# Open Geospatial Consortium Inc.

Date: 2010-03-22

Reference number of this OGC® project document: **OGC 10-001**

Version: 1 (Rev 1.2)

Category: OGC® Discussion Paper

Editor: **Stuart E. Middleton (Ed.)**

# SANY Fusion and Modelling Architecture



**Warning**

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

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

| | |
|---|---|
| Document type: | Abstract Specification |
| Document subtype: | OGC® Discussion Paper |
| Document stage: | Approved |
| Document language: | English |

# Sixth Framework Programme
# Priority IST 2.5.12
# Information Society Technologies


## Integrated Project


**Contract No.: 033564**

## Deliverable D3.3.2.3

Version 1.2

## SANY Fusion and Modelling Architecture

Due date of deliverable: 31/05/2009

Internal release date: 06/07/2009

Actual submission date: 16/12/2009

# Document Control Page

| | | |
|---|---|---|
| **Title** | SANY Fusion and Modelling Architecture | |
| **Creator** | Stuart E. Middleton | |
| **Editor** | Stuart E. Middleton | |
| **Description** | This document reports the considered SANY best practice for using OGC standards to provide generic fusion processing services. Concrete case studies are documented and a detailed appendix is provided with example of XML request and responses. | |
| **Publisher** | SANY Consortium | |
| **Contributors** | Stuart E. Middleton, Galina Veres, Zlatko Zlatev, IT Innovation Centre, University of Southampton, UK **[IT-INNOV]**<br>Kym Watson, Ralf Bommersbach, Siegbert Kunz, Desiree Hilbring, Fraunhofer-Institute IITB, Germany **[IITB]**<br>Mark Lidstone, Tingting Shu, BMT Cordah, UK **[BMT]**<br>Patrick Jacques, Spacebel, Belgium **[SPB]** | |
| **Type** | Text | |
| **Format** | MS Word | |
| **Language** | EN-GB | |
| **Creation date** | 20/03/2009 | |
| **Version number** | 1.2 | |
| **Version date** | 16/12/2009 | |
| **Last modified by** | Stuart E. Middleton | |
| **Rights** | Copyright University of Southampton, Fraunhofer-Institute IITB, BMT Cordah, Spacebel, 2009. Published under the terms of the SANY contract. | |
| **Audience** | ☐ internal<br>☒ public<br>☐ restricted, access granted to: | |
| **Review status** | ☒ Draft<br>☒ WP manager accepted<br>☒ SP manager accepted<br>☒ MB quality controlled<br>☒ Coordinator accepted | Where applicable:<br>☐ Accepted by the GA<br>☒ Accepted by the GA as public document |
| **Action requested** | ☒ to be revised by partners involved in the preparation of the deliverable<br>☒ to be revised by all SANY partners<br>☒ for approval of the WP manager<br>☒ for approval of the SP manager<br>☒ for approval of the Quality Manager<br>☒ for approval of the Project Coordinator<br>☒ for approval of the General Assembly | |
| **Requested deadline** | N/A | |

# Revision history

| Version | Date | Modified by | Comments |
|---|---|---|---|
| Skeleton | 20/03/2009 | IT-INNOV | Skeleton for SP3 approval |
| 0.1 | 20/05/2009 | IT-INNOV | Initial draft |
| 0.2 | 26/05/2009 | IT-INNOV | IT-INNOV section 2 input |
| 0.3 | 02/06/2009 | IT-INNOV, BMT, SPB, IITB | Collated input from BMT, SPB and IITB plus IT-INNOV edits |
| 0.4 | 15/06/2009 | IITB, BMT, IT-INNOV | More collated inputs from IITB, BMT, IT-INNOV |
| 0.5 | 29/06/2009 | IT-INNOV | Version with SP3 meet comments cleaned up |
| 1.0 | 06/07/2009 | IITB, BMT, IT-INNOV | SP3 release candidate ready for partner review |
| 1.0.1 | 20/08/2009 | AIT | MB level QA |
| 1.1 | 01/09/2009 | IT-INNOV | Final versions created for (1) reviewers (2) public |
| 1.2 | 16/12/2009 | IT-INNOV | OGC Discussion Paper |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |
| | | | |

# Table of Contents

# List of Figures

# List of Tables

# List of SANY Best Practice

# 1. Introduction

The SANY project has a high level objective to *develop fusion services that allow environmental risk applications to combine available information into comprehensive knowledge about the problem in hand*. We have broken down this objective [D3.3.1.3] into several concrete sub-objectives, the combination of which we hope fulfils this high level objective:

- Generic algorithm design contributing towards a Generic Fusion and Modelling engine (GFME) concept
- Fusion capabilities integrated into the SANY service environment
- Propagation of sensor uncertainty information throughout algorithmic processing
- Support for ad-hoc and mobile sensor information sources
- Concrete algorithm implementations as case studies applied to available datasets

Data fusion and modelling is an effective way to add value to existing datasets of sensor measurements. This can be achieved via aggregation of datasets or inference of new data from relationships and patterns within existing datasets. Generic fusion provisions data fusion in a way that separates configuration and data from the algorithmic processing itself, allowing re-use of algorithms and pre/post-processing between datasets. Re-use of algorithms and techniques lowers the cost of development, configuration and deployment of fusion services.

SANY fusion services use the OGC service infrastructure, and in particular the SWE standard set for all communication with end-users. Support for such standards allows our processing work to be more easily integrated both now and in the future with other third party services, either hosting new datasets or new processing capabilities.

The metadata provided as part of the SWE standards has allowed us to make progress towards the goal of supporting ad-hoc and mobile sensors through support for an automated 'plug and play' dataset capability. Once processing services can automatically self-configure to the data they are processing, the resulting sensor network processing capability become agile and robust in response to network changes.

Through our concrete case studies we have made significant progress in the representation, sharing and processing of sensor accuracy associated with each of our measurement datasets. Each SANY service that provides a dataset also provides uncertainty metadata, such as the sensor precision values for each measurement device, which processing services can choose to propagate through algorithmic processing. Processing service result sets are also generated with uncertainty metadata, detailing for each result associated variances, standard deviations etc. This representation and use of uncertainty helps us improve accuracy when generating results and make better interpretations when analysing results for decision support.

Since the output of data fusion is often either a domain application or decision support system we support the SANY work on geospatial visualization, providing data in forms that can be rendered in a variety of different ways. Techniques such as heat maps, direction vector fields and contoured geospatial overlays provide interactive ways for users to see the fusion output in the context it was intended.

As a final achievement we have decided to compile and share our considered view of 'SANY best practice' when implementing or using processing services in the context of an OGC service infrastructure. Each best practice recommendation is made from the experience gained within the SANY project and it is hoped that this will guide other processing service / application users in their future work. Concrete examples can also be found in the appendix section to provide grounded exemplars for our recommendations.

## 1.1. Purpose

This document is part of the SP3 fusion and modelling design activity. The deliverable D3.3.2.3 represents the final year (V2) design. The purpose of this document is to identify SANY best practice and case study designs developed within SANY.

## 1.2. Intended audience

This document will be released as a public statement of best practice, from the SANY consortium, regarding how best to design fusion and modelling services using the OGC SWE standard set. The audience is expected to be of a technical nature, consisting of people who are considering using the OGC standards.

## 1.3. Structure of document

In section 2 we outline the considered SANY best practice, both for using OGC SWE standards for developing processing services and for developing generic fusion approaches that can process sensor accuracy and uncertainty introduced by algorithmic processing.

In section 3 we document the architectural designs from IT-INNOV, IITB and BMT, providing concrete case studies for how processing services can be developed.

In section 4 we draw conclusions.

The appendix A provides a complete set of input and output specifications for each case study described in section 3. This is intended to provide the reader with concrete examples of OGC request and responses, useful for a developer considering either using a processing service or developing their own processing service.

## 1.4. Abbreviations and Acronyms

| | |
|---|---|
| 52N | Vendor of an open source implementation of OGC SWE services |
| ANN | Artificial Neural Network |
| ASAR | Advanced Synthetic Aperture Radar |
| ASMI | SANY Application Schema for Meta-information |
| BPEL | Business Process Execution Language |
| CLIPS | C Language Integrated Production System |
| CSV | Comma Separated Values |
| DLL | Dynamic Link Library |
| DSS | Decision Support System |
| ENVISAT | European Space Agency satellite |
| EO | Earth Observation |
| FTP | File Transfer Protocol |
| GFME | Generic Fusion and Modelling Engine |
| GML | Geography Markup Language |
| GPS | Global Positioning System |
| HTTP | Hyper-Text Transfer Protocol |
| IPTA | Interferometric Point Target Analysis |
| KML | Keyhole Markup Language |
| MAPE | Mean Absolute Percentage Error |
| MDS | Map and Diagram Service |
| MSE | Mean Square Error |
| NOAA | National Oceanic and Atmospheric Administration |
| O&M | Observation & Measurement model |
| OGC | Open Geospatial Consortium |
| OWL | Web Ontology Language |
| PCS | Processing Chain Service |
| PT | Point Target |
| SANY | Sensors ANYwhere integrated project |
| SensorML | Sensor Model Language |
| SensorSA | SANY Sensor Service Architecture |
| SLD | Styled Layer Descriptor |
| SOAP | Simple Object Access Protocol |
| SOS | Sensor Observation Service |
| SP<X> | SANY Sub Project |
| SPS | Sensor Planning Service |
| SWE | Sensor Web Enablement |
| UCUM | Unified Code for Units of Measure |
| UI | User Interface |
| UncertML | Uncertainty Markup Language |
| URL | Uniform Resource Locator |
| URN | Uniform Resource Name |
| WFS | Web Feature Service |
| WGS | World Geodetic System |
| WMS | Web Map Service |
| WNS | Web Notification Service |

| | |
|---|---|
| WPS | Web Processing Service |
| WSDL | Web Services Description Language |
| XML | Extensible Markup Language |
| XSL | Extensible Stylesheet Language |
| XSLT | XSL Transformation |

# 2. Services and information models for generic fusion

Data fusion and modelling is an effective way to add value to existing datasets of sensor measurements. Aggregating multiple datasets within a common frame of reference (e.g. a single spatial region) provides context for decision support. Domain relationships can be used to infer new data from aggregated existing data. Data driven interpolation and prediction techniques can discover and use patterns within existing datasets to provide information at interesting points in space or time.

**Value proposition: Data fusion**

Data fusion and models add value to existing datasets. This is achieved through aggregation and correlation of data within a common frame of reference and the use of data-driven interpolation, prediction and inference techniques.

Generic data fusion techniques separate configuration and data from the algorithmic processing itself. Dynamic configuration allows the re-use of existing algorithms for different datasets. Metadata associated with existing datasets allows the automation of pre-processing and post-processing, and the ability to plug and play datasets. Use of knowledge-based techniques allows the semi-automation of the fusion assessment process.

**Value proposition: Generic fusion**

Generic fusion lowers the cost of data fusion. This is achieved through re-use of existing algorithms, dynamic adaption of pre-processing to datasets and semi-automated fusion assessment.

## 2.1. Architectural context

Figure 2.1 shows the functional domains of the SANY Sensor Service Architecture (SensorSA) as defined in SANY Deliverable D2.3.3 ‘Specification of the Sensor Service Architecture V2’, 2009. The Fusion and Modelling Architectural Design of this document addresses part of the Mediation & Processing Domain.

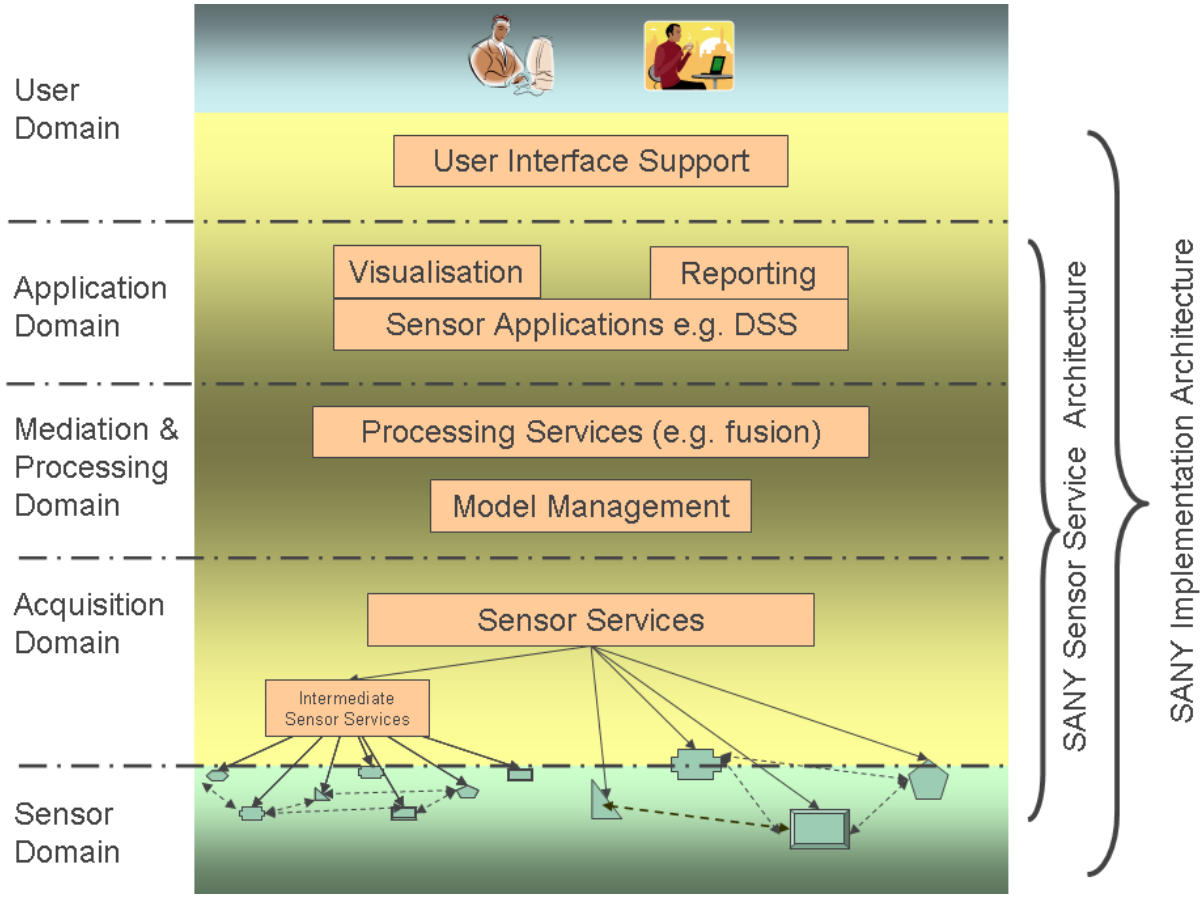



*Figure 2.1. Functional Domains of the SensorSA*

This domain mediates between the Application Domain and the underlying information sources in the Acquisition Domain, whereby the domains should not be understood as a strict layered architecture. The Mediation & Processing Domain provides generic or thematic processing capabilities such as fusion of information (from sensors and other information sources), management of models or access to model results. In addition, service support for the discovery of sensors, data and services, naming resolution or service chaining are grouped in the Mediation & Processing Domain. The support for processing chains is described in D2.3.3 section 10.9.

## 2.2. Service infrastructure

The SANY processing infrastructure consists of a number of services providing fusion and modelling functionality, clients providing user interfaces to both control processing and visualize the results and services to provide access to the datasets. Figure 2.2 shows the SANY processing infrastructure and its associated data flow. The case studies in section 3 relate to each of the software services shown in this figure.

