velocity, acceleration) of an
object</documentation>
        </annotation>
        <complexType>
            <complexContent>
                <extension base="sml:ObjectStateType"/>
            </complexContent>
        </complexType>
    </element>
    <complexType name="ObjectStateType">
        <sequence>
            <element name="location">
                <complexType>
                    <choice>
                        <element ref="gml:Point"/>
                        <element ref="sml:LocationVector"/>
                    </choice>
                </complexType>
            </element>
            <element name="orientation" minOccurs="0">
                <complexType>
                    <choice>
                        <element ref="sml:OrientationVector"/>
                        <element ref="sml:RotationMatrix"/>
                    </choice>
                </complexType>
            </element>
            <element name="velocity" minOccurs="0">
                <complexType>
                    <choice>
                        <element ref="sml:VelocityVector"/>
                    </choice>
                </complexType>
            </element>
            <element name="angularVelocity" minOccurs="0">
```

```
                <complexType>
                    <choice>
                        <element ref="sml:AngularVelocityVector"/>
                        <element ref="sml:RotationMatrix"/>
                    </choice>
                </complexType>
            </element>
            <element name="acceleration" minOccurs="0">
                <complexType>
                    <choice>
                        <element ref="sml:AccelerationVector"/>
                    </choice>
                </complexType>
            </element>
            <element name="angularAcceleration" minOccurs="0">
                <complexType>
                    <choice>
                        <element ref="sml:AngularAccelerationVector"/>
                        <element ref="sml:RotationMatrix"/>
                    </choice>
                </complexType>
            </element>
        </sequence>
        <attribute name="crsReference" type="anyURI" use="optional"/>
    </complexType>
    <!--============================================================================
      ObjectStateCollection Element
    ============================================================================= -->
    <element name="ObjectStateCollection" type="sml:ObjectStateCollectionType"
substitutionGroup="sml:ObjectState"/>
    <complexType name="ObjectStateCollectionType">
        <complexContent>
            <extension base="sml:ObjectStateType">
                <sequence>
                    <element name="objectStateMember" maxOccurs="unbounded">
                        <complexType>
                            <sequence>
                                <element ref="sml:ObjectState" minOccurs="0"/>
                            </sequence>
                            <attributeGroup ref="gml:AssociationAttributeGroup"/>
                        </complexType>
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
</schema>
```

A.11 MATHUTILITY.XSD

Provides support for math types needed for complex transformations (e.g. vectors, matrices, etc.). This schema may be moved over into GML in future releases.

```
<?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) -
->
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (NSSTC) -->
<schema targetNamespace="http://www.opengis.net/sensorML" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" xmlns:sml="http://www.opengis.net/sensorML"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <annotation>
        <appinfo>sensor.xsd v0.7 2002-12-20</appinfo>
        <documentation>Vector and Matrix support useful for SensorML (may move to gml
namespace)</documentation>
```

```
</annotation>
<!-- ==========================================================================
   includes and imports
========================================================================== -->
<import namespace="http://www.opengis.net/gml" schemaLocation="../../gml/3.0/base/basicTypes.xsd"/>
<!-- ==========================================================================
  basic geometry types
========================================================================== -->
<element name="vectorValues" type="sml:vectorValuesType">
    <annotation>
        <documentation>list of values (doubles) with a unit</documentation>
    </annotation>
</element>
<complexType name="vectorValuesType">
    <simpleContent>
        <extension base="gml:doubleList">
            <attribute name="uom" type="anyURI" use="optional"/>
            <attribute name="order" use="optional">
                <simpleType>
                    <restriction base="string">
                        <enumeration value="rowCol"/>
                        <enumeration value="colRow"/>
                        <enumeration value="xyz"/>
                        <enumeration value="rightHandRule"/>
                    </restriction>
                </simpleType>
            </attribute>
        </extension>
    </simpleContent>
</complexType>
<element name="vectorElement">
    <annotation>
        <documentation>vector element value with its unit and corresponding axis</documentation>
    </annotation>
    <complexType>
        <simpleContent>
            <extension base="gml:MeasureType">
                <attribute name="axis" type="anyURI" use="optional"/>
            </extension>
        </simpleContent>
    </complexType>
</element>
<element name="Vector" type="sml:VectorType">
    <annotation>
        <documentation>basic vector</documentation>
    </annotation>
</element>
<complexType name="VectorType">
    <choice>
        <element ref="sml:vectorElement" maxOccurs="unbounded"/>
        <element ref="sml:vectorValues"/>
    </choice>
    <attribute name="dimension" type="positiveInteger" default="3"/>
    <attribute name="referenceCRS" type="anyURI" use="optional"/>
</complexType>
<element name="NormalizedVector" type="sml:NormalizedVectorType">
    <annotation>
        <documentation>normalized 3D vector (should be used to indicate a direction)</documentation>
    </annotation>
</element>
<complexType name="NormalizedVectorType">
    <complexContent>
        <extension base="sml:VectorType"/>
    </complexContent>
</complexType>
<element name="Matrix" type="sml:MatrixType">
```

