Tutorial I

Using SensorML to describe a

Complete Weather Station

February 27th, 2006

Alexandre Robin
(robin@nsstc.uah.edu)

Earth System Science Center - NSSTC
University of Alabama in Huntsville (UAH)
HUNTSVILLE, AL 35899

Table of Contents

I. Scope 4

II. Overview 4

A. System Description 4
B. Logical Structure 5
C. Physical Structure 6

III. Description of the Detectors 8

A. Detector Metadata 8
B. Reference Frame 9
C. Inputs and Outputs 10
D. Response Parameters 11

IV. Description of the System (Weather Station) 16

A. Metadata 16
B. Reference Frame 19
C. Inputs and Outputs 19
D. Internal Processes (Detectors) 21
E. Internal Connections 22
F. Component Positions 23
G. System Interfaces 25

V. Conclusion 26

Appendix A: XML of Thermometer (Temperature Detector) 27

Appendix B: XML of Barometer (Pressure Detector) 29

Appendix C: XML of Anemometer (Wind Speed Detector) 31

Appendix D: XML of Wind Direction Detector 33

Appendix E: XML of Rain Gauge (Rain Fall Detector) 35

Appendix F: XML of Full Weather Station System 37

References 45

Important Definitions

- **Process Model** – Atomic processing block usually used within a more complex *Process Chain*. It is associated to a *Process Method* which defines the process interface as well as how to execute the model. It also precisely defines its own inputs, outputs and parameters.

- **Process Chain** – Composite processing block consisting of interconnected sub-processes, which can in turn be *Process Models* or *Process Chains*. A process chain also includes possible data sources as well as connections that explicitly link input and output signals of sub-processes together. It also precisely defines its own inputs, outputs and parameters.

- **Process Method** – Definition of the behavior and interface of a *Process Model*. It can be stored in a library so that it can be reused by different *Process Model* instances (by using 'xlink' mechanism). It essentially describes the process interface and algorithm, and can point the user to existing implementations.

- **Detector** – Atomic part of a composite *Measurement System* defining sampling and response characteristic of a simple detection device. A detector has only one input and one output, both being scalar quantities. More complex *Sensors* such as a frame camera which are composed of multiple detectors can be described as a detector group or array using a *System* or *Sensor*. In SensorML a detector is a particular type of Process Model.

- **System** – Composite model of a group or array of components, which can include detectors, actuators, or sub-systems. A System relates a *Process Chain* to the real world and therefore provides additional definitions regarding relative positions of its components and communication interfaces.

- **Measurement System** – Specific type of *System* involving primarily sampling devices and *Detectors*.

- **Sensor** – Specific type of *System* representing a complete *Sensor*. This could be for example a complete airborne scanner which includes several *Detectors* (one for each band).

I. Scope

This document is intended to guide new SensorML users through the process of creating a complete *System* description, by including detailed explanations for each section of the resulting document. SensorML XML encodings are the main focus of this tutorial but keep in mind that software will soon be available (one is under development at UAH) to facilitate the creation of SensorML documents by using a more intuitive user interface and removing the need for writing XML documents manually.

The measurement *System* described in this tutorial is simple enough to introduce new users to SensorML but yet provides enough information to illustrate the main features of SensorML and understand its basic concepts.

II. Overview

A. System Description

The *System* selected for this tutorial is a complete weather station inspired from an existing Davis Instruments product. This weather station measures ambient temperature, atmospheric pressure, wind speed, wind direction and rain fall amount. Such a *System* can be very accurately described in SensorML since the language allows both logical and physical aspects of the *System* to be accurately documented.

“Logical Structure” qualifies the fact that the *System* is composed of components that can be interconnected. This is really an abstract view of the *System* since logical components DO NOT have to match real distinct objects (though it matches most of the time). The logical aspect of a *System* is equivalent to a *Process Chain* that describes how real phenomena values (inputs) are converted to the *System* outputs, which usually are measured (or estimated) values.

“Physical Structure” qualifies the real world aspects of the *System*. The SensorML language itself only defines the relative positions of components and the communication interfaces. However, it provides means of referencing other documents in the metadata section. These documents could for example give a precise 3D drawing of the *System* or an electrical diagram.

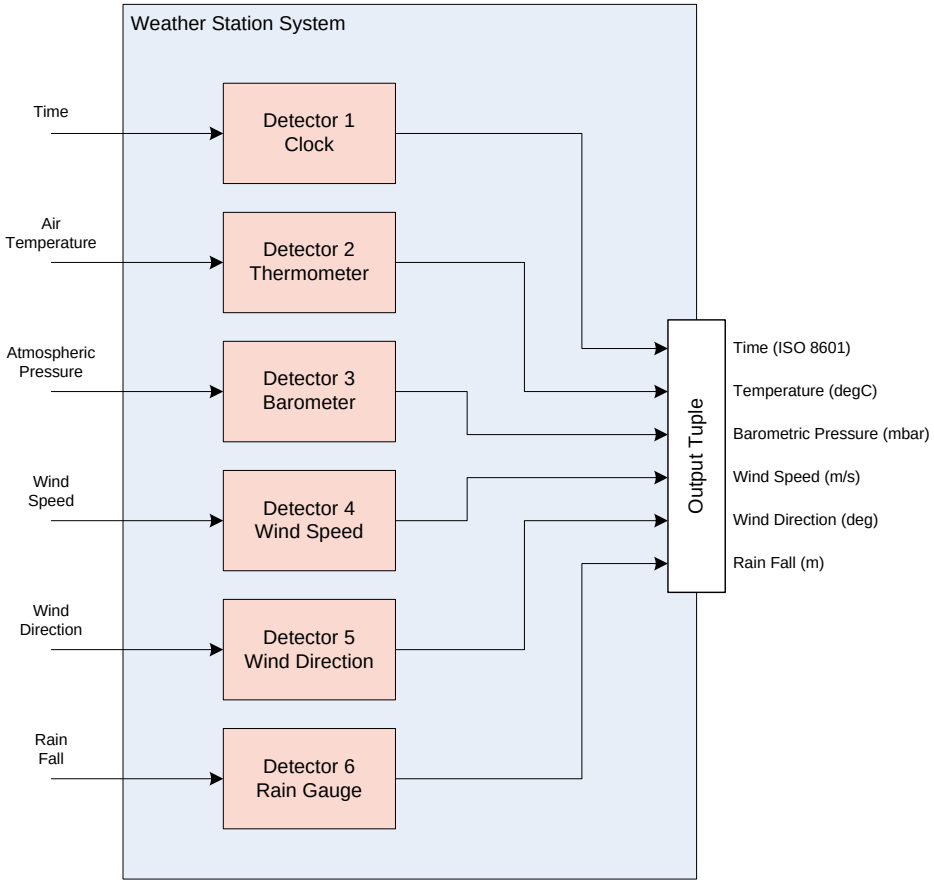
The following two sections describe the logical structure as well as the physical structure of the weather station, so that you can later relate this information to the content of the XML document.

B. Logical Structure

The Weather Station consists of a central monitor receiving signals from the various detectors connected to it. These detectors include:

- A thermometer (or temperature detector)
- A barometer (or barometric pressure detector)
- A wind sensor (wind speed and wind direction detectors)
- A rain gauge (or rain fall detector)

Davis measurement systems are modular so that all detectors can be purchased separately from the main monitor box. All *Detectors* are thus independent from the station itself except the barometer that is integrated into the monitor. The SensorML description will reflect this in a way, since all detectors are described as separate entities and then assembled and connected into a bigger *System* representing the whole station. The following figure shows the logical diagram of such a *System*:



Weather Station Logical Diagram

On this diagram, one can note the addition of a clock as one more detector. This is necessary to allow the precise description of the clock characteristics such as its accuracy. The second important point is that the wind sensor is actually described as two distinct detectors, one for specifying the wind speed measurement characteristics and one for the wind direction. This is necessary since these two detectors measures different quantities and can have different response characteristics. This diagram will be very useful when we will give more detail about SensorML encodings in the next sections. You will see that the structure of the XML document reflects exactly this diagram.

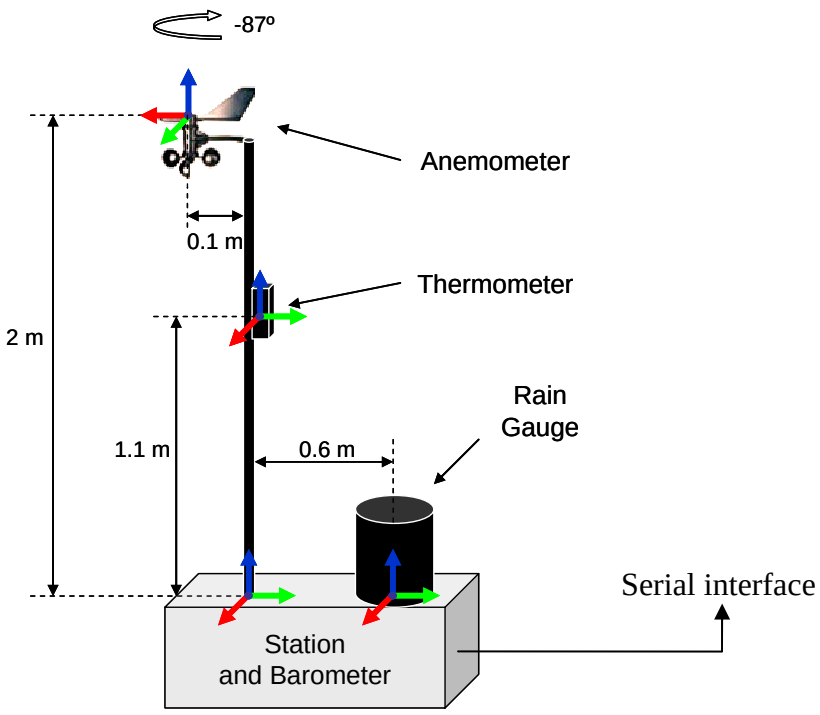
Note: This example is purposely kept simple, but more complex logical structures are of course allowed. System components can be fully interconnected and can incorporate digital processes for example. A System can also have several asynchronous outputs and can be used to describe large arrays of detectors using an indexing mechanism (This will be shown in a more advanced tutorial on scanner sensors used for earth imagery).

Until now, the *System* is equivalent to a process chain that is useful to understand how the output data was obtained from the measured phenomena. It thus represents the lineage of the output data. One could simulate this system behavior by applying input phenomenon values and artificially compute the output. In addition to this logical description, physical characteristics are also given in a SensorML document. This is covered in details in the next paragraph.