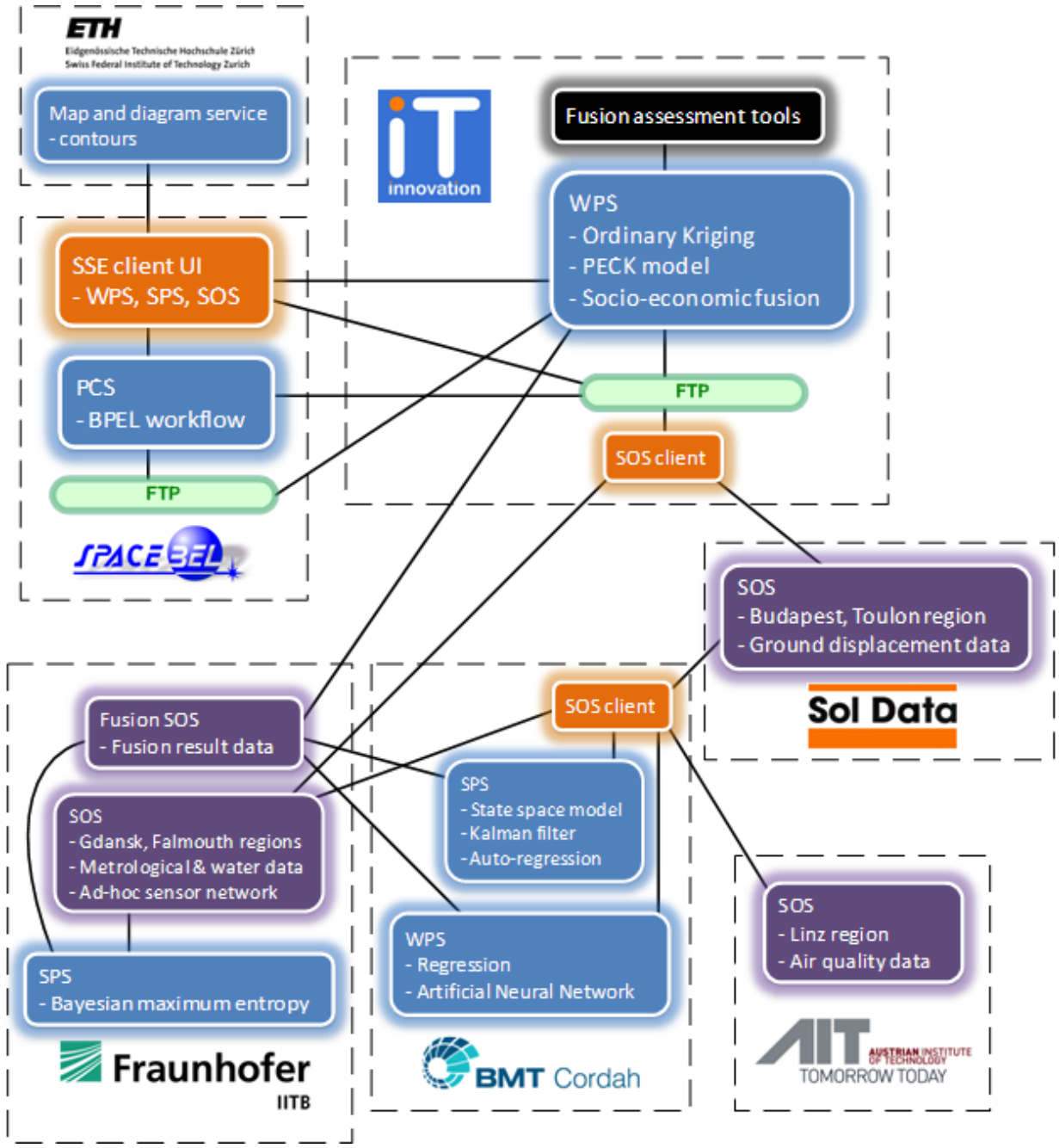



*Figure 2.2. SANY processing service data flow*

We have chosen to support both the OGC Web Processing Service (WPS) and Sensor Planning Service (SPS) standards when writing our processing services. Each standard has different strengths and weaknesses (see section 2.4) and in SANY we wanted to see how they would both perform.

All sensor data in SANY is obtained directly from a Sensor Observation Service (SOS), or indirectly via a cached SOS response loaded onto an FTP site. The FTP mechanism is supported to allow processing services to be part of a larger workflow, orchestrated by a processing chain service (PCS). In SANY we have three SOS's that provide access to our sensor measurement datasets; these datasets contain sensor measurements for air pollution, wind, water, ground displacement and more.

When a processing service (WPS or SPS) needs data it will use a SOS client to issue a SOS request and parse the SOS response. The parsed XML response data will either be used directly or saved in an intermediate result format such as comma separated variable (CSV). In addition to extracting the raw sensor measurements from the SOS responses the associated metadata (units, sensor accuracy etc) is also extracted and made available to use by the processing service. Where the FTP site is used the cached SOS response XML is provided, allowing the SOS client to skip the SOS request stage. Multiple input sources are possible and the processing service must in these cases pre-process the sensor data from each input source, aggregate it together and do the syntax checking (etc) as required prior to algorithmic processing.

To achieve high accuracy / performance fusion algorithms must normally be heavily tailored to the dataset they are working on. In SANY we have separated configuration and data from the algorithms themselves, allowing a number of generic fusion algorithms to be created than can run on a dynamically assigned dataset. Each dataset undergoes a fusion assessment to create a dataset specific configuration file. At runtime the fusion configuration (or 'profile') can be selected along with the dataset. A fusion assessment tool will be created by IT-INNOV to semi-automate this assessment process, allowing dataset owners to prototype fusion algorithms on the fly prior to a fusion expert finishing the job.

After algorithmic processing a number of result formats will be created and made available immediately via a FTP site. In addition, O&M results will be uploaded to a Fusion SOS for longer term storage in a OGC SWE compatible format. Using an SOS to store fusion results allows SANY to treat fusion processes as just another sensor service, making interoperability with third party services and the creation of processing chains easier.

A Service Support Environment (SSE) client has been created for each of the SANY services (WPS, SPS, SOS). This offers a simple user interface to provide the input parameters needed to run fusion processes. When result sets are generated the SSE client visualizes them using geospatial maps over which is rendered a contoured colour map and a representation of the result set shape; the Map and Diagram Service (MDS) is used to achieve this.

## 2.3. OGC standards overview

This section gives a short overview of the OGC standards and information models used for the processing and support services in the SANY SensorSA (see table 2.1)

| OGC Standard | Purpose | Reference |
|---|---|---|
| **Sensor Observation Service (SOS)** | Provides access to observations from sensors and sensor systems that is consistent for all sensor systems including remote, in-situ, fixed and mobile sensors | [SOS] |
| **Sensor Planning Service (SPS)** | Provides an interface to task any kind of sensor to retrieve collection assets. | [SPS] |
| **Web Processing Service (WPS)** | Describes a common interface for services offering processing operations on spatial (vector as well as raster) and non-spatial data. | [WPS] |

**Table 2.1. Overview of the OGC Standards used for the processing and support**

The SensorSA basically adopts the specification of the Observations and Measurements (O&M) model as defined in [O&M Part 1]. This information model is of core relevance for the access and interpretation of the data provided through the Sensor Observation Service. It defines an *observation* as "an act associated with a discrete time instant or period through which a number, term or other symbol is assigned to a phenomenon".

The phenomenon is a *property* of an identifiable object, which is the *feature of interest* of the observation. The observation uses a *procedure*, which is often an instrument or sensor but may be a process chain, human observer, algorithm, computation or simulator. The key idea is that the observation *result* is an estimate of the value of some property of the feature of interest, and the other observation properties provide context or meta-information to support evaluation, interpretation and use of the result.

The SensorSA uses the schema defined by the OGC Sensor Model Language (SensorML) specification [SensorML] for the description of measurement processes.

In SensorSA data arises not only from sensor measurements and observations, but also from data processing with specific services, e.g. a fusion or kriging algorithm to generate a spatial coverage from a set of measurement points, or a time series analysis to produce a temporal interpolation. The results of such data processing steps are themselves uncertain, on the one hand due to the uncertainty of the input data, on the other hand due to the probabilistic or approximate nature of the processing itself. Within the SensorSA, the uncertainty of data sets is described using the descriptive model language UncertML, which was developed within the INTAMAP project [INTAMAP].

## 2.4. Use of OGC standards for generic processing

*Generic processing*

There are four stages to a generic processing service, (1) pre-processing (2) processing (3) post-processing and (4) storage. In pre-processing the input data must be downloaded from either an online service or remote file store, then all sources formatted, aggregated and syntax checked. Once ready a number of algorithm(s) must process the data and generate the result set(s). The post-processing phase will format these result sets, adding metadata as required, in a form usable

by the client. Finally the result set output data must be stored somewhere so the client can access it, either in a service or remote file store.

In SANY we make heavy use of the OGC and OASIS standards for our processing and support services. Datasets are stored and accessed via Sensor Observation Services (SOS) and FTP sites. Processing is performed by Web Processing Services (WPS) and Sensor Planning Services (SPS). Notification of progress and results is achieved using Web Notification Services (WNS) and WS-Notification compliant protocols. This section provides an overview of how and why we use these services.

*Pre-processing: Input data from SOS*

The task of fusion process is to create a new fusion result for an observed property *P* in a time interval *T* and a set of sampling points *S* (e.g. a rectified grid) by applying a fusion algorithm to raw data from available SOS servers. The fusion processing procedure applies the O&M GetObservation operation to each SOS server to obtain the available observations of the desired property. Each available SOS server delivers the requested observations of one or multiple observed properties *P* according to the time interval *T* and within a given geographical area encoded by a Bounding Box. Duplicates are recognized as observations taken by the same procedure (sensor) at the same sampling time; duplicates are deleted from the observation collection.

The fusion processing procedure takes the inaccuracy of the raw sensor data into account. The accuracy metadata should be in the observation result but could be located in SensorML if applicable to all observations by that sensor.

The descriptive model language UncertML developed by the INTAMAP project [INTAMAP] is used to encode the accuracy information into the XML file containing the result of the observation collection (see sections 2.6 and 2.7).

*Processing: Running algorithm(s) via SPS & WPS*

In SANY we have both Sensor Planning Services for fusion processing (WebGenesis-based Fusion SPS) and Web Processing Services. Our processing services act as clients to several Sensor Observation Services (52°North SOS for source data, WebGenesis-based Fusion SOS for storage of fusion results and SOAP protocol (SolData) SOS). We also make use of FTP input for cached SOS responses which allows processing chains to be operated via a Processing Chain Service (PCS).

Both the WPS and SPS standards provide similar functionality, as can be seen from table 2.2. In SANY we have compared and contrasted both approaches via our case studies reported in section 3. The WPS standard is simpler for a service provider, with much flexibility of implementation available to easily handle most processing use cases; this comes at the expense of the client who must potentially support each WPS provider's domain specific input and output formats. The SPS standard is more structured, and provides a pre-defined mechanism to support client notification of progress and result availability. The GetFeasibility operation in the SPS standard allows a client some feedback on the viability of a processing operation before committing to execution. Both standards do not define mechanisms to describe uncertainty or provenance information.

| | Sensor Planning Service (SPS) | Web Processing Service (WPS) |
|---|---|---|
| **Operations** | GetCapabilities<br>DescribeTasking<br>GetFeasibility<br>Submit<br>GetStatus<br>DescribeResultAccess<br>Cancel | GetCapabilities<br>DescribeProcess<br>Execute |
| **Maturity** | Standard is still under development, with V2 expected end 2009 that contains major changes. | Standard well established and stable as of v1.0.0. |
| **Notification** | The standard includes a WNS notification protocol. | Notification protocols are not defined. The WPS standard describes a polling protocol of a status file. |
| **Steering** | Workflow defines sub-process orchestration and SPS operations allow steering (i.e. cancel, update). | No direct support for workflow or process steering. |

**Table 2.2. Comparison of WPS and SPS standards**

**SANY Best Practice: Choice of processing service standard**

The WPS standard is somewhat lightweight, being very flexible in what it can do at the expense of clients who will have to support each provider-specific implementation. The SPS standard defines notification and steering protocols and is thus a good choice if this is important. Both standards are easily extended and can support, for example, uncertainty and provenance metadata as we have demonstrated in SANY via UncertML and SensorML descriptors.

*Post-processing and storage: Output data from PCS, WPS, SPS, Fusion SOS*

After a fusion process run has finished, it is important, that the fusion result contains all necessary information to be able to reproduce the process. SensorML encoding of this information plays an essential role in the Fusion workflow. In both Fusion SPS and WPS it is used for fusion process description. They describe the expected inputs, parameters, outputs, the algorithm itself and the implementation module of the fusion process. This is also useful for fusion service discovery. For each fusion workflow execution a unique SensorML file is created, one per successful executed fusion run, that describes the fusion algorithm executed and the input parameters provided. It contains essentially a copy of the fusion-process description which includes the supplied values (inputs, parameters) for that specific fusion run. The purpose of this SensorML description file is to provide a provenance record to allow a fusion experiment to be reproduced (see below and sections 6.1.4 and 6.2.2).

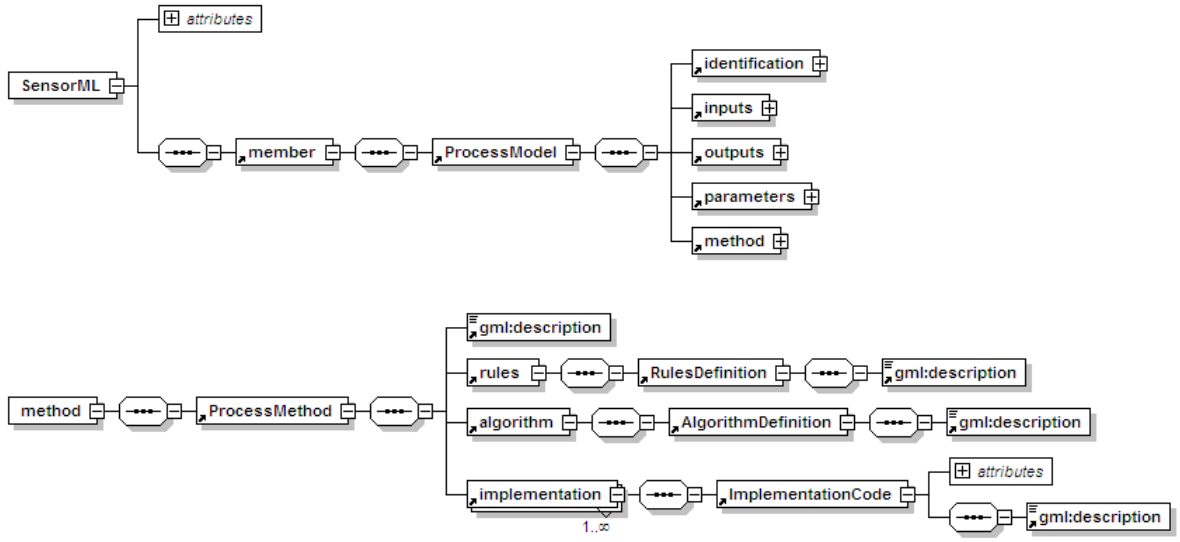
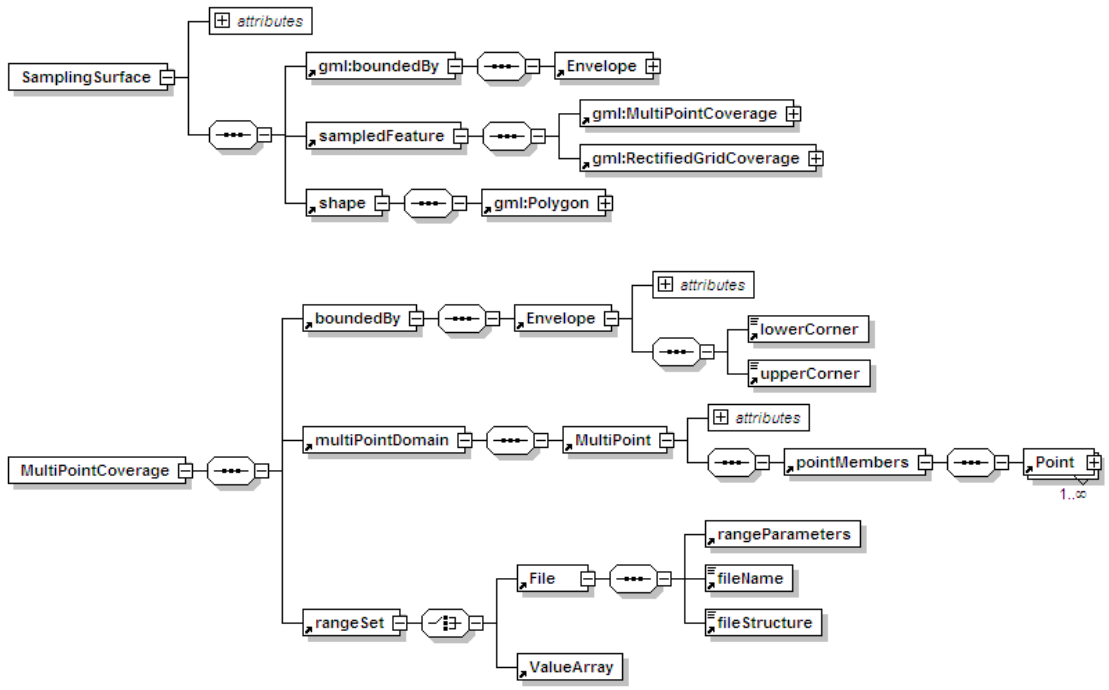



*Figure 2.3. SensorML schema*

The SamplingSurface encoded in SensorML describes the spatial bounds for the processing of fusion data. After a successful fusion execution a sampling surface provides provenance information about the shape of the fusion result set (see below and sections 6.1.4 and 6.2.4).