```
        <annotation>
            <documentation>basic matrix</documentation>
        </annotation>
    </element>
    <complexType name="MatrixType">
        <choice>
            <element name="matrixCoefficients">
                <complexType>
                    <choice>
                        <element name="Values">
                            <complexType>
                                <simpleContent>
                                    <restriction base="sml:vectorValuesType"/>
                                </simpleContent>
                            </complexType>
                        </element>
                        <element name="rowValues" type="sml:vectorValuesType" maxOccurs="unbounded"/>
                        <element name="columnValues" type="sml:vectorValuesType"
maxOccurs="unbounded"/>
                    </choice>
                </complexType>
            </element>
        </choice>
        <attribute name="rows" type="positiveInteger" default="3"/>
        <attribute name="columns" type="positiveInteger" default="3"/>
        <attribute name="referenceCRS" type="anyURI" use="optional"/>
    </complexType>
    <element name="NormalizedMatrix" type="sml:NormalizedMatrixType">
        <annotation>
            <documentation>normalized matrix (can be described through euler angles)</documentation>
        </annotation>
    </element>
    <complexType name="NormalizedMatrixType">
        <complexContent>
            <extension base="sml:MatrixType">
                <choice>
                    <element name="eulerAngles">
                        <complexType>
                            <sequence>
                                <element name="phi" type="gml:MeasureType"/>
                                <element name="theta" type="gml:MeasureType"/>
                                <element name="psi" type="gml:MeasureType"/>
                            </sequence>
                        </complexType>
                    </element>
                </choice>
            </extension>
        </complexContent>
    </complexType>
    <!-- ==========================================================================
       objectState vectors and matrices definition
    ========================================================================== -->
    <element name="LocationVector" type="sml:LocationVectorType" substitutionGroup="sml:Vector"/>
    <complexType name="LocationVectorType">
        <complexContent>
            <extension base="sml:VectorType"/>
        </complexContent>
    </complexType>
    <element name="OrientationVector" type="sml:OrientationVectorType" substitutionGroup="sml:Vector"/>
    <complexType name="OrientationVectorType">
        <complexContent>
            <extension base="sml:VectorType"/>
        </complexContent>
    </complexType>
    <element name="VelocityVector" type="sml:VelocityVectorType" substitutionGroup="sml:Vector"/>
    <complexType name="VelocityVectorType">
```