C. Physical Structure

In order to complete the description, SensorML can provide physical information about the *System* configuration. The information given as part of the XML document includes relative positions of components and *System* interfaces description. Positions can be given as fixed values if the components are not moving or as separate processes if they need to be computed according to some other variable (like time for instance).

Other characteristics such as 3D diagrams, wiring diagrams, electrical circuits, etc, can be referenced in the metadata section. The core SensorML specification doesn't enforce a standard format for these documents (Note that this could be the object of future extensions if required. Comments on this subject are welcomed). The following figure illustrates the configuration of the station used in the example.



Weather Station Physical Diagram

You can see on the figure that the various components are located in different places and SensorML allows you to capture these positions (location + orientation) with as much precision as needed. For a weather station, this information may seem superfluous but becomes important in the case of remote sensors or moving platforms. In this particular example the orientation of the wind direction sensor is of prior importance since it specifies what the wind direction is measured relative to. Typically the weather direction detector will have its reference direction pointing north so that its readings directly indicate the true heading of the wind direction (direction from which the wind is coming).

The following sections will show how to encode this information in a SensorML document. The individual detectors will be described first and then wrapped into a *System* description which will add positions and interface descriptions.

III. Description of the Detectors

Each detector can be described separately with its own input phenomenon, output quantity and set of response parameters. The response parameters include parameters for both the sampling and conversion stages and these can be described using the *Detector* model given as an informative appendix in the SensorML specification document. (This common detector model will become part of the core standard if it is judged generic enough to cover a wide variety of detector types). Only the thermometer will be described in details since all other detectors are similar. The complete XML descriptions of other detectors are available in appendix.

Here is the overall structure of a Detector XML element:

```
<Detector id="DETECTOR_ID">
  <identification> ... </identification>
  <classification> ... </classification>

          ... MORE METADATA ...

  <referenceFrame> ... </referenceFrame>
  <inputs> ... </inputs>
  <outputs> ... </outputs>
  <parameters> ... </parameters>
  <method xlink:href="urn:ogc:def:process:1.0:detector"/>
</Detector>
```

A. Detector Metadata

The first section of the thermometer detector is metadata information. Only identifiers are used for the thermometer detector (more possibilities with metadata will be shown at the *System* level). This is what the identification section looks like:

```
<identification>
  <IdentifierList>
    <identifier name="longName">
      <Term qualifier="urn:ogc:def:identifier:longName">
         Davis Temperature Detector
      </Term>
    </identifier>
    <identifier name="modelNumber">
      <Term qualifier="urn:ogc:def:identifier:modelNumber">
         7817
      </Term>
    </identifier>
  </IdentifierList>
</identification>
```

The `<identification>` section allows one to provide a list of identifiers for the object containing the tag. Identifiers can be of various kinds and are all soft typed (This means that in order to know the real nature of the identifier, the software will need to understand the qualifier URI). Each `<identifier>` element has a `name` attribute that can be used for display purposes and contains a `<Term>` child element. The `<Term>` provides

a `qualifier` attribute specifying the type of identifier and encapsulates a token giving the value of the identifier.

Note: URNs will appear often within this document as a way of specifying qualifiers and definitions of parameters. These URNs are intended to be resolvable to an actual XML object providing more detailed information about the definition itself. They typically point to dictionary entries such as CRS, phenomena or term definitions. OGC is still in the process of defining URN schemes for the OGC namespace but also as guidelines for other organizations namespace. There is also ongoing work for creating URN resolution mechanisms. Consequently, all URNs specified in this document are given as non normative examples and can change without notice, even though they follow recent OGC guidelines.

B. Reference Frame

The next section defines the reference frame attached to the detector. This uses the GML `<EngineeringCRS>` object and documents how the reference frame relates to the real hardware. The `id` attribute of the CRS is especially of interest since it will be used when giving relative positions of *System* components. The XML encodings of the reference frame are shown below:

```
<referenceFrame>
  <gml:EngineeringCRS gml:id="THERMOMETER_FRAME">
    <gml:srsName>
       Temperature Detector Spatial Frame
    </gml:srsName>
    <gml:usesCS xlink:href="urn:ogc:def:cs:xyzFrame"/>
    <gml:usesEngineeringDatum>
      <gml:EngineeringDatum gml:id="THERMOMETER_DATUM">
        <gml:datumName>
           Temperature Detector Spatial Datum
        </gml:datumName>
        <gml:anchorPoint>
           Origin is situated at the connector/case junction.
           X,Y,Z are undetermined since orientation is not critical.
        </gml:anchorPoint>
      </gml:EngineeringDatum>
    </gml:usesEngineeringDatum>
  </gml:EngineeringCRS>
</referenceFrame>
```

This follows GML grammar and provides an `id` and a `name` for the frame as well as a precise description of the datum. The `<anchorPoint>` element defines textually the location of the frame origin and the axis orientations relative to the hardware. Note that the CS used in SensorML should always be "`urn:ogc:def:cs:xyzFrame`". SensorML handles position information in a slightly different way than GML, making it possible to define polar, cylindrical or spherical coordinates in this Cartesian XYZ frame by defining rotation angles around each axis. Thus there is no need for these different types of CS that GML allows.

C. Inputs and Outputs

Input and Output Signals are also rigorously defined for each detector. XML encodings of the input and the output of the thermometer detector are shown below:

```
<inputs>
  <InputList>
    <input name="temperature">
      <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
                    uom="urn:ogc:def:unit:celsius"/>
    </input>
  </InputList>
</inputs>
<outputs>
  <OutputList>
    <output name="measuredTemperature">
      <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
                    uom="urn:ogc:def:unit:celsius"/>
    </output>
  </OutputList>
</outputs>
```

The first important thing to mention is that the *Detector* model is restricted to one scalar input and one scalar output. More complex scenarios like detector groups and detector arrays are modeled using a *System* or *Sensor* (The `Sensor` element is equivalent to a `System` in SensorML) and eventually using an indexing mechanism when dealing with large detector arrays. This will be shown in a future tutorial on scanner sensors.

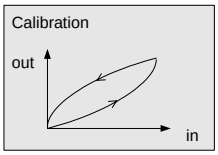
The input element has a scalar parameter element (here a `<Quantity>`) which is used to specify the type of the input using the `definition` attribute and the unit of measure using the `uom` attribute. These two attributes take a URI that points to a dictionary entry further defining the measured phenomenon and the unit respectively. The readings of this detector are converted by the station monitor to real temperature values expressed in degrees Celsius. Since SensorML allows one to choose the degree of details to incorporate in a *System* description, we chose here to hide the fact that the temperature detector actually provides a resistive value which is then converted to temperature by the station. The output phenomenon is thus identical to the input phenomenon, but we gave them different names for clarity.

D. Response Parameters

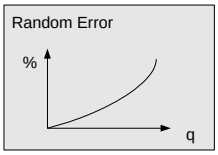
A Detector also provides a list of response parameters that are the core of the description. These parameters are provided using the following XML grammar:

```
<parameters>
  <ParameterList>
    <steadyStateResponse ... />
    <error ... />
    <impulseResponse ... />
    <frequencyResponse ... />
    <radiationFrequencyResponse ... />
    <temporalResponse ... />
    <spatialResponse ... />
 </ParameterList>
</parameters>
```

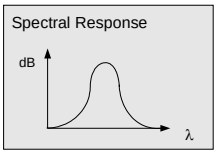
Most parameters are optional and can be provided only when applicable and needed.The definitions of these characteristics are summarized in the following table:



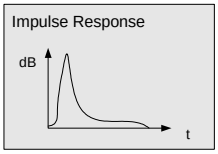
Calibration Curve
Gives the mapping of input to output values for a steady state regime. Two curves are used to describe a Hysteresis behavior.



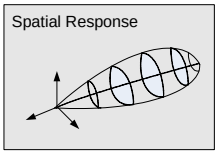
Random Error Curve
Gives the relative measurement error versus the input value itself or any other environmental quantity such as temperature.



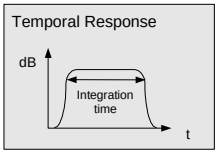
Spectral Response Curve
Specifies dynamic characteristics of the detector in the frequency domain. It gives the sensitivity of the detector versus the frequency or wavelength of the input signal.



Impulse Response Curve
Specifies dynamic characteristics of the detector in the time domain. It represents the normalized output of the detector for an impulse (Δ function) input.



Spatial Response Curve(s)
Gives the sensitivity of the detector relative to spatial coordinates (location of the source, or orientation of the incoming signal, e.g. point spread function, polarization)



Temporal Response Curve
Gives the sensitivity of the detector relative to a temporal coordinate frame (e.g. sampling time). This is a more descriptive form of the integration time.

All these parameters are conditional, which means that the curves are specified for a certain number of conditions such as a time of calibration, ambient temperature or pressure at the time of calibration, etc…

A certain number of scalar parameters are also available if these more general curves are not needed. It is possible to define measurement accuracy, ifov angles, integration time and latency time as simple values instead of using a complete curve.