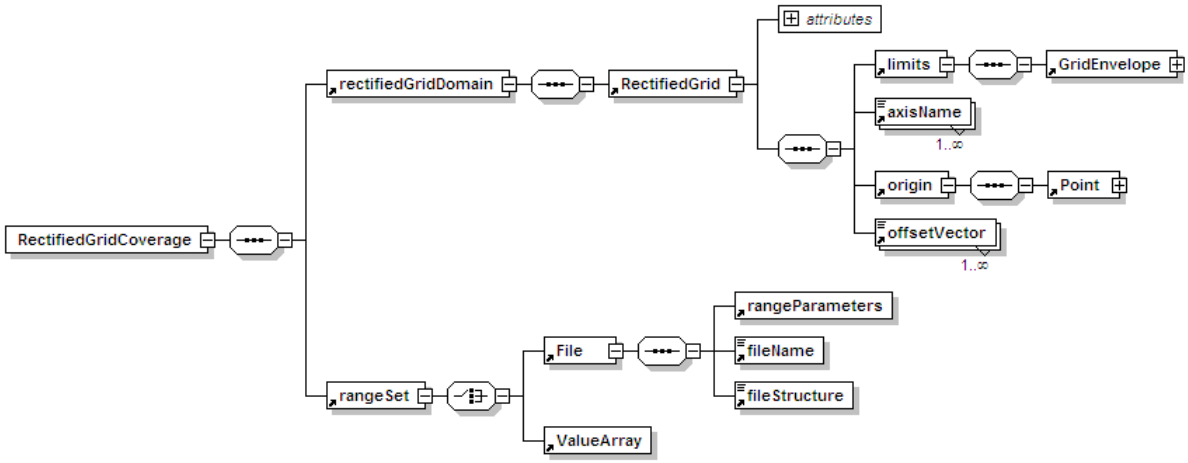



*Figure 2.4. SamplingSurface schema*

As with the SamplingSurface and SensorML described above, the O&M observation also fulfils two roles in the fusion workflow, one for input and one for output. O&M observations, aggregated together in an O&M ObservationCollection if required, serve as input data for the fusion process containing the "source" sensor data that shall be processed. The fusion result is a so-called coverage, a function defined on a space-time grid of sampling points. This coverage is encoded in the result section of an O&M observation using SWE common.

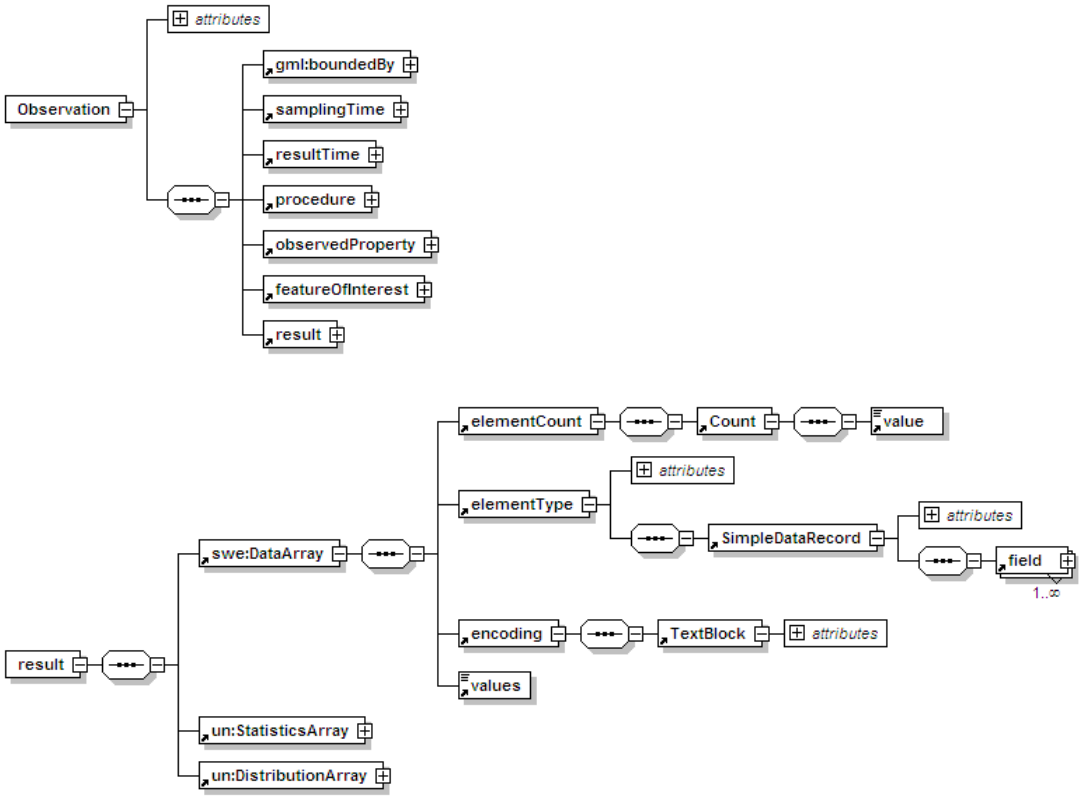



*Figure 2.5. ObservationCollection schema*

The O&M observation includes additional metadata for interpretation of the data and include links to blocks of UncertML embedded within the result block of the O&M observation to

encode the uncertainty of both sensor input data (accuracy) and fusion results (statistical error). (see below and section 2.6 and 2.7).

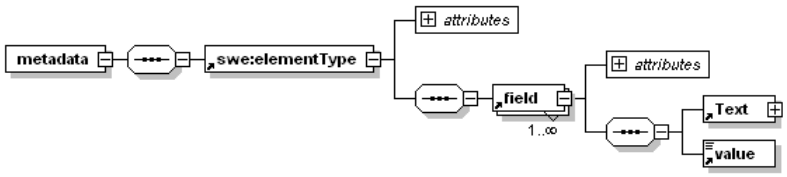



*Figure 2.6.SWE metadata schema*

**SANY Best Practice: Usage of OGC SWE standards for processing services**

Format fusion result sets using the SWE standards for easy reuse in a SWE compliant sensor architecture. Use O&M part 1 (ObservationCollection) standard to format fusion results, along with SWE common, with xlink:href links to UncertML (StatisticsArray, DistributionArray) to store uncertainty information. Dataset specific quality flags can be defined in the om:metadata section. Use SensorML (Process) standard to record both post-processing provenance information and discovery information about what each processing service can offer. Use O&M part 2 (SamplingSurface) standard to describe spatial regions, both as an input parameter and to describe the final result sets.

## 2.5. Discovery of processing services

Discovery of available SOS source data shall be enabled through a SANY Catalogue service. The fusion process queries the Semantic Catalogue for SOS servers with observations of the desired property in a given time interval and in the area of a bounding box around the sampling point set. In addition, clients of processing services can use the Catalogue to query suitable algorithms and WPS / SPS servers.

To ease the discovery of sensor related resources the SANY Application Schema for Meta-information (ASMI) has been specified. Besides conventional resource types like services and data it defines additional meta-information resource types related to the Observation & Measurement Model:

- Feature of interests can be described with the resource type MI_Data_FeatureOfInterest
- Observable properties can be described with the resource type MI_Data_ObservedProperty
- Sensors or workflows can be described with the resource type MI_Data_Procedure. This resource type contains a section to include SensorML data.

The ASMI is specified in detail in section 7.6 of the Deliverable D2.3.3 “Specification of the Sensor Service Architecture V2. It is used in the SANY catalogue. The service interfaces of the catalogue are defined in section 4.3 of the Deliverable D2.4.3 “Sensor Service Specification V2”, in Orchestra WP3.4 OA Service Specifications – Specification of the Catalogue Service [ORCHESTRA1] and in WP3.6 OA Service Implementation – Implementation Specification of the Catalogue Service [ORCHESTRA2].

All meta-information needed for the creation of ASMI documents are available via the SOS operation DescribeSensor and SOS/SPS operation GetCapabilities. Therefore it is possible to create components automatically harvesting SOS instances for the creation of feature of interest, observable properties, procedures and service resources types or SPS instances for the creation of procedures and service resource types.

Once the meta-information is included in the catalogue users can discover it searching the catalogue with the means of specific durables. "DatasetType" and "ServiceType" can be used to discover resources according to their resource or service types. "FeatureOfInterestType", "ObservedPropertyType" and "ProcedureType" can be used to identify specific feature of interests, observable properties, sensors or WPS process or SPS task. A broader search can be performed via "AnyText", which realizes a full text search. All results contain a link to the service, the discovered meta-information was derived from. Therefore the search for specific feature of interests, observable properties and procedures can lead to SWE services accessing theses resources.

The WPS v1.0.0 specification also supports application profiles where a DescribeProcess response can return an URN to an OGC recognized profile. This profile allows a human readable document and WSDL document to be associated with a well known WPS process. However our use of the SANY catalogue service goes one step further associating machine readable SWE metadata to the WPS process, using a SensorML document provided by the WPS and harvested automatically by the catalogue service.

**SANY Best Practice: Usage of application schema for meta-information for discovery**

For the discovery of a sensor, WPS process or SPS task a meta-information schema reflecting specific requirements is needed. The SANY Application Schema for Meta-information (ASMI) can serve as an example supporting these requirements. It eases the discovery of observations according to their observable properties, features of interest and procedures (sensors, processes or tasks).

## 2.6. Quality and irregular metadata

The O&M observation includes additional metadata for interpretation of the data and include xlink:href links to blocks of UncertML embedded within the result block of the O&M observation to define the uncertainty of both sensor input data (accuracy) and fusion results (statistical error).

The fusion processing procedure takes the inaccuracy of the raw sensor data into account. It determines the accuracy of the measurements. So the fusion task executes a *DescribeSensor* operation at the relevant SOS server to acquire this information. The accuracy metadata should be in the observation result but could be located in SensorML if applicable to all observations by that sensor.

The descriptive model language UncertML is used to encode the accuracy information into the XML file containing the result of the observation collection.

**SANY Best Practice: Quality metadata information encoded in SensorML**

In the case of the IITB testbed, this meta-information is encoded in the SensorML of the related procedure (sensor). An example fragment of a SensorML is below:

```
<sml:characteristics>
 <swe:DataRecord
    definition="urn:ogc:def:property:OGC:1.0.1:physicalProperties">
  <swe:field name="measurementProperties">
   <swe:DataRecord
      definition="urn:ogc:def:property:OGC:1.0.1:measurementProperties">
    <swe:field name="Temperature accuracy"
       xlink:href="urn:ogc:def:property:OGC:1.0.1:temperature">
     <swe:QuantityRange
        definition="urn:ogc:def:property:OGC:1.0.1:absoluteAccuracy">
      <swe:uom code="Cel"/>
      <swe:value>-0.5 0.5</swe:value>
     </swe:QuantityRange>
    </swe:field>
   </swe:DataRecord>
  </swe:field>
 </swe:DataRecord>
</sml:characteristics>
```

Special datasets specific tags can be defined in the metadata section to indicate quality flags, e.g. “NaN” indicating that data matches a Not-A-Number value (due to possible sensor failures).

**SANY Best Practice: Metadata flags for irregular data of the fusion result**

In the case of the IITB testbed, the fusion result data section may also contain further quality metadata as a quality flag. An example fragment of XML is below:

```
<om:Observation>
  …
 <om:procedure xlink:href="urn:ogc:def:featureType:OGC:1.0.1:Process:201"/>
 <om:observedProperty
    xlink:href="urn:ogc:def:property:OGC:1.0.1:temperature"/>
  …
 <om:metadata>
  <swe:elementType name="QualityFlag"
     href="http://sanyv1.iitb.fraunhofer.de/Testbedgloss.htm#QualityFlags">
   <swe:field name="Not a Number">
    <swe:Text definition="Data matches a Not-A-Number value from the provider (or error code from
an instrument)"/>
    <swe:value>NaN</swe:value>
   </swe:field>
   <swe:field name="Null">
    <swe:Text definition="Data missing or null"/>
    <swe:value>NULL</swe:value>
   </swe:field>
   <swe:field name="out of engineering range">
    <swe:Text definition="Data exceeds range specified for instrument"/>
    <swe:value>OOER</swe:value>
   </swe:field>
   <swe:field name="out of range for datatype">
    <swe:Text definition="Data is out of valid range for this datatype"/>
    <swe:value>OODTR</swe:value>
   </swe:field>
   <swe:field name="out of calculation range">
    <swe:Text definition="Data exceeds range that its calculation allows"/>
    <swe:value>OOCR</swe:value>
   </swe:field>
  </swe:elementType>
 </om:metadata>
 <swe:DataArray>
   …
  <swe:values>49.0146,8.42612,NaN@@49.0146,8.42614,13.8057@@49.0146,8.42616,
 NULL@@49.0146,8.42618,OOER@@49.0146,8.4262,13.806@@
  </swe:values>
 </swe:DataArray>
  …
</om:Observation>
```

## 2.7. Processing uncertainty

There is a need to represent uncertainty information associated with the sensor input data and fusion processing result data. In SANY our sensor datasets have sensor accuracy information available and all of our algorithms can provide some sort of statistical error information along with the result sets that are generated.

Sensor accuracy information is represented in SANY in terms of accuracy intervals that may or may not be time variant based on the age of the sensor. Often the quality of the accuracy information is based on the level of knowledge about the original sensor measurement campaign that produced the dataset in the first place.

The SOS's within SANY provide O&M observations with xlink:href links to UncertML blocks encoding precision and accuracy ranges either per sensor or per measurement. This best practice is documented in the previous section.

The result sets of our fusion processing services are both temporal and spatial (sometimes both) and come with details about the estimation errors associated with each result. For spatial algorithms we typically generate variance information associated with a result grid, or a probability distribution. For temporal algorithms standard deviation is reported as the measure of uncertainty of the results, with the same unit as the original data.

The SPS and WPS services within SANY provide UncertML blocks embedded within O&M result blocks to define the associated statistical data that comes with the result sets. Standard deviations / variances are represented using a un:StatisticsArray. More complex probability distributions are represented using a un:DistributionArray. In both cases the raw data references the associated UncertML data via xlink:href links in the om:result block. The schema diagram below shows the XML structure we use, and a XML example can be found in sections 6.1.4 and 6.2.3.

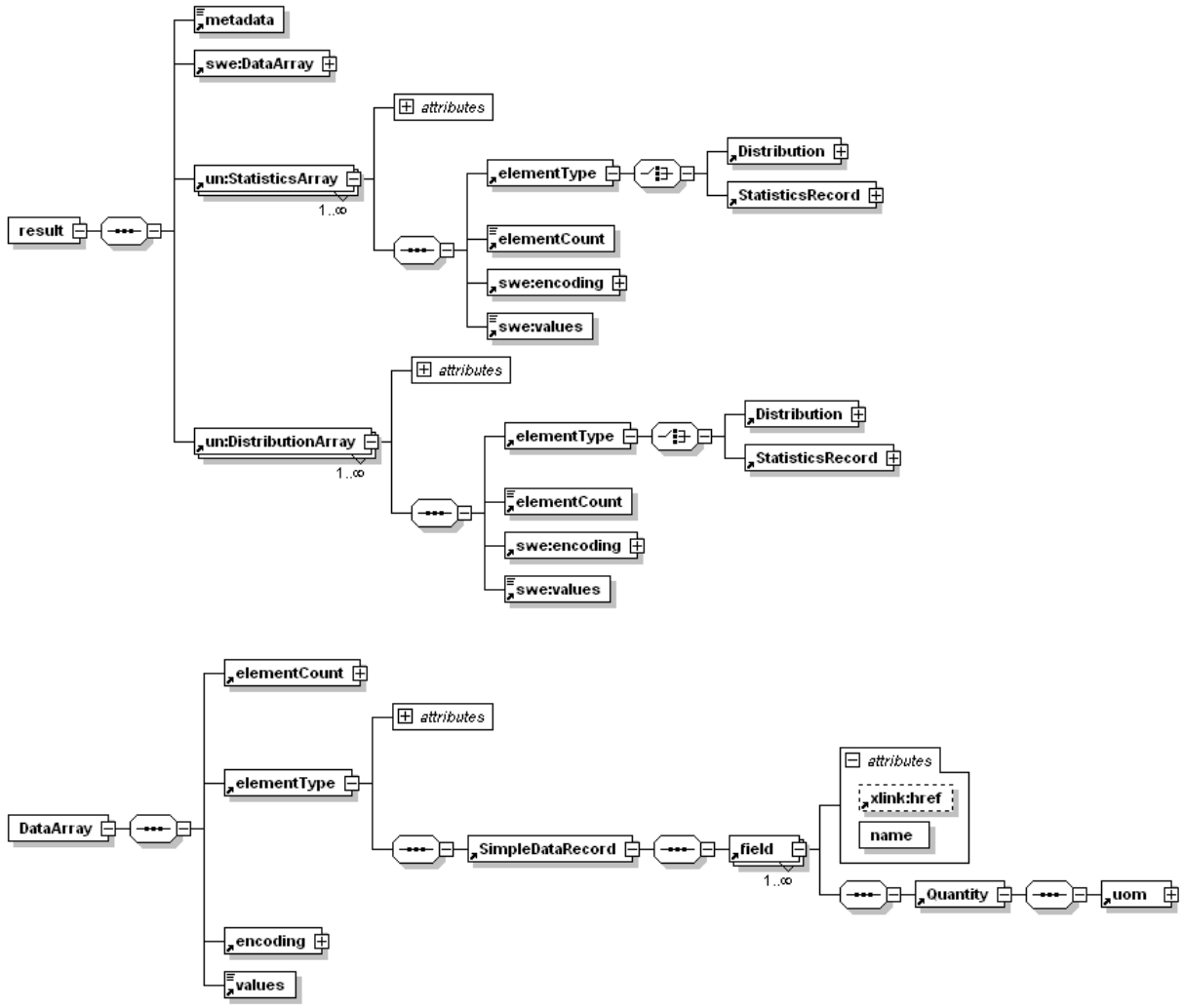



*Figure 2.7. SWE result schema*

It should be noted that current (minor) incompatibility between SWE and UncertML mean that special handling of namespace versions is required. Specifically SWE v1 uses SWE version 1.0.1 whilst UncertML uses SWE v1.0. To validate the SANY observation result file we have used two namespaces, one for SWE v1.0.1 and one for SWE v1.0 and assigned the UncertML swe:values & swe:encoding blocks to the older V1.0 namespace. Clear XML examples can be

found in the appendix 6.1.4 and 6.2.3. Another issue exists with UncertML regarding the linkage of a statistics array to a distribution array; xlink:ref is not currently supported so we have overloaded the attribute definition for now. SANY has engaged directly with the OGC about this and we expect the coming SWE v2 will be fully aligned with UncertML and vice versa.