```
        <complexContent>
            <extension base="sml:VectorType"/>
        </complexContent>
    </complexType>
    <element name="AngularVelocityVector" type="sml:AngularVelocityVectorType"
substitutionGroup="sml:Vector"/>
    <complexType name="AngularVelocityVectorType">
        <complexContent>
            <extension base="sml:VectorType"/>
        </complexContent>
    </complexType>
    <element name="AccelerationVector" type="sml:AccelerationVectorType" substitutionGroup="sml:Vector"/>
    <complexType name="AccelerationVectorType">
        <complexContent>
            <extension base="sml:VectorType"/>
        </complexContent>
    </complexType>
    <element name="AngularAccelerationVector" type="sml:AngularAccelerationVectorType"
substitutionGroup="sml:Vector"/>
    <complexType name="AngularAccelerationVectorType">
        <complexContent>
            <extension base="sml:VectorType"/>
        </complexContent>
    </complexType>
    <element name="RotationMatrix" type="sml:RotationMatrixType"/>
    <complexType name="RotationMatrixType">
        <complexContent>
            <extension base="sml:NormalizedMatrixType"/>
        </complexContent>
    </complexType>
</schema>
```

A.12 TRANSFORMATIONS.XSD

Provides support for transformations needed to support sensors. Currently includes DynamicAffineTransformation and OrbitalElementPropagation (utilized for defining satellite state). mathUtility.xsd

```
<?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) -
->
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (NSSTC) -->
<schema targetNamespace="http://www.opengis.net/sensorML" xmlns="http://www.w3.org/2001/XMLSchema"
xmlns:sml="http://www.opengis.net/sensorML" xmlns:gml="http://www.opengis.net/gml"
elementFormDefault="qualified" attributeFormDefault="unqualified">
    <annotation>
        <appinfo>sensor.xsd v0.7 2002-12-20</appinfo>
        <documentation>Collection of transformations useful for SensorML (may move to gml
namespace)</documentation>
    </annotation>
    <include schemaLocation="./sensorMLBase.xsd"/>
    <complexType name="_BaseTransformation">
        <complexContent>
            <extension base="gml:AbstractCoordinateOperationType"/>
        </complexContent>
    </complexType>
    <!-- ==========================================================================
        DynamicAffineTransformation
    ========================================================================== -->
    <element name="DynamicAffineTransformation" type="sml:DynamicAffineTransformationType"
substitutionGroup="gml:_CoordinateOperation"/>
    <complexType name="DynamicAffineTransformationType">
        <complexContent>
```

```
            <extension base="gml:AbstractCoordinateOperationType">
                <sequence>
                    <element name="usesMethod" type="gml:OperationMethodRefType"/>
                    <element name="usesParameters" minOccurs="0">
                        <complexType>
                            <sequence>
                                <element ref="sml:DynamicParameters" maxOccurs="unbounded"/>
                            </sequence>
                        </complexType>
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <element name="DynamicParameters" type="sml:DynamicParametersType"/>
    <complexType name="DynamicParametersType">
        <sequence>
            <element ref="gml:timeStamp"/>
            <element name="translation" maxOccurs="3">
                <complexType>
                    <simpleContent>
                        <extension base="gml:MeasureType">
                            <attribute name="alongAxis" type="anyURI" use="required"/>
                        </extension>
                    </simpleContent>
                </complexType>
            </element>
            <element name="rotation" minOccurs="0" maxOccurs="3">
                <complexType>
                    <simpleContent>
                        <extension base="gml:MeasureType">
                            <attribute name="aboutAxis" type="anyURI" use="required"/>
                        </extension>
                    </simpleContent>
                </complexType>
            </element>
        </sequence>
    </complexType>
    <!-- ==============================================================================
         OrbitalElementsTransformation
    ========================================================================== -->
    <element name="OrbitalElementsPropagation" type="sml:OrbitalElementsPropagationType"
substitutionGroup="gml:_CoordinateOperation"/>
    <complexType name="OrbitalElementsPropagationType">
        <complexContent>
            <extension base="gml:AbstractCoordinateOperationType">
                <sequence>
                    <element name="usesMethod" type="gml:OperationMethodRefType"/>
                    <element name="usesParameters">
                        <complexType>
                            <choice>
                                <element name="TwoLineElements" type="sml:TwoLineElementType"
maxOccurs="unbounded">
                                    <annotation>
                                        <documentation>pure two-line elements</documentation>
                                    </annotation>
                                </element>
                                <element name="AmsatElements" type="sml:AmsatElementType"
maxOccurs="unbounded"/>
                                <element name="OneLineElements" type="sml:OneLineElementType"
maxOccurs="unbounded">
                                    <annotation>
                                        <documentation>US Navy's abbreviated element set</documentation>
                                    </annotation>
                                </element>
                                <element name="NoradElements" type="sml:NoradElementType">
```