Only the steady state response curve and the error curve will be provided in this thermometer example. They exactly reflect the data provided by the manufacturer. The encoding of the steady state response curve of the thermometer is shown below:

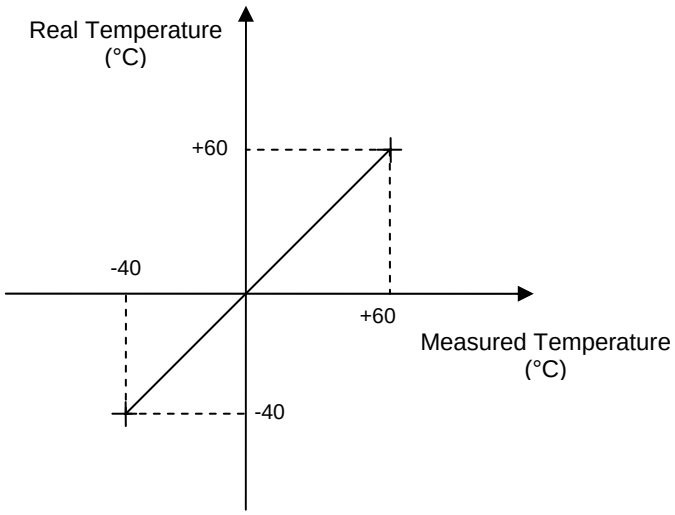
```
<steadyStateResponse>
  <ConditionalCurve fixed="true">
    <condition name="calibrationTime">
      <swe:Time definition="urn:ogc:def:phenomenon:time"
                        uom="urn:ogc:def:unit:iso8601">
         2004-01-01T04:30:00
      </swe:Time>
    </condition>
    <data>
      <swe:NormalizedCurve>
        <swe:function>
          <swe:Curve arraySize="2">
            <swe:definition>
              <swe:Coordinates>
                <swe:axis name="realTemperature">
                  <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
                                uom="urn:ogc:def:unit:celsius"/>
                </swe:axis>
                <swe:axis name="measuredTemperature">
                  <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
                                uom="urn:ogc:def:unit:celsius"/>
                </swe:axis>
              </swe:Coordinates>
            </swe:definition>
            <swe:tupleValues>-40,-40 60,60</swe:tupleValues>
          </swe:Curve>
        </swe:function>
      </swe:NormalizedCurve>
    </data>
  </ConditionalCurve>
</steadyStateResponse>
```

The whole steady state response parameter is given as a conditional curve. The first element is therefore a `<condition>` tag that allows one to specify various conditions for which the curve is valid. In this example, the calibration time is given and specifies the exact time at which this curve was obtained. This uses a `<Time>` element with the correct definition and the unit fixed to ISO8601 syntax (Note that ISO8601 is actually an encoding specification for time but also implies certain units for the different fields. This is why we chose to specify it as the unit). More than one condition can be specified as we will see for the error curve parameter that works in a similar fashion.

After all conditions are specified, the curve data can actually be described. The `<NormalizedCurve>` element will be discussed in details in another tutorial on scanner

sensors. It basically provides an interpolation method and bias and gains that can be applied to the whole curve, but these more complex features are not used here. The `<Curve>` element itself consists of axes definitions and values for all data points. The `<Coordinates>` section lists all curve axes using the scalar parameter elements `<Quantity>`, `<Time>`, `<Count>` and `<Category>`. Each `<axis>` element takes one of these parameters which further specifies the type of variable it represents using the `definition` attribute and its unit using the `uom` attribute.

In this example, we define a curve giving the measured temperature versus the real input temperature. This 2 dimensional curve would classically be represented by a figure similar to the one given below:



The `<tupleValues>` element is then used to give values for all data points that define the shape of the curve. In the case of the thermometer, the response is linear and thus requires only two data points at the beginning and end of the line. This is because a linear interpolation is assumed by default. The `<tupleValue>` thus includes two pairs of values where tokens in each tuple are given in the same order as the axes and separated by commas. Tuples themselves (each tuple corresponds to one data point with two coordinates in this case) are space separated. Values given in the XML snippet correspond to the coordinates shown on the figure. The mandatory `arraySize` attribute on the `<Curve>` element should also reflect the number of data points specified.

The second parameter is the error curve and works exactly the same as the steady state response: The error is also a conditional parameter and the XML grammar is identical. This parameter can express the known range of measurement error made by this detector. The error can be constant, in which case it can be specified as a single value using the `<accuracy>` parameter element. Both accuracy parameter and error curve can be expressed using either absolute (usually in the same unit as the measured phenomenon) or relative values (usually in percent of the output value). One can choose

one or the other by specifying the right URN as the `definition` attribute of the corresponding `<Quantity>` element.

The thermometer datasheet actually specified an error curve because the error is coupled to the ambient temperature. This is reflected in the SensorML description by using the `<error>` parameter rather than `<accuracy>`. The XML encoding for the error curve is shown below:

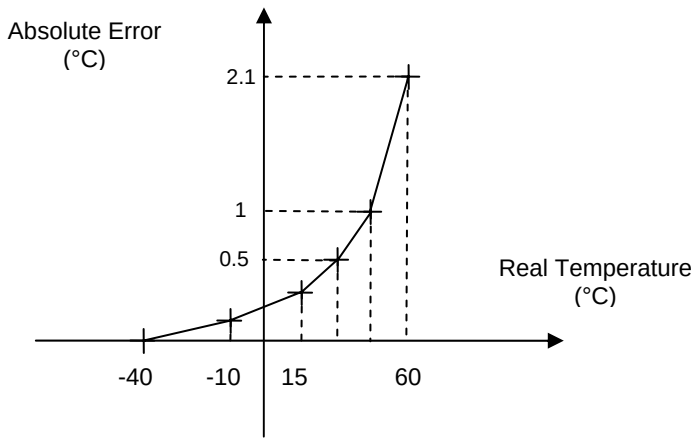
```
<error>
  <ConditionalCurve>
    <condition name="cableLength">
      <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                          uom=" urn:ogc:def:unit:meter">10</swe:Quantity>
    </condition>
    <data>
      <swe:NormalizedCurve>
        <swe:function>
          <swe:Curve arraySize="6">
            <swe:definition>
              <swe:Coordinates>
                <swe:axis name="realTemperature">
                  <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
                                       uom="urn:ogc:def:unit:celsius"/>
                </swe:axis>
                <swe:axis name="absoluteError">
                  <swe:Quantity definition="urn:ogc:def:phenomenon:absoluteError"
                                       uom="urn:ogc:def:unit:celsius"/>
                </swe:axis>
              </swe:Coordinates>
            </swe:definition>
            <swe:tupleValues>
                -40,0 -10,0.1 15,0.3 27,0.5 44,1 60,2.1
            </swe:tupleValues>
          </swe:Curve>
        </swe:function>
      </swe:NormalizedCurve>
    </data>
  </ConditionalCurve>
</error>
```

In this case the condition is the length of cable used to connect the thermometer to the monitor, since the measurement error also varies according to this parameter. The calibration was made with a cable length of 10 meters and that is what is specified using the `<Quantity>` within the `<condition>` element. Thus the type of the condition is a distance and the unit is the meter.

In this case, the curve maps real temperature to the measurement absolute error, therefore the corresponding axes are defined in the curve `<definition>` section. The first (dependent) axis is the real temperature expressed in degrees Celsius and the second (independent) axis is the maximum absolute error expressed in the same unit.

Another main difference with the curve used to describe steady state response is that this error curve is not linear. The `<tupleValues>` element thus requires more than two data points (coordinate tuples) as shown on the figure and in the XML. All data points drawn on the figure are reported as space separated tuples in the `<tupleValues>` element, and the `arraySize` attribute is adjusted to reflect the number of data points (here 6).

The error curve is shown on the following figure:



The last parameter that we can define for the thermometer is the latency time of the measurement process. This parameter expresses the delay between the instant at which the real phenomenon influences the detector and the time at which the measured value is available at the output. In the case of the thermometer, this value depends on the medium in which the detector is used.

```
<latencyTime>
  <ConditionalValue>
    <condition name="medium">
      <swe:Category definition="urn:ogc:def:category:medium">air</swe:Category>
    </condition>
    <data>
      <swe:Quantity definition="urn:ogc:def:phenomenon:duration"
                          uom="urn:ogc:def:unit:second">10</swe:Quantity>
    </data>
  </ConditionalValue>
</latencyTime>
```

The condition specified here is the type of medium. A `<Category>` element is used since a medium is an enumerable entity. The `definition` attribute further defines what type of category is concerned. (Note that there is no `uom` attribute on a Category). The value of the latency time is then given along with a compatible definition and unit.

Each `<Detector>` element shall also contain the method tag that is inherited from the `<Process>` SensorML object. The URN for the detector method is given below and should be used for all detectors described using this model.

```
<method xlink:href="urn:ogc:def:process:1.0:detector"/>
```

All other detectors in the weather station are described similarly. Complete XML encoding of each detector can be found in appendix.

IV. Description of the System (Weather Station)

The *System* represents the whole station in our example. The metadata section of the `<System>` element itself is identical to the one of the `<Detector>`, so that references, discoverable parameters and others can also be specified at this level if desired. The *System* also contains its own reference frame, a list of sub-processes as well as connections between them. Finally the physical part of the *System* defines positions of the various components (and eventually of the *System* itself) and available communication interfaces.

A. Metadata

In our example, the metadata section of the *System* is much more detailed that the one of each *Detector*, because the station as a whole is expected to be registered in a catalog. We first specify a list of identifiers with the following encoding:

```
<identification>
  <IdentifierList>
    <identifier name="longName">
      <Term qualifier="urn:ogc:def:identifier:longName">
             Davis Weather Monitor II Station</Term>
    </identifier>
    <identifier name="shortName">
      <Term qualifier="urn:ogc:def:identifier:shortName">
             Davis Weather Station</Term>
    </identifier>
    <identifier name="modelNumber">
      <Term qualifier="urn:ogc:def:identifier:modelNumber">
             7440</Term>
    </identifier>
    <identifier name="manufacturer">
      <Term qualifier="urn:ogc:def:identifier:manufacturer">
             Davis Instruments</Term>
    </identifier>
  </IdentifierList>
</identification>
```

Similarly to the *Detector* identifiers, each `<identifier>` element takes a `<Term>` that specifies the identifier type with the `qualifier` attribute and takes a text value of type `xs:token`. These identifiers will be useful for discovery of the *System* by using an OGC catalog service. These fields will be parsed and stored for quick access in the catalog database, so that the user can discover the *System* based on theirs values.

The *System* also incorporates classifiers that will be used by the catalog harvesting engine when registering the system. These classifiers give further information on the type of *System*, like the intended application and even the types of detectors it encapsulates.

The encoding of the classification section is as follows:

```
<classification>
  <ClassifierList>
    <classifier name="intendedApplication">
      <Term qualifier="urn:ogc:def:classifier:application">weather</Term>
    </classifier>
    <classifier name="sensorType">
      <Term qualifier="urn:ogc:def:classifier:sensorType">thermometer</Term>
    </classifier>
    <classifier name="sensorType">
      <Term qualifier="urn:ogc:def:classifier:sensorType">barometer</Term>
    </classifier>
    <classifier name="sensorType">
      <Term qualifier="urn:ogc:def:classifier:sensorType">anemometer</Term>
    </classifier>
    <classifier name="sensorType">
      <Term qualifier="urn:ogc:def:classifier:sensorType">rain gauge</Term>
    </classifier>
  </ClassifierList>
</classification>
```

This uses a grammar quasi identical to the identification section (‘identif’ is basically replaced by ‘classif’). Here again `qualifier` attributes are used to further define the type of classifiers used. Classifier type URNs are themselves expected to point to an entry in an online dictionary that textually describes the type of classifier.