## 2.8. Semantic annotation and the vocabulary problem

One of the problems when aggregating datasets from different data sources is how to deal with the different ways each data source labels identical, or sometimes very similar, concepts. This semantic annotation problem can occur at a number of different levels. Within a single dataset there can be inconsistent labelling (e.g. 'wspeed' ,'windspeed', 'ws') for the same concept as well as inconsistent use of units (e.g. 'm', 'cm', 'mm'). When dealing with datasets from two separate domains it is common to find that each domain has different labels for similar concepts (e.g. 'Temp', 'Temperature'), uses different units and scales ('$^{o}$C', '$^{o}$F') and properties that are similar, but not identical (e.g. 'longitude' & 'latitude' using different reference frames such as WGS84 & ETRS89).

In order to overcome these problems in SANY we have developed best practice suggestions for naming URN's in SANY and mapping O&M result data concepts to a unique ontological concept. Universal unit conversion is a very large problem and we suggest adopting a scalable incremental support strategy. In SANY we use Unified Code for Units of Measure (UCUM) as does SWE common.

*Vocabulary mapping problem*

There are two use cases for the vocabulary pseudonym mapping problem. The first is multiple labels for a concept in the same dataset. The second is two independent datasets defining different labels for the same concept.

In SANY we have looked at options available in the OGC schema and concluded that adding in xlink:role attributes for observed properties is the preferred to identify per property which ontological concept is being referenced. An example structure can be seen below.

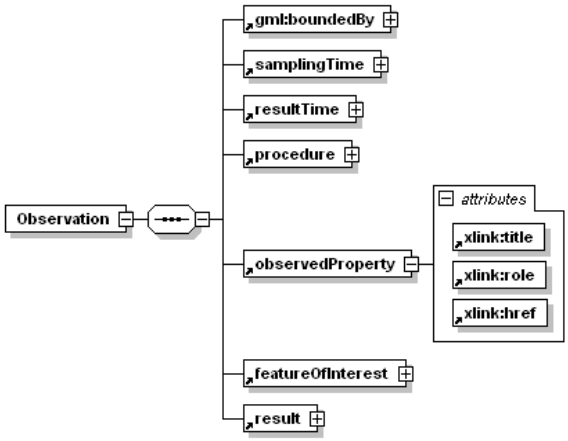



*Figure 2.8. Observation schema*

**SANY Best Practice: Domain ontology linkage**

Observed properties reported by an SOS should be annotated using the xlink:role attribute with a URL to the associated concept in the domain ontology. This then uniquely identifies the concept and allows a client to resolve any vocabulary pseudonym problems.

Where two SOS's refer to the same concept using different domain specific ontologies we recommend using a mapping function, provided by the party interested in aggregating these datasets. Many ontology mapping techniques are available in the literature, each suitable for different problem areas, so we only define a protocol for access to this function and do not define the mapping function itself.

**SANY Best Practice: URN mapping between domains**

Where identical concepts from two datasets must be mapped together a semantic mapping function is recommended. In SANY we suggest the follows the function protocol:

```
<target URN> MapURN( <source URN>,<source ontology URL>,<target ontology URL> )

<source URN> = URN to a concept in source ontology
<source ontology URL> = URL to source ontology (e.g. OWL file)
<target ontology URL> = URL to target ontology (e.g. OWL file)
<target URN> = URN to a concept in target ontology
```

We have also defined a URN definition policy within SANY for all URN's created to describe our datasets.

**SANY Best Practice: URN definition**

Dataset labels should follow a URN definition policy that is compatible with the OGC suggestions [RFC 5165]. Resources that are defined for the purpose of SANY applications follow a naming scheme *urn:ogc:def:objectType:authority[:version]:code*.

Examples are below:

```
urn:ogc:def:data:SANY:2009.05:temperature
urn:ogc:def:statistic:SANY:2009.05:standard-deviation
```

*Unit conversion problem*

Unit information is provided as part of the O&M described data available from a SOS. In SANY our processing services take units into account at the pre-processing stage, performing conversions or transformations as needed. It is not a scalable solution to support all units so in SANY we provide support for units incrementally as we need them. For example the IT-INNOV WPS has a configuration file that specifies, for each unit label, type information and the range of allowed values for syntax checking. As support for more units is needed the unit configuration file is updated; the WPS does not need to be re-deployed for this incremental update. Multiple labels can be added in the unit configuration file to handle unit pseudonyms.

## 2.9. Generic fusion algorithms

**SANY Best Practice: Generic fusion**

In SANY we define the idea of generic fusion as the separation of configuration and data from the algorithmic processing; this allows the development of configurable algorithms that can be re-used on a number of different datasets.

Generic fusion adds value by reducing the overall cost of processing service development via reuse of existing algorithms.

The degree to which each SANY algorithm is generic varies depending on the needs of the application it is supporting. Both the BMT and IITB algorithm sets support soft configuration but are not designed to be configured dynamically at runtime. The IT-INNOV algorithms are hosted within a WPS framework that provides live pre-processing support for profiling of the input data source (FTP, SOS), dataset (offering within an SOS) and dataset specific algorithm parameters.

*Generic fusion: IT-INNOV WPS case study*

The IT-INNOV WPS infrastructure supports generic fusion at two levels (see figure 2.9). First a set of Python scripts provides metadata driven pre-processing and post-processing. Second a fusion framework provides configurable plug and play (via DLL's) support for new algorithms, conversions (e.g. units) and transformations (e.g. coordinates).

The pre and post-processing Python scripts provide metadata driven self-configuration based on the dynamically selected data source SOS/FTP and dataset provided by that data source. Each fusion observed target property (e.g. 'windspeed') is selected at runtime and the O&M metadata obtained from each SOS/FTP data source is used to identify the correct sensor value, associated unit(s) and sensor accuracy information. Syntax checking is driven from the unit metadata provided. Post-processing scripts use the metadata provided in the input (e.g. using the same units) when generating both CSV and O&M formatted result sets.

The Python scripts themselves are designed to be either top level master scripts or generic sub-scripts. The top level master scripts are formulaic in design and can be automatically generated to semi-automate fusion deployment.

The fusion framework provides a simple framework with access to a bought in third party numerical library. The framework supports plug-in DLL's for various common mathematical tasks such as unit conversion, coordinate transformation and fusion algorithms. The idea is that over time new generic algorithms will be plugged in and the metadata and conversion/transformation services re-used.

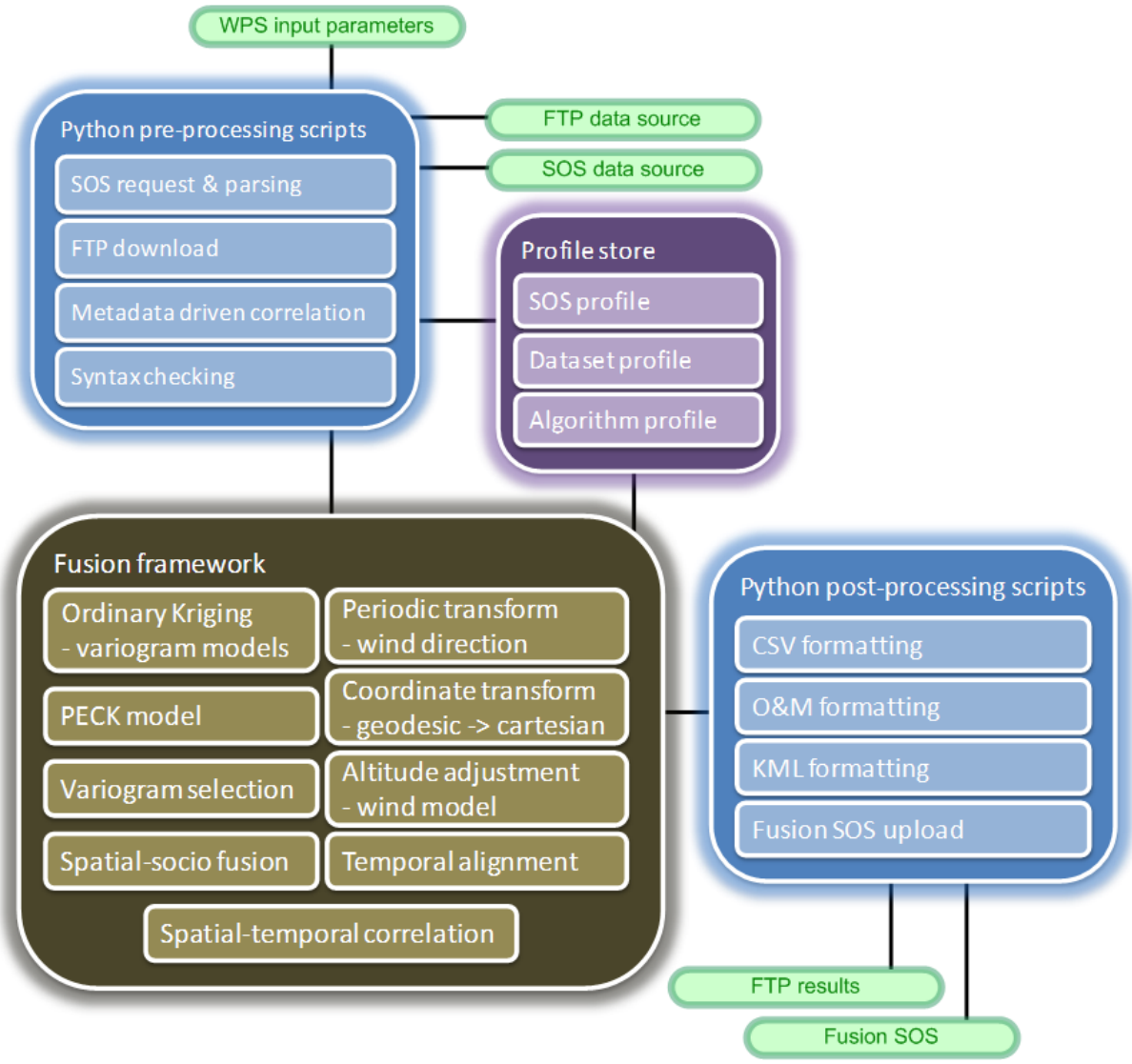



*Figure 2.9. Generic fusion design: IT-INNOV case study*

In SANY this fusion framework is our 'low-level' approach to the objective of a Generic Fusion and Modelling engine (GFME). The 'high-level' approach can be seen by the use of WPS and SPS standard services. The two approaches together provide a powerful approach to developing a generic fusion capability that can provide processing capabilities to a variety of OGC-enabled web services.

## 2.10. Intelligent configuration and deployment of processing

Prior to developing a new algorithm, or configuring an existing generic algorithm, a fusion assessment must be performed on the dataset(s) in question to look at its characteristics and come to a view on both the suitability of candidate algorithms and the specific configuration settings a chosen algorithm should use to optimize performance. This fusion assessment is normally a manual process performed by a fusion expert, looking at the format, quality, statistical properties (e.g. spatial density) and domain relationships between observed properties.

In SANY we have analysed the fusion assessment and offline setup process and tried to model some of the knowledge and expertise required for the algorithms we have developed. To help the offline processes of SOS, dataset and algorithm profile setup domain specific tools are required that can elicit expert knowledge about the domain and perform analysis tasks on datasets that will be available. To help the online process metadata driven pre-processing can be used in combination with dataset profiles to reduce the amount of manual data handling that is required to fulfil application user fusion requests.

*Fusion assessment toolset: IT-INNOV case study*

The fusion assessment toolset has been developed by IT-INNOV. A set of tools have been created, some integrated into a fusion WPS for online use and some created as standalone tools to help offline setup of the fusion WPS.

For online support profile driven automated pre-processing of datasets allows automatic correlation of 'plug and play' sensor datasets by the WPS; this means users can request fusion of datasets on-demand. The kriging spatial interpolation WPS process also supports automated variogram calculation based on configurable algorithm profiles, allowing profiles to be chosen on-demand to suit on-demand dataset domain characteristics (wind, ground displacement etc.).

To help offline setup a SOS crawler utility has been created to automatically extract SOS and dataset profiles from an unknown SOS by using metadata returned from SOS responses. An experimental tool for multi-region kriging has also been developed, making use of spatial region descriptions of known areas of spatial cohesion to improve the kriging results. Regions are created by automatic segmentation of either NASA space shuttle altitude data OR segmenting manually drawn Google Earth polygons indicating regions. This provides a way for domain experts to input explicit knowledge about spatial areas of interest.

## 2.11. Visualization of coverage

Fusion results often consist of spatial data (along a sampling surface) that must be visualized in order to be properly interpreted by decision makers. This data includes both the fused values and their associated uncertainty.

The main mechanism used in SANY to visualize spatially distributed data is the drawing of coloured contours (isolines) as illustrated in figure 2.10.

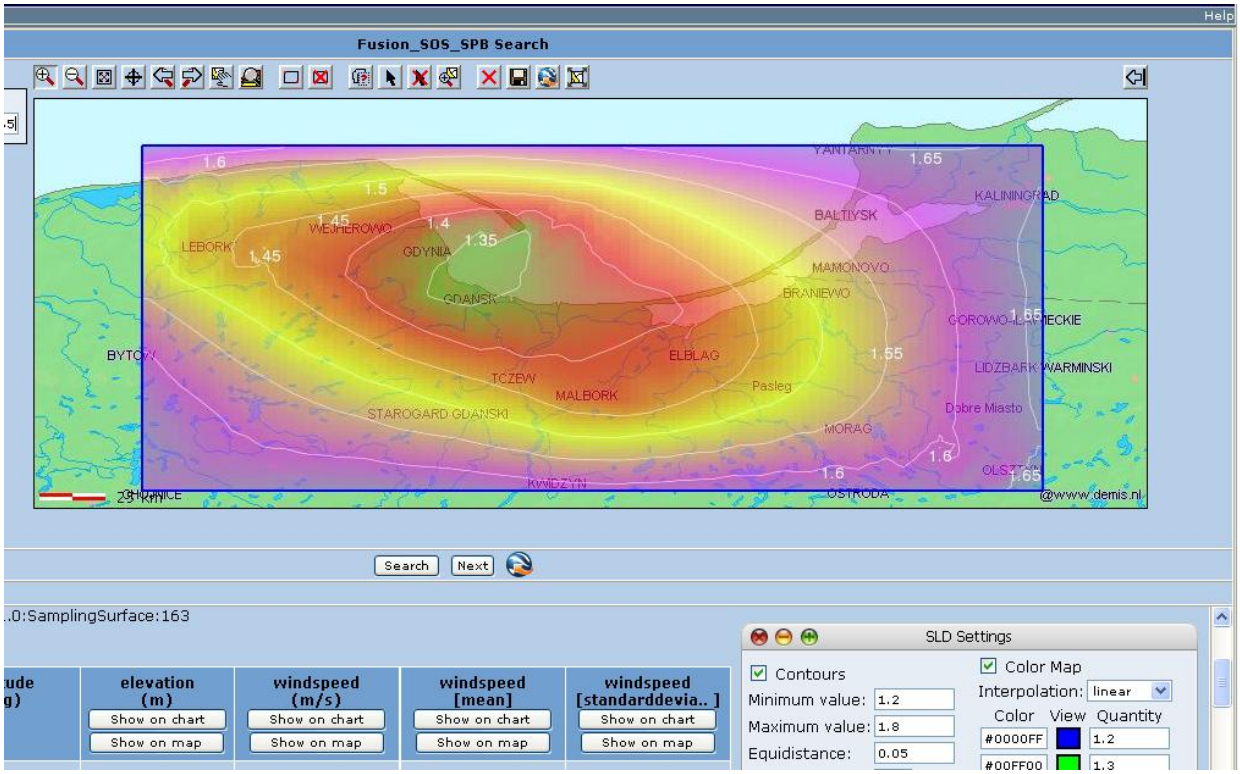



*Figure 2.10. Display of Contours*

The generation of the contours is performed by the Map and Diagram service (MDS) which is a standard Styled Layer Descriptor (SLD) enabled Web Map Service (WMS) with support for extended symbolizers.

Through the Styled Layer Descriptor mechanism, user-defined symbolization of contours becomes possible. The Map and Diagram service is able to build contours from data stored on an FTP server in various formats (e.g. ASCII grid or SOS Observation). The Map and Diagram service is also able to access the spatial data directly from an SOS.