```
                                    <annotation>
                                        <documentation>attempts to break two-line elements into separable
pieces</documentation>
                                    </annotation>
                                </element>
                            </choice>
                        </complexType>
                    </element>
                </sequence>
            </extension>
        </complexContent>
    </complexType>
    <complexType name="TwoLineElementType">
        <choice>
            <sequence>
                <annotation>
                    <documentation>standard tle lines put into xml</documentation>
                </annotation>
                <element name="lineOne" type="string"/>
                <element name="lineTwo" type="string"/>
                <element name="lineThree" type="string"/>
            </sequence>
            <element name="tleFile" type="anyURI">
                <annotation>
                    <documentation>URI to standard tle file</documentation>
                </annotation>
            </element>
        </choice>
    </complexType>
    <complexType name="OneLineElementType">
        <choice>
            <element name="lineOne" type="string">
                <annotation>
                    <documentation>standard one line element</documentation>
                </annotation>
            </element>
            <element name="oneLineElementFile" type="anyURI">
                <annotation>
                    <documentation>uri to standard OneLineElement file</documentation>
                </annotation>
            </element>
        </choice>
    </complexType>
    <complexType name="AmsatElementType">
        <choice>
            <sequence>
                <annotation>
                    <documentation>AMSAT elements put into xml</documentation>
                </annotation>
                <element name="Satellite" type="string"/>
                <element name="CatalogNumber" type="int"/>
                <element name="EpochTime" type="float">
                    <annotation>
                        <documentation>two-digit year, followed by day-of-year, followed by a period, followed by
fractional day</documentation>
                    </annotation>
                </element>
                <element name="ElementSet" type="integer"/>
                <element name="Inclination">
                    <complexType>
                        <simpleContent>
                            <extension base="float">
                                <attribute name="uom" type="anyURI" default="#degrees"/>
                            </extension>
                        </simpleContent>
                    </complexType>
```

```
                </element>
                <element name="RAOfNode">
                    <complexType>
                        <simpleContent>
                            <extension base="float">
                                <attribute name="uom" type="anyURI" default="#degrees"/>
                            </extension>
                        </simpleContent>
                    </complexType>
                </element>
                <element name="Eccentricity" type="float"/>
                <element name="ArgOfPerigee">
                    <complexType>
                        <simpleContent>
                            <extension base="float">
                                <attribute name="uom" type="anyURI" default="#degrees"/>
                            </extension>
                        </simpleContent>
                    </complexType>
                </element>
                <element name="MeanAnomaly">
                    <complexType>
                        <simpleContent>
                            <extension base="float">
                                <attribute name="uom" type="anyURI" default="#degrees"/>
                            </extension>
                        </simpleContent>
                    </complexType>
                </element>
                <element name="MeanMotion">
                    <complexType>
                        <simpleContent>
                            <extension base="float">
                                <attribute name="uom" type="anyURI" default="#revPerDay"/>
                            </extension>
                        </simpleContent>
                    </complexType>
                </element>
                <element name="DecayRate">
                    <complexType>
                        <simpleContent>
                            <extension base="float">
                                <attribute name="uom" type="anyURI" default="#revPerDay^2"/>
                            </extension>
                        </simpleContent>
                    </complexType>
                </element>
                <element name="EpochRev" type="integer"/>
            </sequence>
            <element name="amsatFile" type="anyURI">
                <annotation>
                    <documentation>URI to a standard AMSAT file</documentation>
                </annotation>
            </element>
        </choice>
    </complexType>
    <complexType name="NoradElementType">
        <sequence>
            <element name="SatelliteName" type="string"/>
            <element name="SatelliteNumber" type="integer"/>
            <element name="Classification">
                <simpleType>
                    <restriction base="string">
                        <length value="1"/>
                    </restriction>
                </simpleType>
```