The validity period of the description is then specified using the following encoding:

```
<validTime>
  <StartTime>2005-01-01</StartTime>
  <EndTime>currentTime</EndTime>
</validTime>
```

This illustrates the fact that SensorML can be used to capture the instantaneous configuration of a *System* (if the validity period is short or even a single time instant) as well as describe a *System* for a longer period, thus involving time varying information (if the period is larger).

The next metadata piece is the contact section. It allows the creator of the document to reference persons or organization responsible or connected to the System. For example, a contact can be a manufacturer, a vendor, an operator, an expert, a programmer, etc…

The encodings for a contact entry can be based either on the ISO19115 “Responsible Party” model which include plenty of useful information for getting in touch with the corresponding person, or on the simpler US Intelligence model of a Person. In this example, we list the Davis Instruments company as the manufacturer of the System.

The <ResponsibleParty> encodings for a contact are as follow:

```
<contact role="urn:ogc:def:identifier:manufacturer">
  <ResponsibleParty>
    <organizationName>Davis Instruments</organizationName>
    <contactInfo>
      <phone>
        <voice>+01-510-732-9229</voice>
        <facsimile>+01-510-732-9188</facsimile>
      </phone>
      <address>
        <deliveryPoint>3465, Diablo Avenue</deliveryPoint>
        <city>Hayward</city>
        <administrativeArea>CA</administrativeArea>
        <postalCode>94545-2778</postalCode>
        <country>USA</country>
        <electronicMailAddress>sales@davisnet.com</electronicMailAddress>
      </address>
    </contactInfo>
  </ResponsibleParty>
</contact>
```

It is also possible to list several documents that are related to the *System*. Here we point to the reference manual of the weather station edited by Davis Instruments and available on their website.

```
<documentation role="documents">
  <DocumentList>
    <member name="userManual">
      <Document>
        <description>
          <swe:Discussion>Davis Weather Monitor Manual</swe:Discussion>
        </description>
        <fileLocation
              xlink:href="htp://www.davisnet.com/support/manuals/7440.pdf"/>
      </Document>
    </member>
  </DocumentList>
</documentation>
```

As mentioned in a previous section, the document list is a good place to also reference technical information that cannot be encapsulated in the SensorML as of today. This especially includes electrical diagrams, wiring diagrams, electronic circuits, mechanical drawings, schematics, etc… These documents can be provided in any format in order to support existing documentation. It is clear that everybody would benefit from some kind of standardization at this level, and it will be a subject of discussion for future SensorML versions. We welcome any comments or suggestions on the subject.

B. Reference Frame

The reference frame of the System is a fundamental piece of the description since it will be referenced (using the id) when specifying the position of components relative to the *System* frame. The grammar is identical to the Detector reference frame since it also uses `<gml:EngineeringCRS>`. Details of encodings are shown below:

```
<referenceFrame>
  <gml:EngineeringCRS gml:id="STATION_FRAME">
    <gml:srsName>Weather Station Spatial Frame</gml:srsName>
    <gml:usesCS xlink:href="urn:ogc:def:cs:xyzFrame"/>
    <gml:usesEngineeringDatum>
      <gml:EngineeringDatum gml:id="STATION_DATUM">
        <gml:datumName>Weather Station Spatial Datum</gml:datumName>
        <gml:anchorPoint>
           Origin is at the base of the mounting.
           Z is along the axis of the mounting pole - typically vertical.
           X and Y are orthogonal to Z, along the short and long edges
           of the case respectively.
        </gml:anchorPoint>
      </gml:EngineeringDatum>
    </gml:usesEngineeringDatum>
  </gml:EngineeringCRS>
</referenceFrame>
```

Note that this reference frame also describes how the X, Y and Z axes are related to the hardware because the orientation of the station can be important. This is especially true for positioning the wind vane but also if a video camera is attached to the pole for example.

C. Inputs and Outputs

The *System* inputs and outputs are specified in a similar fashion to the ones of single *Detectors*. The main difference is that a System can of course have several inputs and outputs, and these are not restricted to scalar values. Each input and output can itself take a `<DataGroup>` or a `<DataArray>` element (used for sensor arrays, imagery sensors, etc…).

In our example, all variables measured by the System are grouped in a single output. This is done by using a `<DataGroup>` element which implicitly means that all weather measurement values are synchronously outputted with a time tag.

The encoding for inputs and outputs of the *System* are given below:

```
<inputs>
  <InputList>
    <input name="ambiantTemperature">
      <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"/>
    </input>
    <input name="atmosphericPressure">
      <swe:Quantity definition="urn:ogc:def:phenomenon:pressure"/>
    </input>
    <input name="windSpeed">
      <swe:Quantity definition="urn:ogc:def:phenomenon:windSpeed"/>
    </input>
    <input name="windDirection">
      <swe:Quantity definition="urn:ogc:def:phenomenon:windDirection"/>
    </input>
    <input name="rainFall">
      <swe:Quantity definition="urn:ogc:def:phenomenon:rainFall"/>
    </input>
  </InputList>
</inputs>

<outputs>
  <OutputList>
    <output name="weatherMeasurements">
      <swe:DataGroup>
        <swe:component name="time">
          <swe:Time definition="urn:ogc:def:phenomenon:time"
                          uom="urn:ogc:def:unit:iso8601"/>
        </swe:component>
        <swe:component name="temperature">
          <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
                              uom="urn:ogc:def:unit:celsius"/>
        </swe:component>
        <swe:component name="barometricPressure">
          <swe:Quantity definition="urn:ogc:def:phenomenon:pressure"
                              uom="urn:ogc:def:unit:bar" scale="1e-3"/>
        </swe:component>
        <swe:component name="windSpeed">
          <swe:Quantity definition="urn:ogc:def:phenomenon:windSpeed"
                              uom="urn:ogc:def:unit:meterPerSecond"/>
        </swe:component>
        <swe:component name="windDirection">
          <swe:Quantity definition="urn:ogc:def:phenomenon:windDirection"
                              uom="urn:ogc:def:unit:degree"/>
        </swe:component>
        <swe:component name="rainFall">
          <swe:Quantity definition="urn:ogc:def:phenomenon:rainfall"
                              uom="urn:ogc:def:unit:meter" scale="1e-3"/>
        </swe:component>
      </swe:DataGroup>
    </output>
  </OutputList>
</outputs>
```

Note that units of measure are not specified on the inputs since they are pure phenomena. It is specified on the output though since the values are expressed in a well defined unit.

D. *Internal Processes (Detectors)*

The next element of a *System* is the list of internal processes that the *System* encapsulates. This property is inherited from the fact that a *System* is derived from the *Process Chain* model. In the case of our weather station, the internal processes are all *Detectors* (which are a particular type of *Process Model*). This list of sub-processes is given using the `<ProcessList>` element as shown in the XML snippet below:

```
<processes>
  <ProcessList>
    <process name="clock"
             xlink:href="urn:vast:sensor:davisClock:1.0:001"/>
    <process name="thermometer"
             xlink:href="urn:vast:sensor:davisTemperature:1.0:001"/>
    <process name="barometer"
             xlink:href="urn:vast:sensor:davisPressure:1.0:001"/>
    <process name="anemometer"
             xlink:href="urn:vast:sensor:davisWindSpeed:1.0:001"/>
    <process name=" windDirectionDetector "
             xlink:href="urn:vast:sensor:davisWindDirection:1.0:001"/>
    <process name="rainGauge"
             xlink:href="urn:vast:sensor:davisRainFall:1.0:001"/>
  </ProcessList>
</processes>
```

In this case, the different sub-processes are simply referenced using the `xlink:href` attribute on the process property. This is done through the URN, so the client would need to ask a registry to get the actual XML description of each sub-component. Detectors could also be included inline by using the following syntax:

```
<processes>
  <ProcessList>
    <process name="clock">
      <Detector id="INTERNAL_CLOCK">
         ... DETECTOR DESCRIPTION AS DESCRIBED IN SECTION III ...
      </Detector>
    </process>
    <process name="thermometer">
      <Detector id="DAVIS_THERMOMETER">
         ... DETECTOR DESCRIPTION AS DESCRIBED IN SECTION III ...
      </Detector>
    </process>

      ... OTHER DETECTORS ...

  </ProcessList>
</processes>
```

Processes other than detectors, such as those for calculating wind chill and dew point can also be included in a System when needed. These processes would have their own set of inputs, outputs and parameters that could be linked within the System. This will be shown in a more advanced tutorial.

E. Internal Connections

Internal connections are used to describe how the *System* sub-processes are connected together (logically, not physically) and to the *System* inputs and outputs. In the case of the weather station, the connection pattern is very simple since *Detectors* don't interact with each other. We thus obtain a complete parallel connection pattern as shown on the figure on page 5.

There is a need for connecting each *System* input to the corresponding *Detector*, and connect each *Detector* output to the corresponding element of the *System* output tuple. This is done using the following XML:

```
<connections>
  <ConnectionList>
    <connection name="inputToThermometer">
      <Link>
        <source ref="this/inputs/ambiantTemperature"/>
        <destination ref="thermometer/inputs/temperature"/>
      </Link>
    </connection>
    <connection name="thermometerToOutput">
      <Link>
        <source ref="thermometer/outputs/measuredTemperature"/>
        <destination ref="this/outputs/weatherMeasurements/temperature"/>
      </Link>
    </connection>

          ... OTHER CONNECTIONS ...

  </ConnectionList>
</connections>
```

Each connection is listed separately, so in our example we have two connections for each *Detector*: One that goes from the *System* input to the *Detector* input, and one that goes from the *Detector* output to the *System* output.

You can see that source and destination of the connection are specified using the respective element with a `ref` attribute. This `ref` attribute uses a special token to point hierarchically to the process and then the given signal of the process (input or output). This token is a list of '/' separated names composed in this order of the name of the process, one of `inputs`, `outputs` or `parameters`, the name of the input, output or parameter and eventually the full name of the data component within this signal that needs to be connected.

The process name should be the one specified with the `name` attribute on the `<process>` property element or `this` if the connection should be made with the surrounding *System* or *Process Chain* itself.

The input, output or parameter name should correspond to the `name` attribute of the `<input>`, `<output>` or `<parameter>` element respectively, see a Detector description).

The full name of the component is the hierarchical list of component names (given by on each `<component>` element by the `name` attribute) to get to the desired component within a given input, output or parameter signal. This is needed because inputs, outputs and parameters can themselves be groups or arrays of elements. Using this syntax, it is then possible to reference any element of the signal structure, scalar or composite (a full group for instance). This list should also be '/' separated.