As illustrated in the figure below, the Styled Layer Descriptor description allows for configuration of various parameters such as the line colour, the line width, the equidistance, the interpolation method, the transparency, the colour bands, the labels, etc.

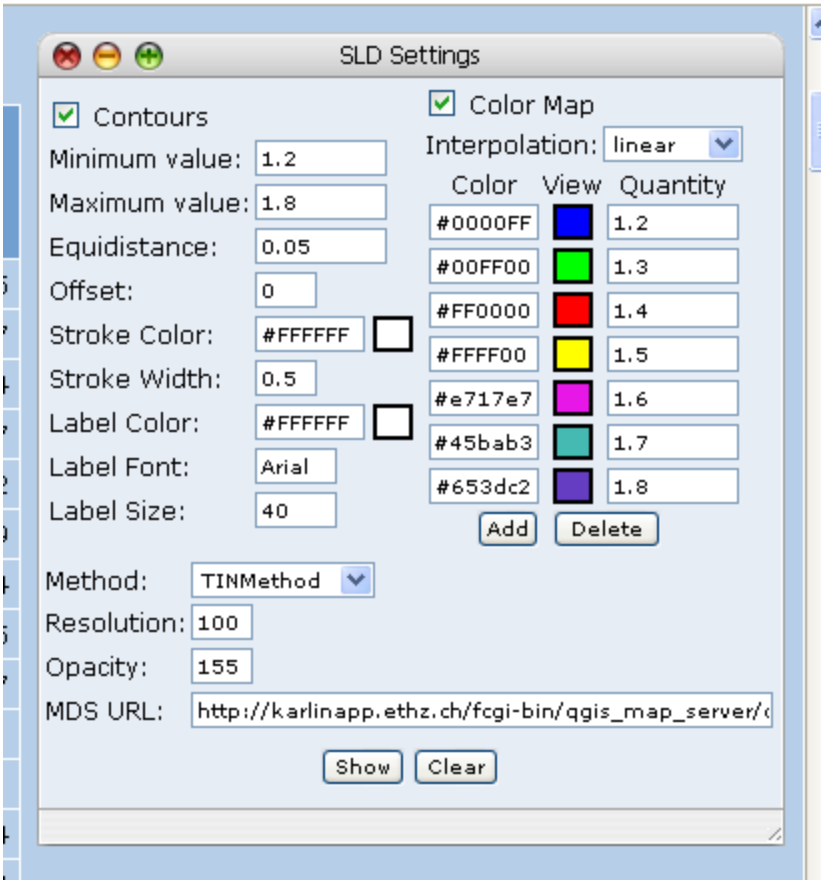



*Figure 2.11. Styled Layer Descriptor (SLD) Configuration*

Figure 2.10 and 2.11 are examples of screen captures from a WPS SSE client used for spatial interpolation of wind speed in the bay of Gdansk. Both the SOS and WPS SSE clients support the display of contours for spatially distributed data.

As a secondary visualization technique Google Earth KML displays of spatial fusion result sets are generated by the IT-INNOV WPS. These can be seen in Figure 2.12 for spatial interpolation over the Gdansk and Dresden areas. Graphs indicate values (e.g. wind speed), heat maps indicate error (e.g. kriging variance), coloured arrow maps indicate value and direction (e.g. wind speed and direction) and yellow pins indicate sensor locations.

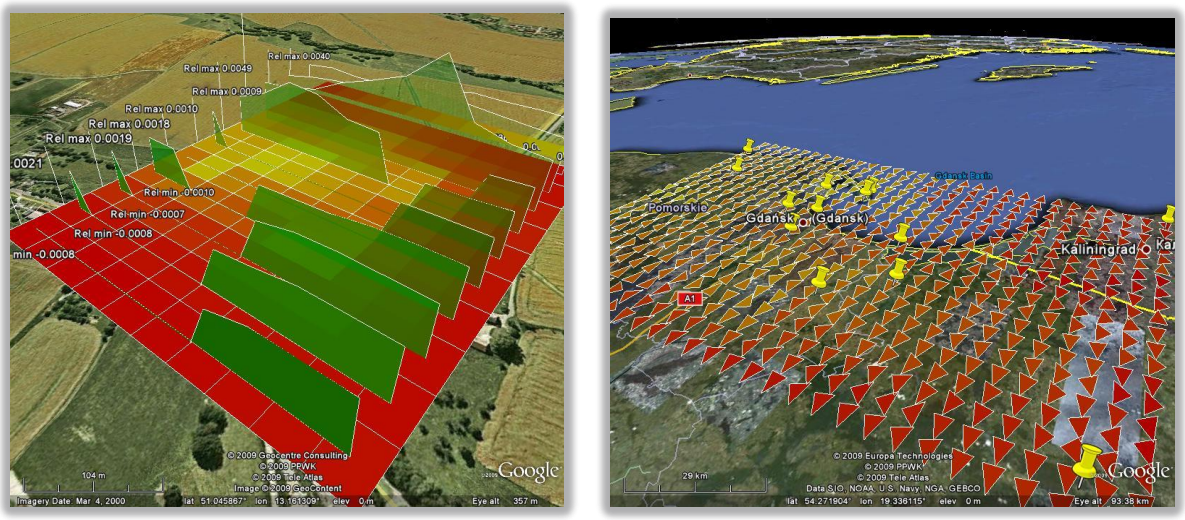



*Figure 2.12. KML visualization of a kriging spatial result set*

# 3. Architectural designs

## 3.1. Service deployment

## 3.2. Case study 1 - Fusion WPS design [IT-INNOV]

### 3.2.1 Architecture

The system architecture developed and deployed by IT-INNOV is based on a central WPS processing unit which clients can access via a web service protocol. Behind the service interface are support tools for pre and post-processing and a central generic algorithm set. Additional support comes with the fusion assessment tool and its web interface to semi-automate the assessment process. Input datasets are obtained either from remote SOS's or via a FTP site. Final fusion results are stored on an FTP site and uploaded to the IITB Fusion SOS. Figure 3.1 shows the IT-INNOV full WPS deployment.

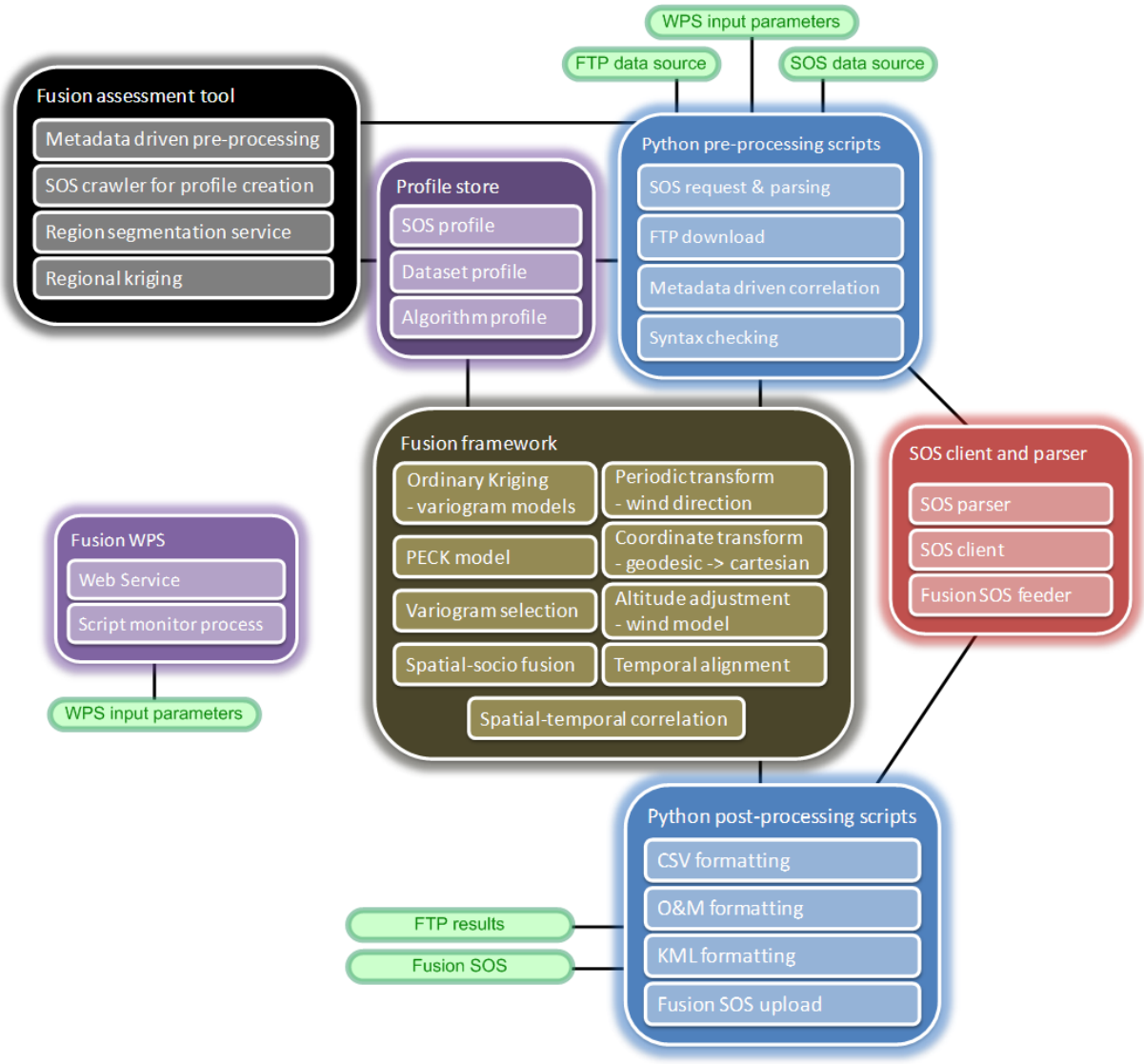



*Figure 3.1. Deployment of the IT-INNOV Fusion WPS and support modules*

Infrastructure developed within SANY:

- Fusion WPS (v1.0.0) **[SP3.Module.ITINNOV.1]**

  This WPS consists of a C# .NET web service software, supporting v1.0.0 of the WPS standard, a Python script invocation manager and a C# script monitor tool to asynchronously manage multiple parallel fusion process lifecycles and generate the appropriate WPS status results for each stage of the processing.

- Python pre and post-processing scripts **[SP3.Module.ITINNOV.2]**

  Master scripts for primary processing and generic pre and post-processing subscripts that the master scripts invoke. The master scripts are designed to be machine generated as part of the semi-automated fusion assessment.

  Pre-processing scripts include input data management (SOS client download, FTP download), metadata driven dataset correlation and metadata driven syntax checking.

The O&M and KML result formatting software is written using Python and XSLT and integrated into the post-processing script. This is in addition to post-processing scripts to manage the Fusion SOS data upload and FTP result data management.

- Fusion framework **[SP3.Module.ITINNOV.3]**

  This module is new to V2 and supports plug-in DLL's for algorithms, conversion tools and transformation tools. The framework is linked with a third party numerical library to provide an environment where algorithms can be easily developed. Third party algorithms (e.g. Matlab compiled EXE's) can be plugged in via a DLL wrapper to an external exec call.

- SOS client and parser (v0.0.0, v1.0.0, IITB Web Genesis) **[SP3.Module.ITINNOV.4]**

  A set of C# tools have been developed to support dataset download from SOS's (SOS v0.0.0 and v1.0.0 supported), parsing of the SOS responses and extract of CSV data and metadata and client driven upload of O&M fusion results to the IITB Web Genesis Fusion SOS.

- Fusion assessment toolset **[SP3.Module.ITINNOV.5]**

  The fusion assessment toolset is a combination of tools integrated into the WPS and separate offline setup tools. Online automated metadata driven pre-processing is integrated into the WPS. Offline tools have been created for profile creation by SOS 'crawling' and tools for experts to specify regions of spatial cohesion (manually drawn Google Earth polygons or extracted from NASA altitude data). This provides a way for domain experts to input explicit knowledge about spatially cohesive areas of interest to further improve kriging algorithm results.

- Algorithm: Orginary kriging (SP5, SP6) **[SP3.Module.ITINNOV.6]**
- Algorithm support: Variogram tuning **[SP3.Module.ITINNOV.7]**
- Algorithm support: Elevation correction **[SP3.Module.ITINNOV.8]**
- Algorithm support: Time synchronization **[SP3.Module.ITINNOV.9]**
- Algorithm: PECK model (SP6) **[SP3.Module.ITINNOV.10]**
- Algorithm: Spatial-social fusion (SP6) **[SP3.Module.ITINNOV.11]**
- Algorithm: Spatial-Temporal correlation (SP5) **[SP3.Module.ITINNOV.12]**

### 3.2.2 Module definitions

*Fusion WPS (v1.0.0)*

The IT-INNOV WPS is a C# .NET web service that provides a SOAP protocol to access the WPS operations as per the v1.0.0 specification. Figure 3.2 provides a description of the WPS interface, and appendix A provides examples of the XML request and responses that are used. This WPS supports reference and literal output types and returns status information via an XML response document located on an FTP site. Final result sets are also available on an FTP site as well as an O&M formatted result set uploaded into the IITB Fusion SOS.

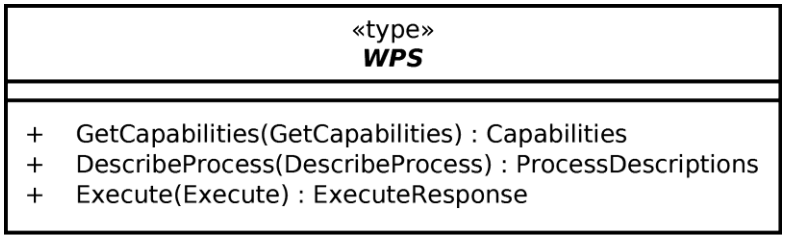



*Figure 3.2. WPS module interface*

When a new Execute() operation is invoked a Python script process is launched along with a monitor process to asynchronously check on the progress of the fusion result (via a progress XML file). The monitor process continuously writes to the WPS status report file to report progress and eventually report the location of the output files.

The Fusion WPS itself can be deployed using an ANT script and collection of configuration files. Being stateless new WPS processes can be added to the live WPS, simply by adding new profiles to the profile store. The fusion assessment tool allows this process to be semi-automated, helping to reduce the cost of deployment.

*Python scripts, fusion framework and the profile store*

All pre and post-processing functionality is driven from a set of Python scripts. There is a master Python script for each WPS process, and these master scripts are able to invoke a set of generic sub-scripts that perform the pre and post-processing work. The master scripts are deliberately designed to support machine generation, via an XSL transformation, allowing the fusion assessment tool to do this.

The fusion framework executes plug-in DLL algorithms, conversion and transformation operations based on an XML configuration file. This algorithm configuration is stored in the profile store along with SOS profiles (version, namespaces, URL etc) and dataset profiles (correlation keys, special observation properties etc). As such a single generic Python script is needed to run the framework.

All dataset specific algorithm and pre-processing configuration is stored in a profile store (a directory containing XML profile files) on the server machine. One of the most important WPS input parameters is the profile name that should be used for the processing session, allowing clients to dynamically choose from a set of pre-configured SOS's, datasets and algorithms. This profile mechanism and the metadata driven automation in the pre-processing scripts allows us to 'plug and play' datasets at runtime as long as an initial profile has been configured.

The following master scripts will be available in V2:

- Kriging spatial interpolation
- PECK model
- Spatial-socio fusion
- Spatial-temporal correlation

The following generic sub-scripts will be available in V2:

- FTP response download
- SOS GetCapabilities request
- SOS DescribeSensor request
- SOS GetFeatureOfInterest request
- SOS GetObservation request
- Metadata driven CSV response correlation
- Metadata driven syntax checking
- Displacement delta calculator
- Fusion framework invoker
- Result formatter (O&M)
- Result formatter (KML)
- SOS fusion result upload
- Helper script - functions
- Helper script - constants

*SOS client*

We have developed two SOS clients, one for a normal SOS containing datasets (v0.0.0 and v1.0.0 supported) and one for the Web Genesis fusion SOS. The first SOS client is able to issue SOS requests (GetCapabilities, DescribeSensor, GetFeatureOfInterest, GetObservation), parse the results and save CSV files containing the data extracted. A sparse matrix population parsing pattern is used to be robust to missing data and multiple ways to describe the same data. The second SOS allows O&M result data to be uploaded to the IITB Fusion SOS, providing long term storage capability in a SWE compliant service.

*Fusion assessment tool*

The fusion assessment toolset is a combination of tools integrated into the WPS and separate offline setup tools. Profile driven automated pre-processing of datasets and correlation of sensor data is integrated into the WPS along with automated variogram calculation for the kriging algorithm based on configurable algorithm profiles. A SOS crawler utility has been created to automatically extract SOS and dataset profiles from an unknown SOS by using metadata return by the SOS responses. An experimental tool for multi-region kriging has been developed, including a spatial analysis tool to create regions for experimental multi-region kriging. Regions are created by automatic segmentation of either NASA Space Shuttle altitude data OR segmenting manually drawn Google Earth polygons indicating regions. This provides a way for domain experts to input explicit knowledge about spatially cohesive areas of interest.