```
            </element>
            <element name="InternationalDesignator" type="string" minOccurs="0"/>
            <choice>
                <element name="EpochTime" type="float">
                    <annotation>
                        <documentation>two-digit year, followed by day-of-year, followed by a period, followed by
fractional day</documentation>
                    </annotation>
                </element>
                <element name="Time" type="dateTime"/>
            </choice>
            <choice>
                <element name="FirstDerivMeanMotion" type="float"/>
                <element name="BallisticCoefficient" type="float"/>
            </choice>
            <element name="SecondDerivMeanMotion" type="float" minOccurs="0"/>
            <choice>
                <element name="BstarDrag" type="float"/>
                <element name="RadPressCoefficient" type="float"/>
            </choice>
            <element name="EphemerisType">
                <simpleType>
                    <restriction base="integer">
                        <totalDigits value="1"/>
                    </restriction>
                </simpleType>
            </element>
            <element name="ElementNumber" type="integer"/>
            <element name="Inclination">
                <complexType>
                    <simpleContent>
                        <extension base="float">
                            <attribute name="uom" type="anyURI" default="#degrees"/>
                        </extension>
                    </simpleContent>
                </complexType>
            </element>
            <element name="RightAscensionOfNode">
                <complexType>
                    <simpleContent>
                        <extension base="float">
                            <attribute name="uom" type="anyURI" default="#degrees"/>
                        </extension>
                    </simpleContent>
                </complexType>
            </element>
            <element name="Eccentricity" type="float"/>
            <element name="ArgumentOfPerigee">
                <complexType>
                    <simpleContent>
                        <extension base="float">
                            <attribute name="uom" type="anyURI" default="#degrees"/>
                        </extension>
                    </simpleContent>
                </complexType>
            </element>
            <element name="MeanAnomaly">
                <complexType>
                    <simpleContent>
                        <extension base="float">
                            <attribute name="uom" type="anyURI" default="#degrees"/>
                        </extension>
                    </simpleContent>
                </complexType>
            </element>
            <element name="MeanMotion">
```

```
                <complexType>
                    <simpleContent>
                        <extension base="float">
                            <attribute name="uom" type="anyURI" default="#revPerDay"/>
                        </extension>
                    </simpleContent>
                </complexType>
            </element>
            <element name="RevNumber" type="integer"/>
        </sequence>
    </complexType>
</schema>
```

A.13 SENSORGLOSSARY.XSD

Provides support for defining sensor glossaries to enable referencing of term, coordinate reference systems, and coordinate operations used in SensorML.

```
<?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/sensorML"
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:appinfo>sensorGlossary.xsd v0.7 2002-12-20</xs:appinfo>
        <xs:documentation>definitions for terms and coorindate reference systems for sensors
(SensorML)</xs:documentation>
    </xs:annotation>
    <!-- ==========================================================================
       includes and imports
    ========================================================================== -->
    <xs:include schemaLocation="./sensorMLBase.xsd"/>
    <xs:import namespace="http://www.isotc211.org/iso19115/"
schemaLocation="../../iso19115/0.1.21/iso19115.xsd"/>
    <xs:import namespace="http://www.opengis.net/gml"
schemaLocation="../../gml/3.0/base/coordinateReferenceSystems.xsd"/>
    <xs:import namespace="http://www.opengis.net/ows" schemaLocation="../../OM/0.1.20/observable.xsd"/>
    <!-- ==========================================================================
       Sensor Glossary Element
    ========================================================================== -->
    <xs:element name="SensorGlossary">
        <xs:annotation>
            <xs:documentation>Defines coordinate systems, coordinate operations, and terms for
sensors</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element ref="sml:Definition" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="gml:_CoordinateReferenceSystem" minOccurs="0" maxOccurs="unbounded"/>
                <xs:element ref="gml:_CoordinateOperation" minOccurs="0" maxOccurs="unbounded"/>
            </xs:sequence>
            <xs:attribute name="codeSpace" type="xs:anyURI" use="optional"/>
        </xs:complexType>
    </xs:element>
    <xs:element name="Definition">
        <xs:annotation>
            <xs:documentation>Provides for definition of terms</xs:documentation>
        </xs:annotation>
        <xs:complexType>
            <xs:sequence>
                <xs:element name="term" type="xs:string"/>
                <xs:element name="alias" type="gml:CodeType" minOccurs="0" maxOccurs="unbounded"/>
```