F. Component Positions

This section is used to define all *System* components' position and eventually the *System*'s position itself. Each position in the list can be expressed relative to any reference frame. In our example the position of the Weather Station is given in the EPSG4329 coordinate system and the position of each detector is given relative to the station's reference frame.

The following XML snippet shows how to encode the position of the station given in EPSG4329:

```
<position name="stationPosition">
  <swe:Position localFrame="#STATION_FRAME"
                referenceFrame="urn:ogc:def:crs:EPSG:4329">
    <swe:location>
      <swe:Location definition="urn:ogc:def:phenomenon:location">
        <swe:coordinate name="latitude">
          <swe:Quantity definition="urn:ogc:def:phenomenon:latitude"
                        uom="urn:ogc:def:unit:degree">34.72450</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="longitude">
          <swe:Quantity definition="urn:ogc:def:phenomenon:longitude"
                        uom="urn:ogc:def:unit:degree">-86.94533</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="altitude">
          <swe:Quantity definition="urn:ogc:def:phenomenon:altitude"
                        uom="urn:ogc:def:unit:meter">20.1169</swe:Quantity>
        </swe:coordinate>
      </swe:Location>
    </swe:location>
    <swe:orientation>
      <swe:Orientation definition="urn:ogc:def:phenomenon:orientation">
        <swe:coordinate name="trueHeading">
          <swe:Quantity definition="urn:ogc:def:phenomenon:angleToNorth"
                  axisCode="Z" uom="urn:ogc:def:unit:degree">87.0</swe:Quantity>
        </swe:coordinate>
      </swe:Orientation>
    </swe:orientation>
  </swe:Position>
</position>
```

The next snippet shows how the position of a detector (here the barometer) is specified relative to the station frame:

```
<position name="barometerPosition">
  <swe:Position localFrame="#BAROMETER_FRAME"
                referenceFrame="#STATION_FRAME">
    <swe:location>
      <swe:Location definition="urn:ogc:def:phenomenon:location">
        <swe:coordinate name="x">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                  axisCode="X" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="y">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                  axisCode="Y" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="z">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                  axisCode="Z" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
        </swe:coordinate>
      </swe:Location>
    </swe:location>
  </swe:Position>
</position>
```

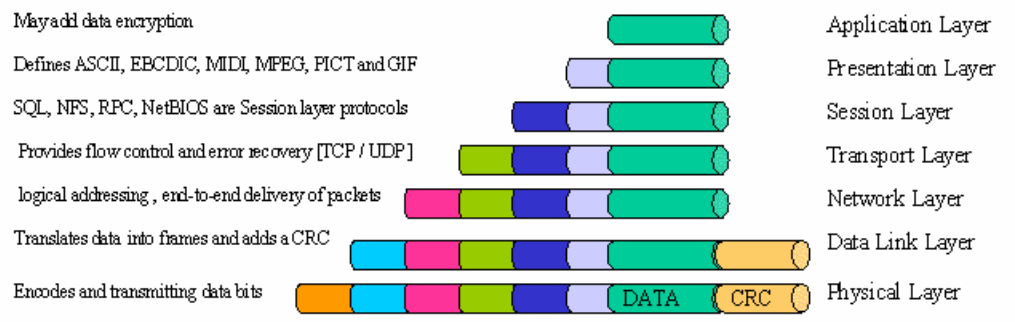
The position of the wind direction detector is slightly more complex since orientation is also specified:

```
<position name="windDirectionDetectorPosition">
  <swe:Position localFrame="#WIND_DIRECTION_DETECTOR_FRAME"
                referenceFrame="#STATION_FRAME">
    <swe:location>
      <swe:Location definition="urn:ogc:def:phenomenon:location">
        <swe:coordinate name="x">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                  axisCode="X" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="y">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                 axisCode="Y" uom="urn:ogc:def:unit:meter">-0.1</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="z">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                  axisCode="Z" uom="urn:ogc:def:unit:meter">2.0</swe:Quantity>
        </swe:coordinate>
      </swe:Location>
    </swe:location>
    <swe:orientation>
      <swe:Orientation definition="urn:ogc:def:phenomenon:orientation">
        <swe:coordinate name="z">
          <swe:Quantity definition="urn:ogc:def:phenomenon:angle"
                  axisCode="Z" uom="urn:ogc:def:unit:degree">-87.0</swe:Quantity>
        </swe:coordinate>
      </swe:Orientation>
    </swe:orientation>
  </swe:Position>
</position>
```

In the examples above, the id of the reference frames previously defined in the *System* and the *Detectors* is used to specify what frame we are giving the position of (using the `localFrame` attribute) and what frame it is located relative to (using the `referenceFrame` attribute).

G. System Interfaces

The last section of the System description is the list of communication interfaces that can be used to control or get data from the *System*. This is defined with the widely used OSI stack of communication protocols as show on the next figure:



In the weather station example, there is a serial (RS232) interface that can be used to communicate with the monitor. Here is the XML used to describe this interface:

```
<interface name="serial">
  <InterfaceDefinition>
    <!-- http://www.interfacebus.com/Design_Connector_RS232.html -->
    <applicationLayer>
      <Protocol definition="urn:davis:def:protocol:weatherLink"/>
    </applicationLayer>
    <physicalLayer>
      <Protocol definition="urn:ogc:def:protocol:RS232">
        <property name="num-bits">
          <swe:Count>8</swe:Count>
        </property>
        <property name="parity">
          <swe:Boolean>false</swe:Boolean>
        </property>
      </Protocol>
    </physicalLayer>
    <mechanicalLayer>
      <Connector definition="urn:ogc:def:connector:DB9">
        <property name="pin-out">
          <swe:Category definition="urn:ogc:def:pinout">EIA574</swe:Category>
        </property>
      </Connector>
    </mechanicalLayer>
  </InterfaceDefinition>
</interface>
```

Similarly to processes, each interface is given a name for further reference. The interface is defined by listing all applicable protocols (one for each layer used) as well as their parameters. Two additional layers are added to the classic OSI stack. The first one is called “mechanical layer” and allows one to define the type of connector used. The second one is called “service layer” (not used in this example) and is used to define a particular web service interface if needed.

V. Conclusion

This tutorial was a simple overview of the description of *Measurement Systems* with the SensorML language. The weather station described here allowed us to introduce basic SensorML concepts such as the *Detector* object and the composite *System* structure. You should know be familiar with the definition of inputs, outputs and parameters using the SWE common grammar, as well as the addition of metadata at different levels in the document.

The next tutorial will go through the description of a scanner sensor carried on board a satellite platform using the french satellite SPOT5 as an example. This will involve more complex concepts such as detector arrays and variable positions.

Appendix A: XML of Thermometer (Temperature Detector)

```
<Detector id="DAVIS_THERMOMETER">
  <identification>
    <IdentifierList>
      <identifier name="longName">
        <Term qualifier="urn:ogc:def:identifier:longName">
           Davis Temperature Detector</Term>
      </identifier>
      <identifier name="modelNumber">
        <Term qualifier="urn:ogc:def:identifier:modelNumber">7817</Term>
      </identifier>
    </IdentifierList>
  </identification>
  <!-- REFERENCE FRAME -->
  <referenceFrame>
    <gml:EngineeringCRS gml:id="THERMOMETER_FRAME">
      <gml:srsName>Temperature Detector Spatial Frame</gml:srsName>
      <gml:usesCS xlink:href="urn:ogc:def:cs:xyzFrame"/>
      <gml:usesEngineeringDatum>
        <gml:EngineeringDatum gml:id="THERMOMETER_DATUM">
          <gml:datumName>Temperature Detector Spatial Datum</gml:datumName>
          <gml:anchorPoint>
             Origin is situated at the connector/case junction.
             X,Y,Z are undetermined since orientation is not critical.
          </gml:anchorPoint>
        </gml:EngineeringDatum>
      </gml:usesEngineeringDatum>
    </gml:EngineeringCRS>
  </referenceFrame>
  <!-- INPUT/OUTPUT -->
  <inputs>
    <InputList>
      <input name="temperature">
        <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
                      uom="urn:ogc:def:unit:celsius"/>
      </input>
    </InputList>
  </inputs>
  <outputs>
    <OutputList>
      <output name="measuredTemperature">
        <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
                      uom="urn:ogc:def:unit:celsius"/>
      </output>
    </OutputList>
  </outputs>
  <!-- PARAMETERS -->
  <parameters>
    <ParameterList>
      <steadyStateResponse>
        <ConditionalCurve fixed="true">
          <condition name="calibrationTime">
            <swe:Time definition="urn:ogc:def:phenomenon:time"
                      uom="urn:ogc:def:unit:iso8601">2004-01-01T04:30:00
            </swe:Time>
          </condition>
          <data>
            <swe:NormalizedCurve>
              <swe:function>
                <swe:Curve arraySize="2">
                  <swe:definition>
                    <swe:Coordinates>
```

```
                    <swe:axis name="realTemperature">
                      <swe:Quantity
                         definition="urn:ogc:def:phenomenon:temperature"
                                uom="urn:ogc:def:unit:celsius"/>
                    </swe:axis>
                    <swe:axis name="measuredTemperature">
                      <swe:Quantity
                         definition="urn:ogc:def:phenomenon:temperature"
                                uom="urn:ogc:def:unit:celsius"/>
                    </swe:axis>
                  </swe:Coordinates>
                </swe:definition>
                <swe:tupleValues>-40,-40 60,60</swe:tupleValues>
              </swe:Curve>
            </swe:function>
          </swe:NormalizedCurve>
        </data>
      </ConditionalCurve>
    </steadyStateResponse>
    <error>
      <ConditionalCurve>
        <condition name="cableLength">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                        uom="urn:ogc:def:unit:meter">10</swe:Quantity>
        </condition>
        <data>
          <swe:NormalizedCurve>
            <swe:function>
              <swe:Curve arraySize="6">
                <swe:definition>
                  <swe:Coordinates>
                    <swe:axis name="realTemperature">
                      <swe:Quantity
                         definition="urn:ogc:def:phenomenon:temperature"
                                uom="urn:ogc:def:unit:celsius"/>
                    </swe:axis>
                    <swe:axis name="absoluteError">
                      <swe:Quantity
                         definition="urn:ogc:def:phenomenon:absoluteError"
                                uom="urn:ogc:def:unit:celsius"/>
                    </swe:axis>
                  </swe:Coordinates>
                </swe:definition>
                <swe:tupleValues>
                   -40,0 -10,0.1 15,0.3 27,0.5 44,1 60,2.1</swe:tupleValues>
              </swe:Curve>
            </swe:function>
          </swe:NormalizedCurve>
        </data>
      </ConditionalCurve>
    </error>
    <latencyTime>
      <ConditionalValue>
        <condition name="medium">
          <swe:Category definition="urn:ogc:def:category:medium">Air</swe:Category>
        </condition>
        <data>
          <swe:Quantity definition="urn:ogc:def:phenomenon:duration"
                        uom="urn:ogc:def:unit:second">10</swe:Quantity>
        </data>
      </ConditionalValue>
    </latencyTime>
  </ParameterList>
 </parameters>
 <!-- METHOD -->
 <method xlink:href="urn:ogc:def:process:1.0:detector"/>
</Detector>
```