*Algorithm and support tools*

Details of the algorithms are defined in section 3.5.

### 3.2.3 Module interactions

*WPS process discovery*

The SANY catalogue service will automatically extract metadata from the result of a OGC WPS DescribeProcess response. This allows the catalogue to automatically harvest WPS metadata as required, keeping its cache of service metadata up-to-date. Figure 3.3 shows the sequence of operations required for a client to obtain service details about a new WPS.

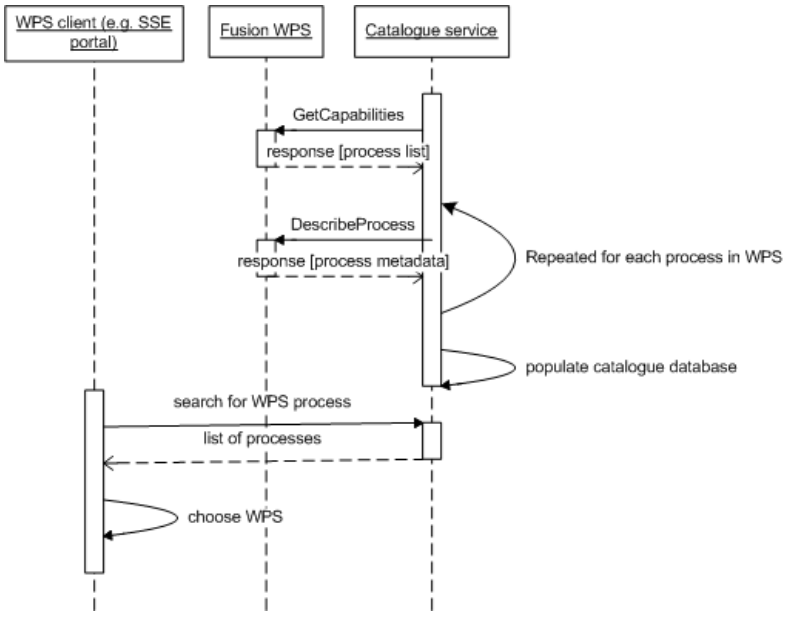



*Figure 3.3. WPS process discovery sequence*

*WPS process execution*

When a client requests that a fusion process is executed a number of different services and modules are invoked as a result. The client does not see this complexity, and just needs to poll

the status file and download the final result data. Figure 3.4 shows the full sequence of actions that results from invocation of the IT-INNOV WPS.

The first thing to happen is the WPS spawns two new processes, one for the selected fusion process master script and one for the script monitor that will regularly parse the master scripts progress file. As the master script runs it will regularly report progress via a progress.xml file. The script monitor polls this file and generates, on the FTP site, updates to the WPS status file. The client can then download this file at will to see what the job progress is and at the end where to download the result files from.

Once started the pre-processing subscript will be invoked by the master script. This will cause the SOS client to run on all input SOS's, and also any available cached SOS response files to be downloaded from remote FTP sites. These response files are parsed and CSV data stored on the local FTP site in a working directory. There will be a single CSV file per SOS response and all these CSV files are correlated together into a single CSV input file. This file is syntax checked and cleaned up ready for input into the algorithm.

The algorithm sub-script configures and invokes the fusion framework to run the appropriate fusion and conversion/transformation DLL's on the CSV input data provided. Depending on which algorithms are run different result sets will be produced. However all results are stored in CSV format ready for post-processing.

The pos-processing subscript takes the CSV result data and formats it into O&M XML result data suitable for upload to the IITB Fusion SOS. An appropriate SOS client feeder does this. Extra formats such as KML are also generated and the final result data URL's reported via the progress.xml file. The script handler will pickup these URLs and move the output data to an output directory on the FTP site and create a final status report file for the client to pickup.

Finally the client reads the final status report and downloads all the available result files. If required the client can also get access to the results via the IITB Fusion SOS, useful if the client knows how to handle SWE services.

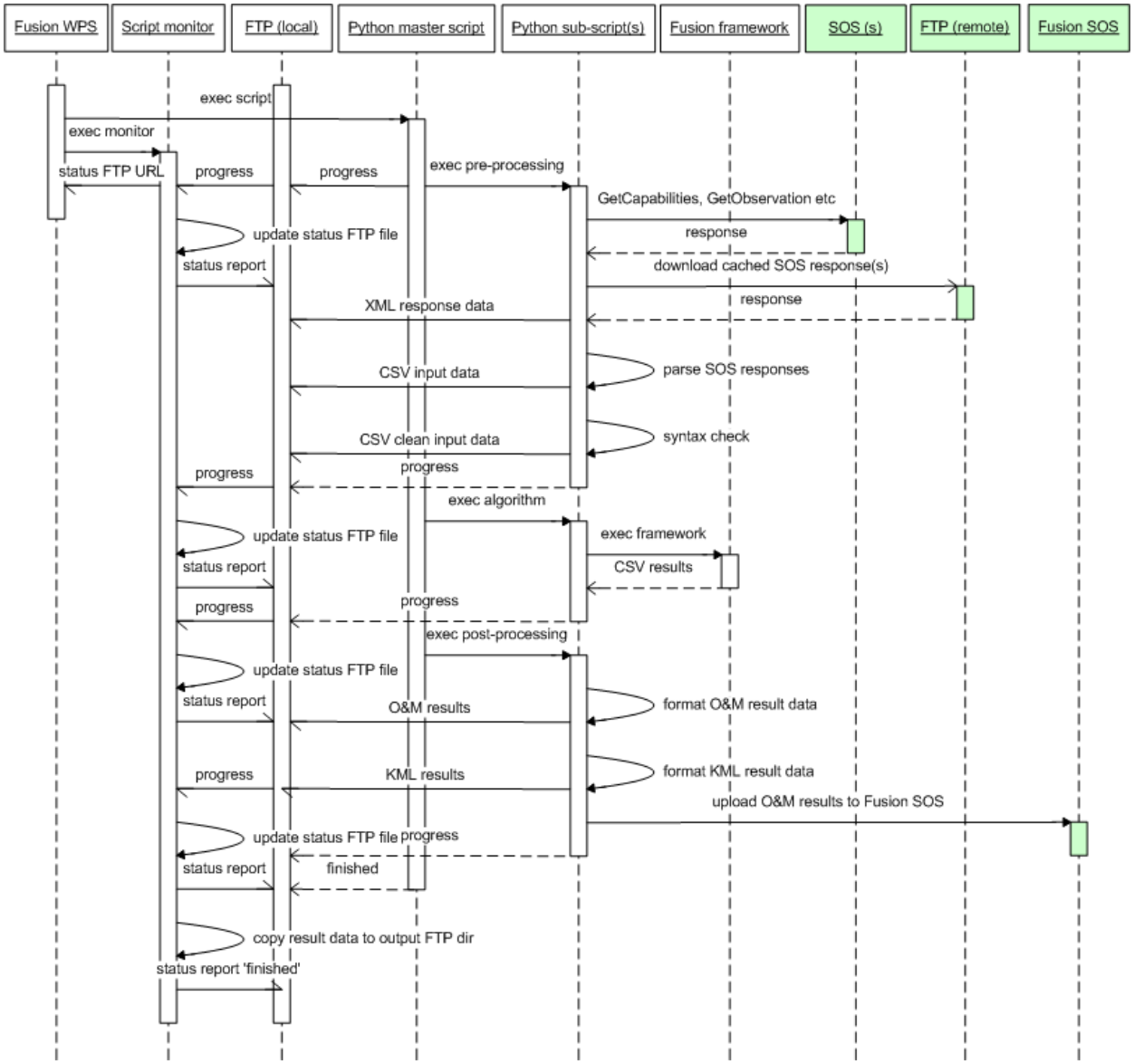



*Figure 3.4. WPS process execution sequence*

## 3.3. Case study 2 - Fusion WPS / SPS design [BMT]

### 3.3.1 Architecture

The system architecture developed by BMT is based around a BMT developed pre-existing framework for OGC web services. This framework has allowed for the rapid creation of OGC compliant web services with a minimum of effort allowing the majority of time to be spent on the models and interactions between services.

Infrastructure developed within SANY

- Fusion SPS server **[SP3.Module.BMT.1]**
- Fusion WPS v1.0.0 interface **[SP3.Module.BMT.2]**

Fusion procedure developed within SANY

- Algorithm: Multiple Regression v1 (SP5) **[SP3.Module.BMT.3]**
- Algorithm: Multiple Regression V2 (SP5) **[SP3.Module.BMT.4]**
- Algorithm: Artificial Neural Network (SP5) **[SP3.Module.BMT.5]**
- Algorithm: Probability Index (SP5) **[SP3.Module.BMT.6]**
- Algorithm: Time Series (SP4) **[SP3.Module.BMT.7]**
- Algorithm: State Space (SP4, SP6) **[SP3.Module.BMT.8]**

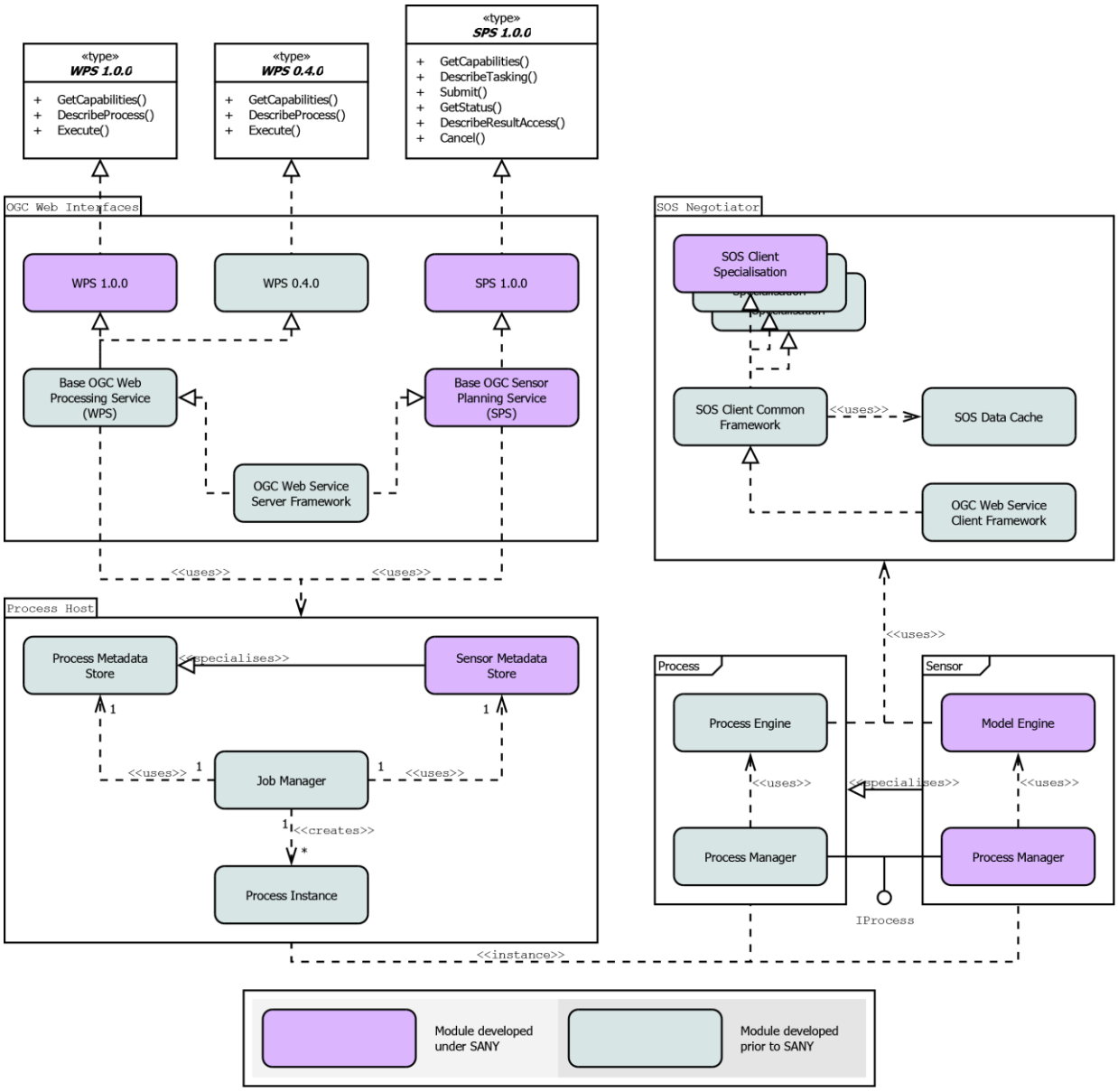



*Figure 3.5. Modules in the BMT architecture*

Under the SANY project, BMT has updated the interface version of their commercial WPS (to v1.0.0) to come into line with other SP3 implementations, and also created an instance of a new process type – the Sensor Planning Service (v1.0.0). Both of these are used to execute fusion modelling processes for different use cases and are built around BMT Cordah's "Benbow" OGC web service framework. Other services built on this framework (not under the SANY project) include a Transactional Web Feature Service and a Web Map Service.

The metadata for the algorithms ("Processes" and "Sensors") is stored in such a way as to facilitate the automatic generation of WPS and SPS output documents, and to ensure a generic framework exists for the addition of any possible future algorithms.

Not included in the diagram is the use of a Web Feature Service for the storage of algorithm parameters for the Bathing Water Quality pilot. The WFS implementation used is the open-source GeoServer version 1.7.3 retrieving data from a PostGIS database. These are completely stock versions of the software.

### 3.3.2 Module definitions

*WPS 1.0.0 Interface*

BMT Cordah's Web Processing Service, as it existed before the SANY project, conformed to the then-latest specification version 0.4.0. When version 1.0.0 of the specification was released by the OGC, SP3 partners agreed on a move to this newer version as it included clarifications to the standard. All partners moved to the new version to ensure interoperability between servers and clients.

The WPS implementations support the mandatory GetCapabilities, DescribeProcess and Execute calls. Using these three calls it is possible to discover what process types the service provides, retrieve metadata on those processes, and then request the process be executed. WPS does not appear to have been designed with long-term models or processes in mind as it lacks a method to modify or stop an existing process and the method used to monitor the status of an ongoing process ("@statusLocation" in an ExecuteResponse document) is optional.

The OGC Web Service Server Framework is developed in a mixture of C++ and C# under Microsoft .Net and the WPS framework is built directly on top of this in C#. Service instances are hosted in Microsoft IIS.

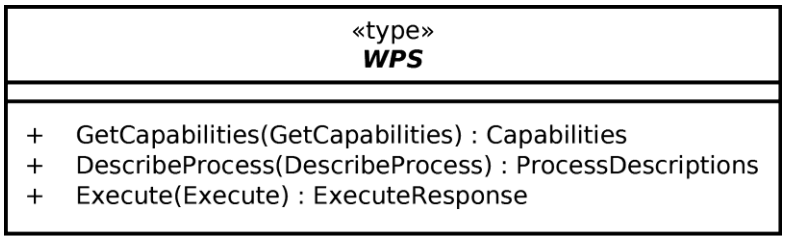



*Figure 3.6. BMT WPS module interface*

*Sensor Planning Service*

By building on BMT Cordah's existing OGC Web Service Server Framework, a Sensor Planning Service based on the v1.0.0 specification was created.

To simplify the development of this exemplar, only a subset of the requests specified in the SPS specification are implemented. The implemented requests include all of the mandatory requests and a subset of the optional requests.

| **Request (mandatory core operations are in bold)** | **Function** | **Implemented** |
|---|---|---|
| **GetCapabilities** | This operation allows a client to request and receive service metadata (or Capabilities) documents that describe the abilities of the specific server implementation. This operation also supports negotiation of the specification version being used for client-server interactions. | Yes |
| **DescribeTasking** | This operation allows a client to request the information that is needed in order to prepare an assignment request targeted at the assets that are supported by the SPS and that are selected by the client. The server will return information about all parameters that have to be set by the client in order to perform a Submit operation. | Yes |
| GetFeasibility | This operation is to provide feedback to a client about the feasibility of a tasking request. Dependent on the asset type facade by the SPS, the SPS server action may be as simple as checking that the request parameters are valid, and are consistent with certain business rules, or it may be a complex operation that calculates the utilizability of the asset to perform a specific task at the defined location, time, orientation, calibration etc. | No |
| **Submit** | This operation submits the assignment request. Dependent on the facade asset, it may perform a simple modification of the asset or start a complex mission. | Yes |
| GetStatus | This operation allows a client to receive information about the current status of the requested task. | Yes |
| Update | This operation allows a client to update a previously submitted task. | No |
| Cancel | This operation allows a client to cancel a previously submitted task. | Yes |
| **DescribeResultAccess** | This operation allows a client to retrieve information about how and where data that was produced by the asset can be accessed. The server | Yes |

| | | |
|---|---|---|
| | response may contain links to any kind of data accessing OGC Web Services such as SOS, WMS, GVS or WFS. | |

**Table 3.1. BMT SPS operation support matrix**

These functions give enough control over the processes to make then useful for the example uses they are put to in the SANY project.