```
                <xs:element name="description" type="gml:StringOrRefType"/>
                <xs:element name="authority" type="sml:ResponsiblePartyPropertyType" minOccurs="0"/>
                <xs:element ref="sml:additionalInformation" minOccurs="0"/>
            </xs:sequence>
            <xs:attribute name="id" type="xs:ID" use="required"/>
            <xs:attribute name="crossReference" type="xs:anyURI" use="optional"/>
        </xs:complexType>
    </xs:element>
</xs:schema>
```

A.14 GMLSTUB.XSD

Allows import of multiple schemas from gml.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.opengis.net/gml" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:gml="http://www.opengis.net/gml" elementFormDefault="qualified" attributeFormDefault="unqualified">
    <xs:include schemaLocation="../../gml/3.0/base/temporal.xsd"/>
    <xs:include schemaLocation="../../gml/3.0/base/operations.xsd"/>
    <xs:include schemaLocation="../../gml/3.0/base/coordinateReferenceSystems.xsd"/>
    <xs:include schemaLocation="../../gml/3.0/base/dynamicFeature.xsd"/>
</xs:schema>
```

A.15 OWSSTUB.XSD

Allows import of multiple schemas from Observations and Measurements.

```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema targetNamespace="http://www.opengis.net/ows" xmlns:ows="http://www.opengis.net/ows"
xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified"
attributeFormDefault="unqualified">
    <xs:include schemaLocation="../../OM/0.1.20/observable.xsd"/>
    <xs:include schemaLocation="../../OM/0.1.20/typedValue.xsd"/>
</xs:schema>
```

Annex B: XML Examples for SensorML

A.1 Simple In-situ Sensor (minimal information)

This XML Instance document is a SensorML example of a wind speed sensor that was installed on February 3, 2002 and is still operational. It has a dynamic range from 0-134 mph, a threshold of 2.2 mph, an operational temperature range of -40 to 40 degrees Celsius, and a survivable windspeed range from 0-220 mph. It is located at longitude -85.8 degrees and latitude 35.4 degrees.

```
<?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:iso19115="http://www.isotc211.org/iso19115/"
xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://www.opengis.net/ows"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xmlns:xlink="http://www.w3.org/1999/xlink"
xsi:schemaLocation="http://www.opengis.net/sensorML ../sensor.xsd http://www.opengis.net/sensorML
../responseModels/generalResponse.xsd" id="YSI_WS_0001">
    <identifiedAs>
        <shortName>Wind Speed Sensor 0001</shortName>
        <longName>YSI Wind Speed Sensor at Station 0001</longName>
        <type>http://www.opengis.net/sensor#anerometer</type>
    </identifiedAs>
    <documentConstrainedBy>
        <validTime>
            <gml:TPeriod>
                <gml:begin>
                    <gml:TInstant>
                        <gml:tPosition>2002-02-03T12:09:00.00</gml:tPosition>
                    </gml:TInstant>
                </gml:begin>
                <gml:end>
                    <gml:TInstant>
                        <gml:tPosition indeterminatePosition="after"/>
                    </gml:TInstant>
                </gml:end>
            </gml:TPeriod>
        </validTime>
    </documentConstrainedBy>
    <measures>
        <Measurand>
            <name>Wind Speed</name>
            <ows:observable xlink:href="http://www.opengis.net/observables#windSpeed"/>
            <characterizedBy>
                <dynamicRange axis="http://www.opengis.net/observables#windSpeed">
                    <low uom="http://www.opengis.net/units#mph">0.0</low>
                    <high uom="http://www.opengis.net/units#mph">134.0</high>
                </dynamicRange>
                <threshold uom="http://www.opengis.net/units#mph"
axis="http://www.opengis.net/observables#windSpeed">2.2</threshold>
                <operationalRange axis="http://www.opengis.net/observables#temperature">
                    <low uom="http://www.opengis.net/units#celsius">-40.0</low>
                    <high uom="http://www.opengis.net/units#celsius">40.0</high>
                </operationalRange>
                <survivableRange axis="http://www.opengis.net/observables#windSpeed">
                    <low uom="http://www.opengis.net/units#mph">0.0</low>
                    <high uom="http://www.opengis.net/units#mph">220.0</high>
                </survivableRange>
            </characterizedBy>
            <basedOn>
```

```
                <Sample>
                    <locatedUsing>
                        <ObjectState>
                            <location>
                                <gml:Point>
                                    <gml:pos srsName="http://www.opengis.net/crs#epsg4326"
dimension="2">35.4 -85.8</gml:pos>
                                </gml:Point>
                            </location>
                        </ObjectState>
                    </locatedUsing>
                </Sample>
            </basedOn>
        </Measurand>
    </measures>
</Sensor>
```