Appendix B: XML of Barometer (Pressure Detector)

```
<Detector id="DAVIS_BAROMETER">
  <identification>
    <IdentifierList>
      <identifier name="longName">
        <Term qualifier="urn:ogc:def:identifier:longName">
           Davis Internal Barometer</Term>
      </identifier>
    </IdentifierList>
  </identification>
  <!-- REFERENCE FRAME -->
  <referenceFrame>
    <gml:EngineeringCRS gml:id="BAROMETER_FRAME">
      <gml:srsName>Barometric Detector Spatial Frame</gml:srsName>
      <gml:usesCS xlink:href="urn:ogc:def:cs:xyzFrame"/>
      <gml:usesEngineeringDatum>
        <gml:EngineeringDatum gml:id="BAROMETER_DATUM">
          <gml:datumName>Barometric Detector Spatial Datum</gml:datumName>
          <gml:anchorPoint>
              Origin is situated at the connector/case junction.
              X,Y,Z are undetermined since orientation is not critical.
          </gml:anchorPoint>
        </gml:EngineeringDatum>
      </gml:usesEngineeringDatum>
    </gml:EngineeringCRS>
  </referenceFrame>
  <!-- INPUT -->
  <inputs>
    <InputList>
      <input name="barometricPressure">
        <swe:Quantity definition="urn:ogc:def:phenomenon:pressure"
                      uom="urn:ogc:def:unit:bar" scale="1e-3"/>
      </input>
    </InputList>
  </inputs>
  <!-- OUTPUT -->
  <outputs>
    <OutputList>
      <output name="measuredBarometricPressure">
        <swe:Quantity definition="urn:ogc:def:phenomenon:pressure"
                      uom="urn:ogc:def:unit:bar" scale="1e-3"/>
      </output>
    </OutputList>
  </outputs>
  <!-- PARAMETERS -->
  <parameters>
    <ParameterList>
      <steadyStateResponse>
        <ConditionalCurve fixed="true">
          <condition name="calibrationTime">
            <swe:Time definition="urn:ogc:def:phenomenon:time"
                      uom="urn:ogc:def:unit:iso8601">2004-01-01T04:30:00
            </swe:Time>
          </condition>
          <condition name="temperature">
            <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
                          uom="urn:ogc:def:unit:celsius">20</swe:Quantity>
          </condition>
          <data>
            <swe:NormalizedCurve>
              <swe:function>
                <swe:Curve arraySize="2">
```

```
                    <swe:definition>
                      <swe:Coordinates>
                        <swe:axis name="atmosphericPressure">
                          <swe:Quantity definition="urn:ogc:def:phenomenon:pressure"
                                        uom="urn:ogc:def:unit:bar" scale="1e-3"/>
                        </swe:axis>
                        <swe:axis name="measuredBarometricPressure">
                          <swe:Quantity definition="urn:ogc:def:phenomenon:pressure"
                                        uom="urn:ogc:def:unit:bar" scale="1e-3"/>
                        </swe:axis>
                      </swe:Coordinates>
                    </swe:definition>
                    <swe:tupleValues>880.0,880.0 1080.0,1080.0</swe:tupleValues>
                  </swe:Curve>
                </swe:function>
              </swe:NormalizedCurve>
            </data>
          </ConditionalCurve>
        </steadyStateResponse>
      </ParameterList>
    </parameters>
    <!-- METHOD -->
    <method xlink:href="urn:ogc:def:process:1.0:detector"/>
  </Detector>
```

Appendix C: XML of Anemometer (Wind Speed Detector)

```
<Detector id="DAVIS_ANEMOMETER">
  <identification>
    <IdentifierList>
      <identifier name="longName">
        <Term qualifier="urn:ogc:def:identifier:longName">
           Davis Wind Speed Detector</Term>
      </identifier>
      <identifier name="modelNumber">
        <Term qualifier="urn:ogc:def:identifier:modelNumber">7911</Term>
      </identifier>
    </IdentifierList>
  </identification>
  <!-- REFERENCE FRAME -->
  <referenceFrame>
    <gml:EngineeringCRS gml:id="ANEMOMETER_FRAME">
      <gml:srsName>Wind Speed Detector Spatial Frame</gml:srsName>
      <gml:usesCS xlink:href="urn:ogc:def:cs:xyzFrame"/>
      <gml:usesEngineeringDatum>
        <gml:EngineeringDatum gml:id="ANEMOMETER_DATUM">
          <gml:datumName>Wind Speed Detector Spatial Datum</gml:datumName>
          <gml:anchorPoint>
             Origin is at the base of the rotating part.
             X,Y,Z axes are undefined since orientation is not needed.
          </gml:anchorPoint>
        </gml:EngineeringDatum>
      </gml:usesEngineeringDatum>
    </gml:EngineeringCRS>
  </referenceFrame>
  <!-- INPUT -->
  <inputs>
    <InputList>
      <input name="windSpeed">
        <swe:Quantity definition="urn:ogc:def:phenomenon:windSpeed"
                      uom="urn:ogc:def:unit:meterPerSecond"/>
      </input>
    </InputList>
  </inputs>
  <!-- OUTPUT -->
  <outputs>
    <OutputList>
      <output name="measuredWindSpeed">
        <swe:Quantity definition="urn:ogc:def:phenomenon:windSpeed"
                      uom="urn:ogc:def:unit:meterPerSecond"/>
      </output>
    </OutputList>
  </outputs>
  <!-- PARAMETERS -->
  <parameters>
    <ParameterList>
      <steadyStateResponse>
        <ConditionalCurve fixed="true">
          <condition name="calibrationTime">
            <swe:Time definition="urn:ogc:def:phenomenon:time"
                      uom="urn:ogc:def:unit:iso8601">2004-01-01T04:30:00
            </swe:Time>
          </condition>
          <data>
            <swe:NormalizedCurve>
              <swe:function>
                <swe:Curve arraySize="2">
                  <swe:definition>
```

```
                    <swe:Coordinates>
                      <swe:axis name="windSpeed">
                        <swe:Quantity definition="urn:ogc:def:phenomenon:windSpeed"
                                      uom="urn:ogc:def:unit:meterPerSecond"/>
                      </swe:axis>
                      <swe:axis name="measuredWindSpeed">
                        <swe:Quantity definition="urn:ogc:def:phenomenon:windSpeed"
                                      uom="urn:ogc:def:unit:meterPerSecond"/>
                      </swe:axis>
                    </swe:Coordinates>
                  </swe:definition>
                  <swe:tupleValues>0.9,0.9 78,78</swe:tupleValues>
                </swe:Curve>
              </swe:function>
            </swe:NormalizedCurve>
          </data>
        </ConditionalCurve>
      </steadyStateResponse>
    </ParameterList>
  </parameters>
  <!-- METHOD -->
  <method xlink:href="urn:ogc:def:process:1.0:detector"/>
</Detector>
```

Appendix D: XML of Wind Direction Detector

```
<Detector id="DAVIS_WIND_DIRECTION">
  <identification>
    <IdentifierList>
      <identifier name="longName">
        <Term qualifier="urn:ogc:def:identifier:longName">
           Davis Wind Direction Detector</Term>
      </identifier>
      <identifier name="modelNumber">
        <Term qualifier="urn:ogc:def:identifier:modelNumber">7911</Term>
      </identifier>
    </IdentifierList>
  </identification>
  <!-- REFERENCE FRAME -->
  <referenceFrame>
    <gml:EngineeringCRS gml:id="WIND_DIRECTION_DETECTOR_FRAME">
      <gml:srsName>Wind Direction Detector Spatial Frame</gml:srsName>
      <gml:usesCS xlink:href="urn:ogc:def:cs:xyzFrame"/>
      <gml:usesEngineeringDatum>
        <gml:EngineeringDatum gml:id="WIND_DIRECTION_DETECTOR_DATUM">
          <gml:datumName>Wind Direction Detector Spatial Datum</gml:datumName>
          <gml:anchorPoint>
             Origin is at the base of the rotating part.
             Z is along the axis around which measurement is made.
             X is the reference (0) direction that should point north when installed
             correctly. Y is orthogonal to X and Z.
          </gml:anchorPoint>
        </gml:EngineeringDatum>
      </gml:usesEngineeringDatum>
    </gml:EngineeringCRS>
  </referenceFrame>
  <!-- INPUTS -->
  <inputs>
    <InputList>
      <input name="windDirection">
        <swe:Quantity definition="urn:ogc:def:phenomenon:windDirection"
                      uom="urn:ogc:def:unit:degree"/>
      </input>
    </InputList>
  </inputs>
  <!-- OUTPUTS -->
  <outputs>
    <OutputList>
      <output name="measuredWindDirection">
        <swe:Quantity definition="urn:ogc:def:phenomenon:windDirection"
                      uom="urn:ogc:def:unit:degree"/>
      </output>
    </OutputList>
  </outputs>
  <!-- PARAMETERS -->
  <parameters>
    <ParameterList>
      <steadyStateResponse>
        <ConditionalCurve fixed="true">
          <condition name="calibrationTime">
            <swe:Time definition="urn:ogc:def:phenomenon:time"
                      uom="urn:ogc:def:unit:iso8601">2004-01-01T04:30:00
            </swe:Time>
          </condition>
          <data>
            <swe:NormalizedCurve>
              <swe:function>
```

```
                <swe:Curve arraySize="2">
                  <swe:definition>
                    <swe:Coordinates>
                      <swe:axis name="windDirection">
                        <swe:Quantity definition="urn:ogc:phenomenon:windDirection"
                                      uom="urn:ogc:def:unit:degree"/>
                      </swe:axis>
                      <swe:axis name="measuredWindDirection">
                        <swe:Quantity definition="urn:ogc:phenomenon:windDirection"
                                      uom="urn:ogc:def:unit:degree"/>
                      </swe:axis>
                    </swe:Coordinates>
                  </swe:definition>
                  <swe:tupleValues>0.0,0.0 360.0,360.0</swe:tupleValues>
                </swe:Curve>
              </swe:function>
            </swe:NormalizedCurve>
          </data>
        </ConditionalCurve>
      </steadyStateResponse>
    </ParameterList>
  </parameters>
  <!-- METHOD -->
  <method xlink:href="urn:ogc:def:process:1.0:detector"/>
</Detector>
```