The OGC Web Service Server Framework is developed in a mixture of C++ and C# under Microsoft .Net and the SPS framework is built directly on top of this in C#. Service instances are hosted in Microsoft IIS.

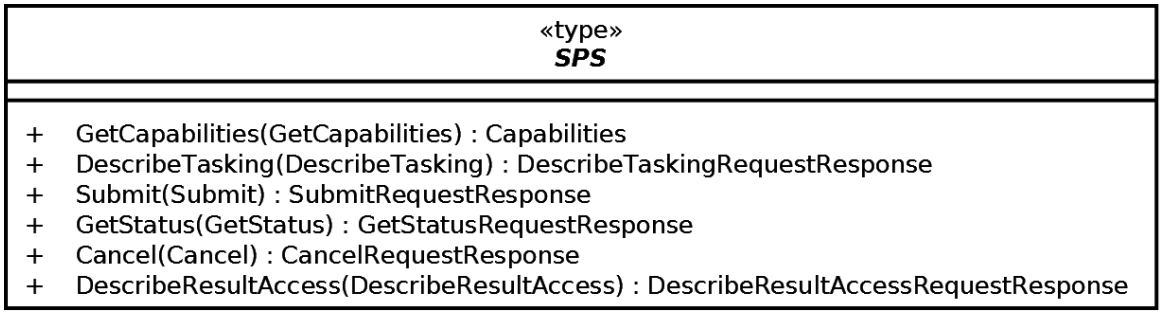



*Figure 3.7. BMT SPS module interface*

*SOS Client Specialisations*

Although the OGC standards were designed to allow interoperability between systems written by differing authors, in practice very few service implementations are identical enough to allow for a truly generic client. As a result the BMT Cordah OGC Web Service Client Framework was built to allow clients to effectively detect the "dialect" of a particular language spoken by a service. This allows the client to make an attempt to speak to a server using the "dialect" the server uses, while also giving the ability to add new "dialects" simply by dropping a dll into the appropriate folder.

A number of the SOS instances in the SANY testbed required changes to the standard versions of the client already deployed.

This is also the system used as an interface to the fusion WPS processes supplied by IT-INNOV. In this case, the SOS specialisation is able to detect the absence of the requested data from the SOS and spawn a WPS call to request the data be produced.

The OGC Web Service Client Framework is developed in C# under Microsoft .Net and the SOS framework is built directly on top of this in C#. While specialisations can be written in any .Net language, the ones developed under SANY have all been developed in C#.

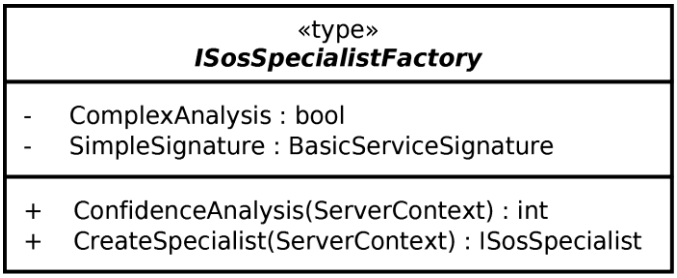



*Figure 3.8. BMT SOS client module interface*

### 3.3.3 Module interactions

*Data Discovery*

The web application developed in the pilot uses the SANY catalogue service, as supplied by Fraunhofer IITB, to search for SOS instances containing the data that is needed for the current simulation type.

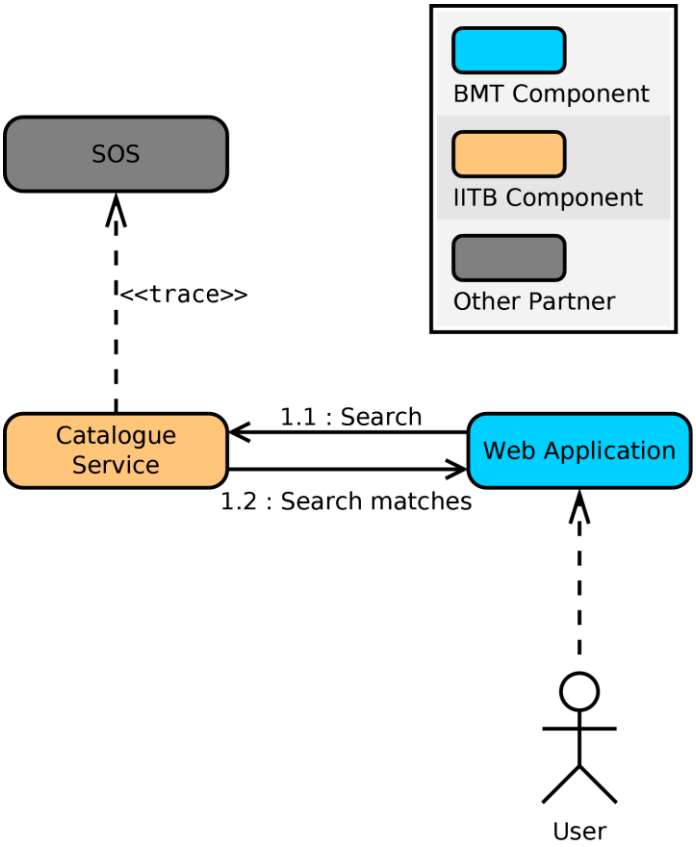



*Figure 3.9. Basic catalogue interaction*

Module interactions in service discovery are very simple. In the SP5 pilot applications, service discovery happens at the user interface, allowing the user to make a decision on the data sources to use. Automatic choice of the data sources to use is outside the scope of these pilots.

*Basic WPS Execution*

The BMT fusion WPS has two main methods of operation. It can either be passed the data for it's simulation directly, or it can retrieve the data from the specified SOS or WPS. Here we cover the first method.

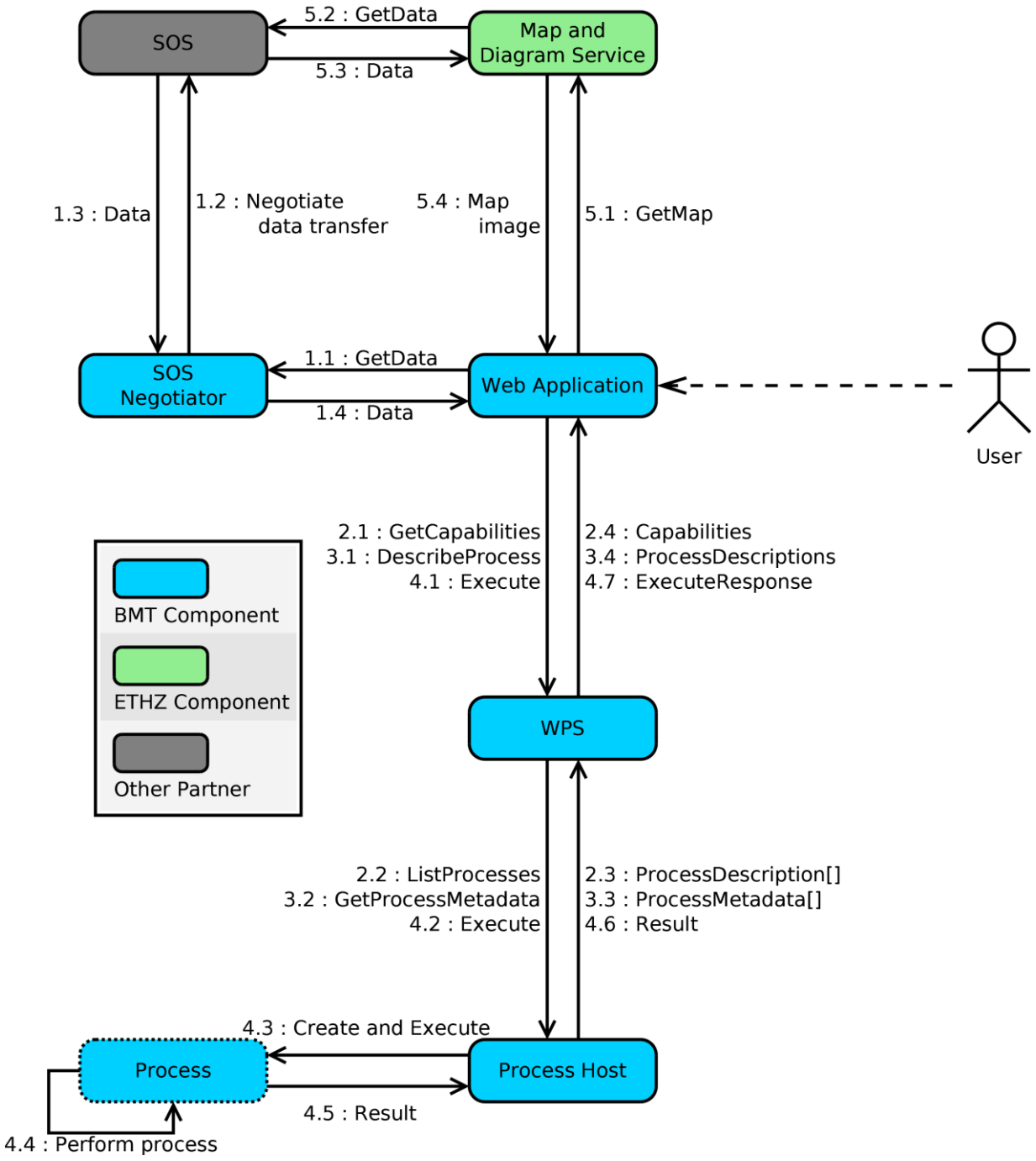



*Figure 3.10. Basic WPS execute interactions*

Assuming a data source has been chosen, the user interface is able to retrieve data for the simulation. This data can then be encoded into the WPS execute request.

Once the simulation has been executed, the ETHZ Map and Diagram Service (MDS) can be used to visualise the both the model results and the contributing data.

*WPS Execution with SOS Retrieval*

This section covers the interactions involved in executing a simulation where the WPS has been told to retrieve the data from a SOS itself.

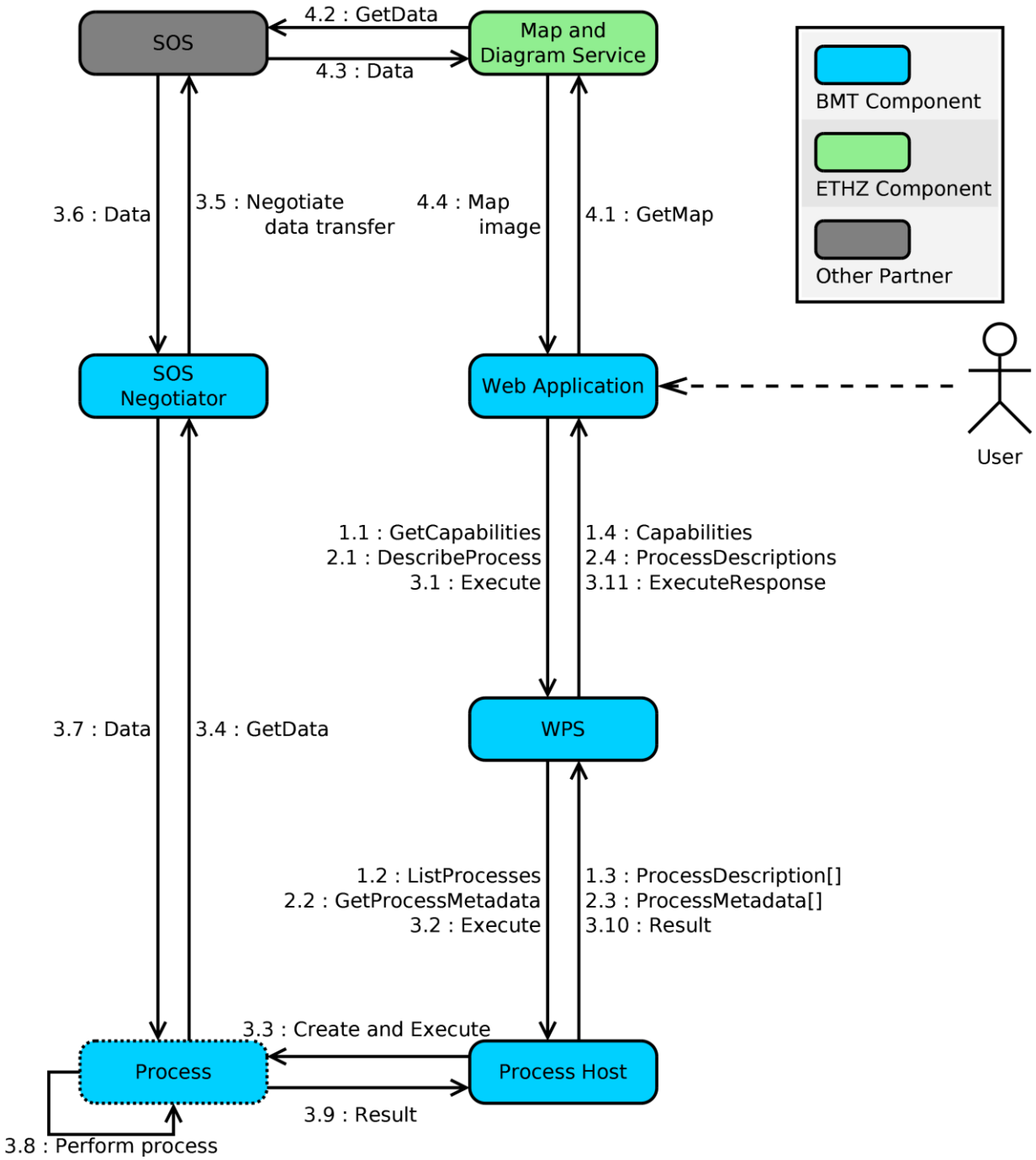



*Figure 3.11. WPS execute interactions with SOS data retrieval*

*Data Retrieval from IT-INNOV Fusion Services*

The fusion services supplied by IT-INNOV in the SANY project provide more functionality than can easily be supplied by any single OGC web service. For this reason, communication with these services needs to be handled in a more structured way than would be necessary for disparate data sources.

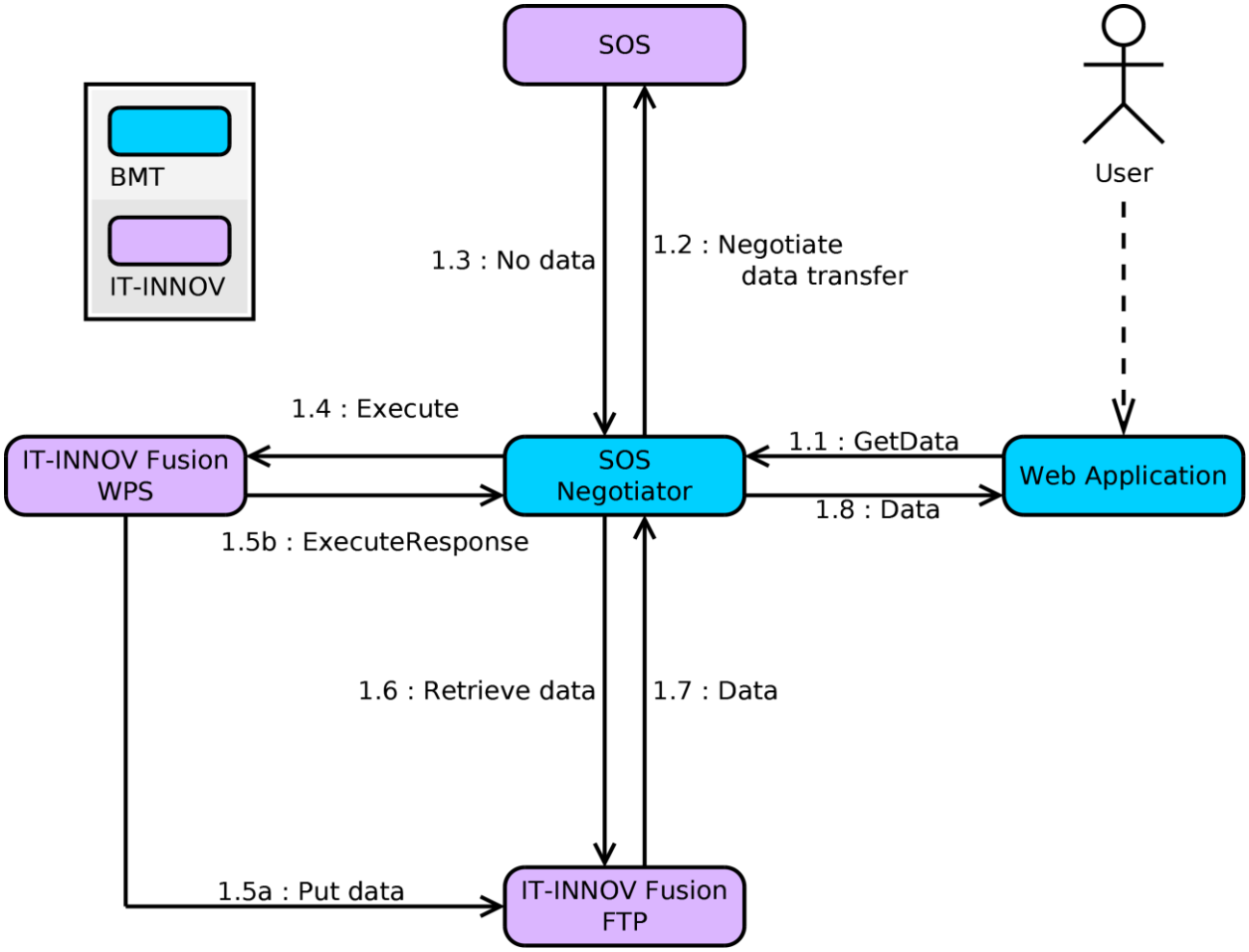



*Figure 3.12. Interaction with IT-INNOV fusion data services*

## 3.4. Case study 3 - Fusion SOS / SPS design [IITB]

### 3.4.1 Architecture

Infrastructure developed within SANY

- Fusion SOS server **[SP3.Module.IITB.1]**
- Fusion SPS server **[SP3.Module.IITB.3]**

Fusion procedure developed within SANY

- BME (Bayesian Maximum Entropy) **[SP3.Module.IITB.2]**

Fraunhofer IITB has realised a special Sensor Observation Service (compliant to OGC SOS v1.0) in its testbed, called a Fusion SOS which is used for storage of fusion results (coverages) and associated metadata. It also has implemented a Sensor Planning Service (compliant to OGC SPS v1.0) which is used to parameterize and execute generic fusion algorithms. Both the Fusion SOS and SPS are implemented using IITBs *WebGenesis* information management system [WEBGENESIS].