A.2 SIMPLE IN-SITU SENSOR (MORE COMPLETE INFORMATION)

This SensorML example is a raingauge sensor. Many properties that are optional in SensorML have been included to provide a more complete description of the sensor.

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited with XMLSPY v5 U (http://www.xmlspy.com) by Alex Robin (NSSTC) -->
<Sensor id="YSI_RG_0001" documentDate="2002-10-25T14:00:00.000" documentVersion="0.2"
xmlns:iso19115="http://www.isotc211.org/iso19115/" xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml" xmlns:ows="http://www.opengis.net/ows"
xmlns="http://www.opengis.net/sensorML" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="
http://www.opengis.net/sensorML ..\sensor.xsd
http://www.opengis.net/sensorML ..\responseModels\generalResponse.xsd">
    <identifiedAs>
        <shortName>YSI 6215</shortName>
        <longName>YSI 6215 Rain Gauge withTipping Bucket Mechanism</longName>
        <type>http:/www.opengis.net/sensorML/sensors#RainGauge</type>
    </identifiedAs>
    <documentConstrainedBy>
        <validTime>
            <gml:TPeriod>
                <gml:description>This document is valid from the creation date to an unknow end
date</gml:description>
                <gml:begin>
                    <gml:TInstant>
                        <gml:tPosition>2002-10-25T14:00:00.000</gml:tPosition>
                    </gml:TInstant>
                </gml:begin>
                <gml:end>
                    <gml:TInstant>
                        <gml:tPosition indeterminatePosition="after"/>
                    </gml:TInstant>
                </gml:end>
            </gml:TPeriod>
        </validTime>
        <legalUse>
            <iso19115:accessConstraints>
                <iso19115:MD_RestrictionCode_CodeList>copyright</iso19115:MD_RestrictionCode_CodeList>
            </iso19115:accessConstraints>
            <iso19115:useConstraints>
                <iso19115:MD_RestrictionCode_CodeList>copyright</iso19115:MD_RestrictionCode_CodeList>
            </iso19115:useConstraints>
        </legalUse>
    </documentConstrainedBy>
```