Appendix E: XML of Rain Gauge (Rain Fall Detector)

```
<Detector id="DAVIS_RAIN_GAUGE">
  <identification>
    <IdentifierList>
      <identifier name="longName">
        <Term qualifier="urn:ogc:def:identifier:longName">
           Davis Rain Fall Detector
        </Term>
      </identifier>
      <identifier name="modelNumber">
        <Term qualifier="urn:ogc:def:identifier:modelNumber">7852</Term>
      </identifier>
    </IdentifierList>
  </identification>
  <!-- REFERENCE FRAME -->
  <referenceFrame>
    <gml:EngineeringCRS gml:id="RAIN_GAUGE_FRAME">
      <gml:srsName>Rain Fall Detector Spatial Frame</gml:srsName>
      <gml:usesCS xlink:href="urn:ogc:def:cs:xyzFrame"/>
      <gml:usesEngineeringDatum>
        <gml:EngineeringDatum gml:id="RAIN_GAUGE_DATUM">
          <gml:datumName>Rain Fall Detector Spatial Datum</gml:datumName>
          <gml:anchorPoint>
              Origin is situated at the connector/case junction.
              X,Y,Z are undetermined since orientation is not critical.
          </gml:anchorPoint>
        </gml:EngineeringDatum>
      </gml:usesEngineeringDatum>
    </gml:EngineeringCRS>
  </referenceFrame>
  <!-- INPUTS -->
  <inputs>
    <InputList>
      <input name="rainFall">
        <swe:Quantity definition="urn:ogc:def:phenomenon:rainFall"
                      uom="urn:ogc:def:unit:meter" scale="1e-3"/>
      </input>
    </InputList>
  </inputs>
  <!-- OUTPUTS -->
  <outputs>
    <OutputList>
      <output name="measuredRainFall">
        <swe:Quantity definition="urn:ogc:def:phenomenon:rainFall"
                      uom="urn:ogc:def:unit:meter" scale="1e-3"/>
      </output>
    </OutputList>
  </outputs>
  <!-- PARAMETERS -->
  <parameters>
    <ParameterList>
      <steadyStateResponse>
        <ConditionalCurve fixed="true">
          <condition name="calibrationTime">
            <swe:Time definition="urn:ogc:def:phenomenon:time"
                      uom="urn:ogc:def:unit:iso8601">2004-01-01T04:30:00
            </swe:Time>
          </condition>
          <data>
            <swe:NormalizedCurve>
              <swe:function>
                <swe:Curve arraySize="2">
```

```
                    <swe:definition>
                      <swe:Coordinates>
                        <swe:axis name="rainFall">
                          <swe:Quantity definition="urn:ogc:def:phenomenon:rainFall"
                                        uom="urn:ogc:def:unit:meter" scale="1e-3"/>
                        </swe:axis>
                        <swe:axis name="measuredRainFall">
                          <swe:Quantity definition="urn:ogc:def:phenomenon:rainFall"
                                        uom="urn:ogc:def:unit:meter" scale="1e-3"/>
                        </swe:axis>
                      </swe:Coordinates>
                    </swe:definition>
                    <swe:tupleValues>0,0 999,999</swe:tupleValues>
                  </swe:Curve>
                </swe:function>
              </swe:NormalizedCurve>
            </data>
          </ConditionalCurve>
        </steadyStateResponse>
      </ParameterList>
    </parameters>
    <!-- METHOD -->
    <method xlink:href="urn:ogc:def:process:1.0:detector"/>
  </Detector>
```