```
    <locatedUsing>
        <ObjectState>
            <location>
                <LocationVector>
                    <vectorValues>0 0 0</vectorValues>
                </LocationVector>
            </location>
        </ObjectState>
    </locatedUsing>
    <measures>
        <Measurand>
            <name>Rainfall Amount     </name>
            <ows:observable>
                <ows:ObservableDefinition>
                    <gml:name>http://www.opengis.net/observables#rainfallAmount</gml:name>
                </ows:ObservableDefinition>
            </ows:observable>
            <characterizedBy>
                <relativeAccuracy axis="http://www.opengis.net/observables#rainfallAmount">
                    <condition>
                        <between axis="http://www.opengis.net/observables#rainfallRate">
                            <gml:low>
                                <gml:measure
uom="http://www.opengis.net/units#inchesPerHour">22</gml:measure>
                            </gml:low>
                            <gml:high>
                                <gml:measure
uom="http://www.opengis.net/units#inchesPerHour">22</gml:measure>
                            </gml:high>
                        </between>
                    </condition>
                    <low uom="http://www.opengis.net/units#none"
axis="http://www.opengis.net/observables#rainfallAmount">-0.01</low>
                    <high uom="http://www.opengis.net/units#none"
axis="http://www.opengis.net/observables#rainfallAmount">0.01</high>
                </relativeAccuracy>
                <resolution uom="http://www.opengis.net/units#in"
axis="http://www.opengis.net/observables#rainfallAmount">0.01</resolution>
                <capacity uom="http://www.opengis.net/units#inph"
axis="http://www.opengis.net/observables#rainfallRate">30</capacity>
                <operationalRange axis="http://www.opengis.net/observables#temperature">
                    <low uom="http://www.opengis.net/units#Farenheit"
axis="http://www.opengis.net/observables#temperature">32</low>
                    <high uom="http://www.opengis.net/units#Farenheit"
axis="http://www.opengis.net/observables#temperature">140</high>
                </operationalRange>
            </characterizedBy>
            <basedOn>
                <Sample/>
            </basedOn>
        </Measurand>
    </measures>
    <operatedBy>
        <responsibleParty>
            <iso19115:contactInfo>
                <iso19115:phone>
                    <iso19115:voice>1 (256) 961-7978</iso19115:voice>
                    <iso19115:facsimile>1 (256) 961-7979</iso19115:facsimile>
                </iso19115:phone>
                <iso19115:address>
                    <iso19115:deliveryPoint>320, Sparkman Dr.</iso19115:deliveryPoint>
                    <iso19115:city>HUNTSVILLE</iso19115:city>
                    <iso19115:postalCode>AL 35805</iso19115:postalCode>
                    <iso19115:country>U.S.A</iso19115:country>

    <iso19115:electronicMailAddress>jchamilton@nasa.msfc.gov</iso19115:electronicMailAddress>
```

```
                </iso19115:address>
                <iso19115:hoursOfService>9h - 17h</iso19115:hoursOfService>
            </iso19115:contactInfo>
            <iso19115:role>
                <iso19115:CI_RoleCode_CodeList>owner</iso19115:CI_RoleCode_CodeList>
            </iso19115:role>
            <iso19115:individualName>John C. Hamilton</iso19115:individualName>
            <iso19115:organisationName>NASA MSFC</iso19115:organisationName>
            <iso19115:positionName>Principal Research Scientist</iso19115:positionName>
        </responsibleParty>
    </operatedBy>
    <describedBy>
        <manufacturedBy>
            <iso19115:contactInfo>
                <iso19115:phone>
                    <iso19115:voice>1 (508) 748-0366</iso19115:voice>
                    <iso19115:facsimile>1 (508) 748-2543</iso19115:facsimile>
                </iso19115:phone>
                <iso19115:address>
                    <iso19115:deliveryPoint>13, Atlantis Dr.</iso19115:deliveryPoint>
                    <iso19115:city>MARION</iso19115:city>
                    <iso19115:administrativeArea>Factory Service Center</iso19115:administrativeArea>
                    <iso19115:postalCode>MA 02738</iso19115:postalCode>
                    <iso19115:country>U.S.A</iso19115:country>
                </iso19115:address>
            </iso19115:contactInfo>
            <iso19115:role>
                <iso19115:CI_RoleCode_CodeList>author</iso19115:CI_RoleCode_CodeList>
            </iso19115:role>
            <iso19115:organisationName>YSI Massachusetts</iso19115:organisationName>
        </manufacturedBy >
        <ModelNumber>YSI 6215</ModelNumber>
        <IdentificationNumber>6215</IdentificationNumber>
    </describedBy>
</Sensor>
```

References

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

[OGCAS7] 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 IPR*, OGC 02-027.

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

[SPS] Jeff Lansing (ed), Sensor Planning Service IPR, OGC 02-0xx. http://ip.opengis.org/cgi-bin/ip/ows1view.htm

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