Appendix F: XML of Full Weather Station System

```
<?xml version="1.0"?>
<SensorML xmlns="http://www.opengis.net/sensorML"
          xmlns:swe="http://www.opengis.net/swe"
          xmlns:gml="http://www.opengis.net/gml"
          xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
          xmlns:xlink="http://www.w3.org/1999/xlink"
          xsi:schemaLocation="http://www.opengis.net/sensorML
                 http://vast.uah.edu/schemas/sensorML/1.0.30/base/sensorML.xsd"
          version="1.0">
  <System id="DAVIS_SIMPLE_STATION">
    <!--============================-->
    <!-- Station Discovery Metadata -->
    <!--============================-->
    <description>
      <swe:Discussion>
         Simple Weather Station measuring temperature, pressure,
         wind and rain fall</swe:Discussion>
    </description>
    <identification>
      <IdentifierList>
        <identifier name="longName">
          <Term qualifier="urn:ogc:def:identifier:longName">
             Davis Weather Monitor II Station</Term>
        </identifier>
        <identifier name="shortName">
          <Term qualifier="urn:ogc:def:identifier:shortName">
             Davis Weather Station</Term>
        </identifier>
        <identifier name="modelNumber">
          <Term qualifier="urn:ogc:def:identifier:modelNumber">7440</Term>
        </identifier>
        <identifier name="manufacturer">
          <Term qualifier="urn:ogc:def:identifier:manufacturer">
             Davis Instruments</Term>
        </identifier>
      </IdentifierList>
    </identification>
    <classification>
      <ClassifierList>
        <classifier name="intendedApplication">
          <Term qualifier="urn:ogc:def:classifier:application">weather</Term>
        </classifier>
        <classifier name="sensorType">
          <Term qualifier="urn:ogc:def:classifier:sensorType">thermometer</Term>
        </classifier>
        <classifier name="sensorType">
          <Term qualifier="urn:ogc:def:classifier:sensorType">barometer</Term>
        </classifier>
        <classifier name="sensorType">
          <Term qualifier="urn:ogc:def:classifier:sensorType">anemometer</Term>
        </classifier>
        <classifier name="sensorType">
          <Term qualifier="urn:ogc:def:classifier:sensorType">rain gauge</Term>
        </classifier>
      </ClassifierList>
    </classification>
    <validTime>
      <StartTime>2005-01-01</StartTime>
      <EndTime>currentTime</EndTime>
    </validTime>
```

```
<contact role="urn:ogc:def:identifier:manufacturer">
  <ResponsibleParty>
    <organizationName>Davis Instruments</organizationName>
    <contactInfo>
      <phone>
        <voice>+01-510-732-9229</voice>
        <facsimile>+01-510-732-9188</facsimile>
      </phone>
      <address>
        <deliveryPoint>3465, Diablo Avenue</deliveryPoint>
        <city>Hayward</city>
        <administrativeArea>CA</administrativeArea>
        <postalCode>94545-2778</postalCode>
        <country>USA</country>
        <electronicMailAddress>sales@davisnet.com</electronicMailAddress>
      </address>
    </contactInfo>
  </ResponsibleParty>
</contact>
<documentation role="documents">
  <DocumentList>
    <member name="userManual">
      <Document>
        <description>
          <swe:Discussion>Davis Weather Monitor Manual</swe:Discussion>
        </description>
        <fileLocation
            xlink:href="http://www.davisnet.com/support/manuals/7440.pdf"/>
      </Document>
    </member>
  </DocumentList>
</documentation>
<!--============================-->
<!-- Station Coordinate Frame -->
<!--============================-->
<referenceFrame>
  <gml:EngineeringCRS gml:id="STATION_FRAME">
    <gml:srsName>Weather Station Spatial Frame</gml:srsName>
    <gml:usesCS xlink:href="urn:ogc:def:cs:xyzFrame"/>
    <gml:usesEngineeringDatum>
      <gml:EngineeringDatum gml:id="STATION_DATUM">
        <gml:datumName>Weather Station Spatial Datum</gml:datumName>
        <gml:anchorPoint>
           Origin is at the base of the mounting.
           Z is along the axis of the mounting pole - typically vertical.
           X and Y are orthogonal to Z but undetermined.
        </gml:anchorPoint>
      </gml:EngineeringDatum>
    </gml:usesEngineeringDatum>
  </gml:EngineeringCRS>
</referenceFrame>
<!--================-->
<!-- Station Inputs -->
<!--================-->
<inputs>
  <InputList>
    <input name="ambiantTemperature">
      <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"/>
    </input>
    <input name="atmosphericPressure">
      <swe:Quantity definition="urn:ogc:def:phenomenon:pressure"/>
    </input>
    <input name="windSpeed">
      <swe:Quantity definition="urn:ogc:def:phenomenon:windSpeed"/>
    </input>
    <input name="windDirection">
      <swe:Quantity definition="urn:ogc:def:phenomenon:windDirection"/>
```

```
    </input>
    <input name="rainFall">
      <swe:Quantity definition="urn:ogc:def:phenomenon:rainFall"/>
    </input>
  </InputList>
</inputs>
<!--==================-->
<!-- Station Outputs -->
<!--==================-->
<outputs>
  <OutputList>
    <output name="weatherMeasurements">
      <swe:DataGroup>
        <swe:component name="time">
          <swe:Time definition="urn:ogc:def:phenomenon:time"
                          uom="urn:ogc:def:unit:iso8601"/>
        </swe:component>
        <swe:component name="temperature">
          <swe:Quantity definition="urn:ogc:def:phenomenon:temperature"
                              uom="urn:ogc:def:unit:celsius"/>
        </swe:component>
        <swe:component name="barometricPressure">
          <swe:Quantity definition="urn:ogc:def:phenomenon:pressure"
                              uom="urn:ogc:def:unit:bar" scale="1e-3"/>
        </swe:component>
        <swe:component name="windSpeed">
          <swe:Quantity definition="urn:ogc:def:phenomenon:windSpeed"
                              uom="urn:ogc:def:unit:meterPerSecond"/>
        </swe:component>
        <swe:component name="windDirection">
          <swe:Quantity definition="urn:ogc:def:phenomenon:windDirection"
                              uom="urn:ogc:def:unit:degree"/>
        </swe:component>
        <swe:component name="rainFall">
          <swe:Quantity definition="urn:ogc:def:phenomenon:rainfall"
                              uom="urn:ogc:def:unit:meter" scale="1e-3"/>
        </swe:component>
      </swe:DataGroup>
    </output>
  </OutputList>
</outputs>
<!--=======================-->
<!-- Station Detector List -->
<!--=======================-->
<processes>
  <ProcessList>
    <process name="clock"
             xlink:href="urn:vast:sensor:davisClock:1.0:001"/>
    <process name="thermometer"
             xlink:href="urn:vast:sensor:davisTemperature:1.0:001"/>
    <process name="barometer"
             xlink:href="urn:vast:sensor:davisPressure:1.0:001"/>
    <process name="anemometer"
             xlink:href="urn:vast:sensor:davisWindSpeed:1.0:001"/>
    <process name=" windDirectionDetector "
             xlink:href="urn:vast:sensor:davisWindDirection:1.0:001"/>
    <process name="rainGauge"
             xlink:href="urn:vast:sensor:davisRainFall:1.0:001"/>
  </ProcessList>
</processes>
<!--==============================-->
<!-- Station Internal Connections -->
<!--==============================-->
<connections>
  <ConnectionList>
    <connection name="inputToThermometer">
      <Link>
```

```
        <source ref="this/inputs/ambiantTemperature"/>
        <destination ref="thermometer/inputs/temperature"/>
      </Link>
    </connection>
    <connection name="thermometerToOutput">
      <Link>
        <source ref="thermometer/outputs/measuredTemperature"/>
        <destination ref="this/outputs/weatherMeasurements/temperature"/>
      </Link>
    </connection>
    <connection name="inputToBarometer">
      <Link>
        <source ref="this/inputs/atmosphericPressure"/>
        <destination ref="barometer/inputs/barometricPressure"/>
      </Link>
    </connection>
    <connection name="barometerToOutput">
      <Link>
        <source ref="barometer/outputs/measuredBarometricPressure"/>
        <destination ref="this/outputs/weatherMeasurements/barometricPressure"/>
      </Link>
    </connection>
    <connection name="inputToAnemometer">
      <Link>
        <source ref="this/inputs/windSpeed"/>
        <destination ref="anemometer/inputs/windSpeed"/>
      </Link>
    </connection>
    <connection name="anemometerToOutput">
      <Link>
        <source ref="anemometer/outputs/measuredWindSpeed"/>
        <destination ref="this/outputs/weatherMeasurements/windSpeed"/>
      </Link>
    </connection>
    <connection name="inputToWindDirection">
      <Link>
        <source ref="this/inputs/windDirection"/>
        <destination ref="windDirectionDetector/inputs/windDirection"/>
      </Link>
    </connection>
    <connection name="windDirectionToOutput">
      <Link>
        <source ref="windDirectionDetector/outputs/measuredWindDirection"/>
        <destination ref="this/outputs/weatherMeasurements/windDirection"/>
      </Link>
    </connection>
    <connection name="inputToRainGauge">
      <Link>
        <source ref="this/inputs/rainFall"/>
        <destination ref="rainGauge/inputs/rainFall"/>
      </Link>
    </connection>
    <connection name="rainGaugeToOutput">
      <Link>
        <source ref="rainGauge/outputs/measuredRainFall"/>
        <destination ref="this/outputs/weatherMeasurements/rainFall"/>
      </Link>
    </connection>
    <connection name="clockToOutput">
      <Link>
        <source ref="clock/outputs/measuredTime"/>
        <destination ref="this/outputs/weatherMeasurements/time"/>
      </Link>
    </connection>
  </ConnectionList>
</connections>
```

```
<!--=======================-->
<!-- Components Positions -->
<!--=======================-->
<positions>
  <PositionList>
    <!--================================-->
    <!-- Position of Station in EPSG4329 -->
    <!--================================-->
    <position name="stationPosition">
      <swe:Position localFrame="#STATION_FRAME"
                    referenceFrame="urn:ogc:def:crs:EPSG:4329">
        <swe:location>
          <swe:Location definition="urn:ogc:def:phenomenon:location">
            <swe:coordinate name="latitude">
              <swe:Quantity definition="urn:ogc:def:phenomenon:latitude"
                            uom="urn:ogc:def:unit:degree">34.72450</swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="longitude">
              <swe:Quantity definition="urn:ogc:def:phenomenon:longitude"
                            uom="urn:ogc:def:unit:degree">-86.94533</swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="altitude">
              <swe:Quantity definition="urn:ogc:def:phenomenon:altitude"
                            uom="urn:ogc:def:unit:meter">20.1169</swe:Quantity>
            </swe:coordinate>
          </swe:Location>
        </swe:location>
        <swe:orientation>
          <swe:Orientation definition="urn:ogc:def:phenomenon:orientation">
            <swe:coordinate name="trueHeading">
              <swe:Quantity definition="urn:ogc:def:phenomenon:angleToNorth"
                axisCode="Z" uom="urn:ogc:def:unit:degree">87.0</swe:Quantity>
            </swe:coordinate>
          </swe:Orientation>
        </swe:orientation>
      </swe:Position>
    </position>
    <!--============================================-->
    <!-- Position of Barometer in Station Ref Frame -->
    <!--============================================-->
    <position name="barometerPosition">
      <swe:Position localFrame="#BAROMETER_FRAME"
                    referenceFrame="#STATION_FRAME">
        <swe:location>
          <swe:Location definition="urn:ogc:def:phenomenon:location">
            <swe:coordinate name="x">
              <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                axisCode="X" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="y">
              <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                axisCode="Y" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="z">
              <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                axisCode="Z" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
            </swe:coordinate>
          </swe:Location>
        </swe:location>
      </swe:Position>
    </position>
    <!--============================================-->
    <!-- Position of Thermometer in Station Ref Frame -->
    <!--============================================-->
    <position name="thermometerPosition">
      <swe:Position localFrame="#THERMOMETER_FRAME"
                    referenceFrame="#STATION_FRAME">
```

```
    <swe:location>
      <swe:Location definition="urn:ogc:def:phenomenon:location">
        <swe:coordinate name="x">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
            axisCode="X" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="y">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
            axisCode="Y" uom="urn:ogc:def:unit:meter">0.02</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="z">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
            axisCode="Z" uom="urn:ogc:def:unit:meter">1.1</swe:Quantity>
        </swe:coordinate>
      </swe:Location>
    </swe:location>
  </swe:Position>
</position>
<!--============================================-->
<!-- Position of Anemometer in Station Ref Frame -->
<!--============================================-->
<position name="anemometerPosition">
  <swe:Position localFrame="#ANEMOMETER_FRAME"
                referenceFrame="#STATION_FRAME">
    <swe:location>
      <swe:Location definition="urn:ogc:def:phenomenon:location">
        <swe:coordinate name="x">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
            axisCode="X" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="y">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
            axisCode="Y" uom="urn:ogc:def:unit:meter">-0.1</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="z">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
            axisCode="Z" uom="urn:ogc:def:unit:meter">2.0</swe:Quantity>
        </swe:coordinate>
      </swe:Location>
    </swe:location>
  </swe:Position>
</position>
<!--============================================================-->
<!-- Position of Wind Direction Detector in Station Ref Frame -->
<!--============================================================-->
<position name="windDirectionDetectorPosition">
  <swe:Position localFrame="#WIND_DIRECTION_DETECTOR_FRAME"
                referenceFrame="#STATION_FRAME">
    <swe:location>
      <swe:Location definition="urn:ogc:def:phenomenon:location">
        <swe:coordinate name="x">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
            axisCode="X" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="y">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
            axisCode="Y" uom="urn:ogc:def:unit:meter">-0.1</swe:Quantity>
        </swe:coordinate>
        <swe:coordinate name="z">
          <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
            axisCode="Z" uom="urn:ogc:def:unit:meter">2.0</swe:Quantity>
        </swe:coordinate>
      </swe:Location>
    </swe:location>
    <swe:orientation>
      <swe:Orientation definition="urn:ogc:def:phenomenon:orientation">
        <swe:coordinate name="z">
```

```
              <swe:Quantity definition="urn:ogc:def:phenomenon:angle"
                axisCode="Z" uom="urn:ogc:def:unit:degree">-87.0</swe:Quantity>
            </swe:coordinate>
          </swe:Orientation>
        </swe:orientation>
      </swe:Position>
    </position>
    <!--============================================-->
    <!-- Position of Rain Gauge in Station Ref Frame -->
    <!--============================================-->
    <position name="rainGaugePosition">
      <swe:Position localFrame="#RAIN_GAUGE_FRAME"
                    referenceFrame="#STATION_FRAME">
        <swe:location>
          <swe:Location definition="urn:ogc:def:phenomenon:location">
            <swe:coordinate name="x">
              <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                axisCode="X" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="y">
              <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                axisCode="Y" uom="urn:ogc:def:unit:meter">0.6</swe:Quantity>
            </swe:coordinate>
            <swe:coordinate name="z">
              <swe:Quantity definition="urn:ogc:def:phenomenon:distance"
                axisCode="Z" uom="urn:ogc:def:unit:meter">0.0</swe:Quantity>
            </swe:coordinate>
          </swe:Location>
        </swe:location>
      </swe:Position>
    </position>
  </PositionList>
</positions>
<!--==================================-->
<!-- System Communication Interfaces -->
<!--==================================-->
<interfaces>
  <InterfaceList>
    <interface name="serial">
      <InterfaceDefinition>
        <!-- http://www.interfacebus.com/Design_Connector_RS232.html -->
        <applicationLayer>
          <Protocol definition="urn:davis:def:protocol:weatherLink"/>
        </applicationLayer>
        <physicalLayer>
          <Protocol definition="urn:ogc:def:protocol:RS232">
            <property name="num-bits">
              <swe:Count>8</swe:Count>
            </property>
            <property name="parity">
              <swe:Boolean>false</swe:Boolean>
            </property>
          </Protocol>
        </physicalLayer>
        <mechanicalLayer>
          <Connector definition="urn:ogc:def:connector:DB9">
            <property name="pin-out">
              <swe:Category definition="urn:ogc:def:pinout">EIA574</swe:Category>
            </property>
          </Connector>
        </mechanicalLayer>
      </InterfaceDefinition>
    </interface>
    <connection>
      <Link>
        <source ref="this/outputs/weatherMeasurements"/>
        <destination ref="serial"/>
```

```
          </Link>
        </connection>
      </InterfaceList>
    </interfaces>
  </System>
</SensorML>
```

References

- Sensor Model Language (SensorML) Implementation Specification, OGC-05-086r2.
- Sensor Model Language UAH - VAST webpage, http://vast.uah.edu/SensorML/