Internally, these algorithms are described as a fusion process (including inputs, parameters, algorithm, outputs etc.) inside so called “assets” (an available means of collecting information; synonyms: sensor, simulation, process) in *SensorML* and uploaded to the Fusion SPS. All available processes (in SPS also called “tasks”) are advertised in the capabilities of the Fusion SPS. A client can then request a description of a specific process (i.e. the fusion algorithm) from the Fusion SPS to learn about the inputs and parameters needed for execution of the process. The input data of a fusion process typically consists of a SamplingFeature (e.g. a SamplingSurface) describing the shape of the spatial or spatial-temporal interpolation, and aggregated sensor data obtained from several SOS (this is the “source data” that shall be interpolated by the fusion process). Also, in most cases additional parameters used for configuration of the fusion process itself must be supplied (though it is conceivable that the SensorML description of the fusion process contains default parameter values that make this step optional). After successful completion of a fusion process run, the resulting coverage (encoded as O&M observation), along with its associated SamplingFeature (encoded as O&M SamplingSurface) and the used procedure (the fusion process with its supplied inputs/parameters, encoded as SensorML) are stored in the Fusion SOS for later retrieval / traceability. Table 3.2 shows a summary of the general concepts used in Fusion SOS/SPS.

| General concept | Fusion SOS usage | Fusion SPS usage | Encoded as | Description |
|---|---|---|---|---|
| **Sensor** | *Procedure* (for provenance/ traceability)<br><br>A procedure is referenced by the Observation uploaded as the result of a successful fusion task | *Asset* (description of fusion proccess)<br><br>*Task* (description of fusion process during or after execution) | SensorML<br><br>(with values)<br><br>(without values) | An 'Asset' describes an available Fusion process in the SPS. It does not yet contain concrete values for the inputs/parameters of that process.<br><br>A 'Task' represents a fusion process during or after execution. It contains the concrete inputs/parameter values needed for execution. After successful completion, the task is uploaded (together with the result observation) to the Fusion SOS as a 'procedure' for provenance/traceability reasons. |
| **Feature of Interest** | For provenance/ traceability.<br><br>Referenced by the result observation | Input for an Asset (Fusion process), needed for execution of that process | SamplingSurface | The SamplingSurface is one particular input for fusion process (asset). After successful fusion run, it is uploaded (along with the result observation) to the Fusion SOS |
| **Observation** | Fusion result is stored here as Observation with related Feature of Interest, procedure and observation collection for input data | Result of a fusion task that has completed without failure. Contains the result data (e.g. the grid coverage) | O&M Observation | A successful fusion run produces an Observation containing the result data. It is then uploaded to the Fusion SOS for later retrieval. The Observation references the used SamplingSurface and procedure (SensorML of task) |

**Table 3.2. Summary of Fusion SOS/SPS concepts**

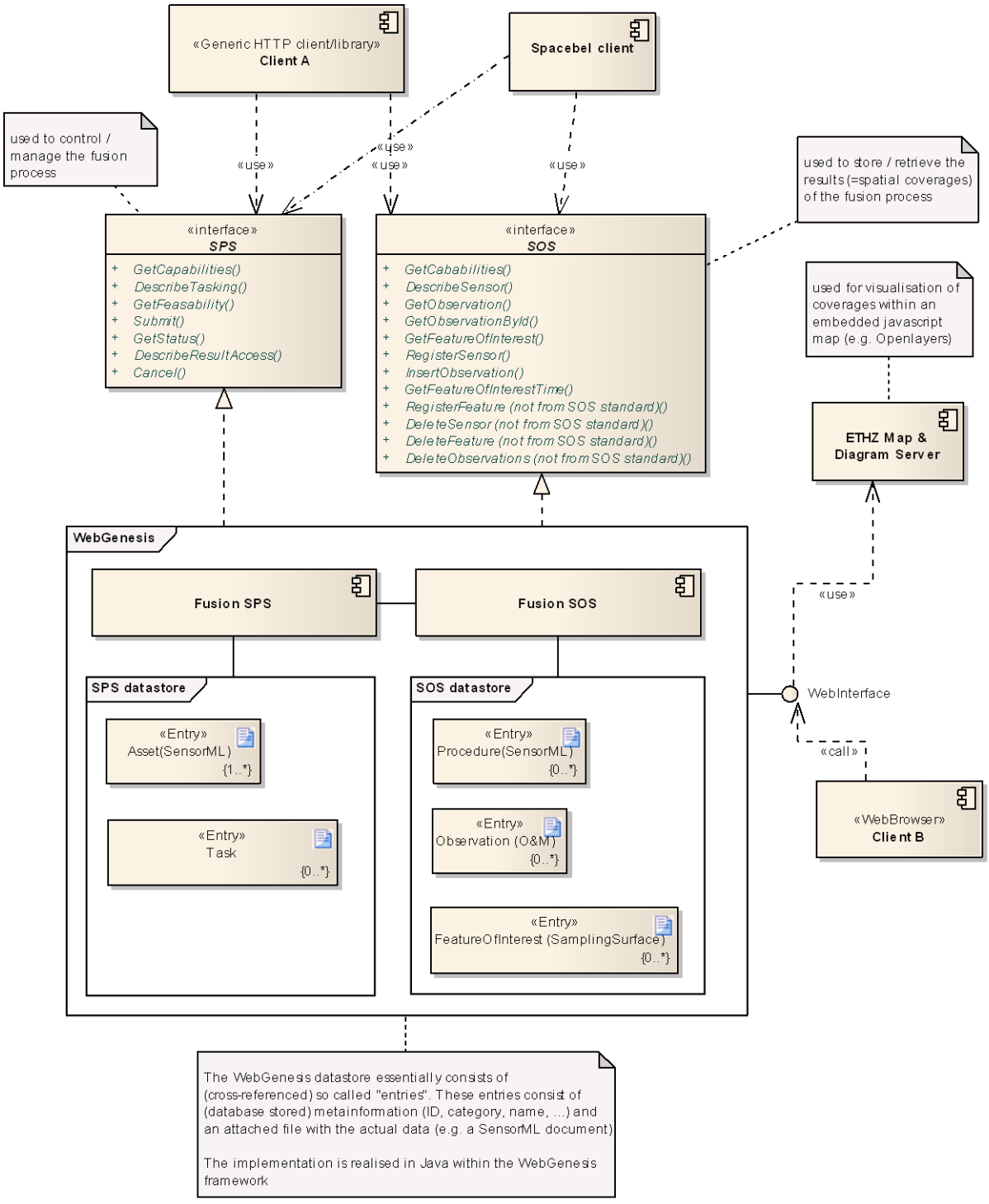



*Figure 3.13. Component diagram of Fusion SPS/SOS architecture*

### 3.4.2 Module definitions

*Fusion SOS server*

| SOS Operation (mandatory core operations are in bold) | Realisation in Fusion SOS |
|---|---|
| **GetCapabilities** | Lists service metadata (available observation offerings, service provider, etc.) |
| **DescribeSensor** | Returns description of fusion procedure in SensorML |
| **GetObservation** | Returns O&M Observation (or ObservationCollection) Supported constraints are:<br>• Feature of Interest<br>• Temporal filter: Time interval for the SamplingTime or ResultTime<br>• Spatial filter: Envelope (of type bounding box) for the SamplingFeature. If envelope and shape of the SamplingFeature intersect, then the relatedObservations of the complete shape of the SamplingFeature are returned.<br>• Procedure: returns relatedObservation of a procedure |
| RegisterSensor | Used to register a new procedure in the Fusion SOS. This operation will not be exposed to external clients. |
| InsertObservation | Used to insert a new relatedObservation in the Fusion SOS. This operation will not be exposed to external clients. |
| GetObservationById | The ID of an observation is assigned during the fusion process and stored by the information management system WebGenesis |
| GetFeatureOfInterest | Returns the SamplingFeature |
| GetFeatureOfInterestTime | Returns the time periods for which the SOS will return data for a given FeatureOfInterest. |

**Table 3.3. IITB SOS operation support matrix**

*Fusion SPS server*

| SPS Operation (mandatory core operations are in bold) | Realisation in Fusion SPS |
|---|---|
| **GetCapabilities** | Supported, returns Sensor- and Phenomenon-Offerings which list taskable sensors and their observed Properties (phenomena) |
| **DescribeTasking** | Supported, constraint is one or more sensorIDs; returns for each sensorID one taskingDescriptor which contains n InputDescriptors (where n is the number of configurable parameters for that sensor) |
| **Submit** | Supported, important constraints are sensorID, parameters and timeframe; the task (e.g. the fusion process, represented by the asset with that sensorID) is executed with the parameters that were submitted |
| GetFeasability | Supported, important constraints are sensorID, parameters and timeframe; this operation evaluates the feasibility of the constraints before the submit request |
| GetStatus | Supported, constraint is a taskID; returns the current status of that task |
| **DescribeResultAccess** | Supported, for a finished and successful executed task, returns the SOS request needed to access the according fusion result (spatial coverage) stored in the Fusion SOS |
| Cancel | Cancels the execution of a running task as soon as possible |

**Table 3.4. IITB SPS operation support matrix**

### 3.4.3 Module interactions

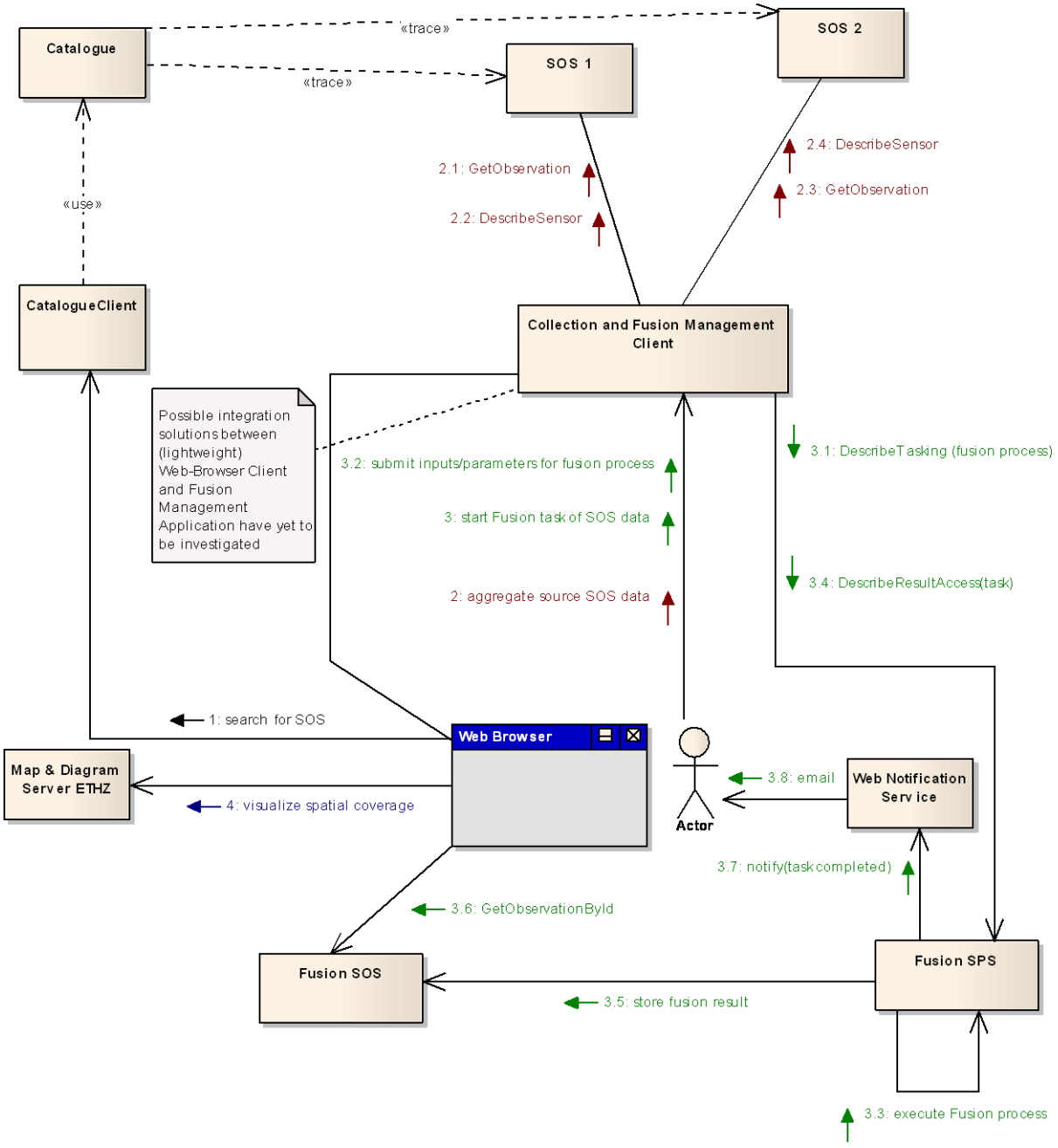



*Figure 3.14. General information workflow of IITB Fusion approach*

The Fusion SOS complies with the OGC SWE SOS 1.0 specification and serves as a data storage and retrieval service for spatial coverages that are the result of a "fusion process". More precisely, the spatial coverages are stored in the result block of an O&M Observations in the Fusion SOS. Furthermore, each Observation references a *FeatureOfInterest* of the type "SamplingSurface" and a *Procedure* of the type "SensorML". Each referenced SensorML document describes the concrete parameter configurations of the fusion process that generated this Observation. Each referenced SamplingSurface describes the area (a *gml:RectifiedGrid* with a surrounding shape) on which the fusion process was applied. This ensures a reproducible trace of the processing steps (see also XML examples in section 6.2).

The Fusion SPS complies with the OGC SPS v1.0 specification and serves as an interface to configure and control the execution of the fusion processes mentioned above. It allows the retrieval of a description of the configurable (taskable) parameters and inputs of the process with the *DescribeTasking* operation. The Fusion SPS itself uses assets to describe a given "taskable" process. In the IITB architecture of the Fusion design, an asset is a SensorML document that describes a fusion process (inputs, parameters, algorithm, outputs etc.). The two main inputs are:

- URL to the observations obtained from other SOS servers (usually aggregated as single members in one ObservationCollection XML-document)
- URL to an SamplingSurface describing the point collection (grid) on which the Fusion algorithm shall be applied

The parameters are used to further configure the fusion process (e.g. covariance model, variance and range of covariance model, etc.), see also XML examples in section 6.2.7.

After a client has retrieved these "taskable" parameters and inputs, a user can supply concrete values for them (e.g. over a GUI, Web Browser forms etc.). The client can then (optionally) use the *GetFeasability* operation to test if the fusion process (the "task") can be executed with these concrete values (e.g. are these valid values for this task? can the task be accomplished in a realistic timespan? etc.)

If the task is "feasible", the client can issue a *submit* request to the SPS which executes the fusion process (i.e. the aggregation and interpolation of the input SOS data as separate tasks or as integrated tasks). After successful completion, the resulting spatial coverage (along with the SamplingSurface and the SensorML describing the process) is then stored in the Fusion SOS (as explained at the beginning of this section). Also, the client is notified (for example via a notification service like WNS or WS-Notification) that the task has completed. The client can then invoke the *DescribeResultAccess* operation of the SPS, which will return the exact *GetObservation* request needed for retrieval of the produced coverage from the Fusion SOS (as an O&M Observation).

At any time, a client can request the current status of a fusion task with the *GetStatus* operation. A *Cancel* request can be issued to ask the Fusion SPS to abort a currently running process as soon as possible.

The following figure shows the typical message flow between the Fusion SOS and SPS modules:

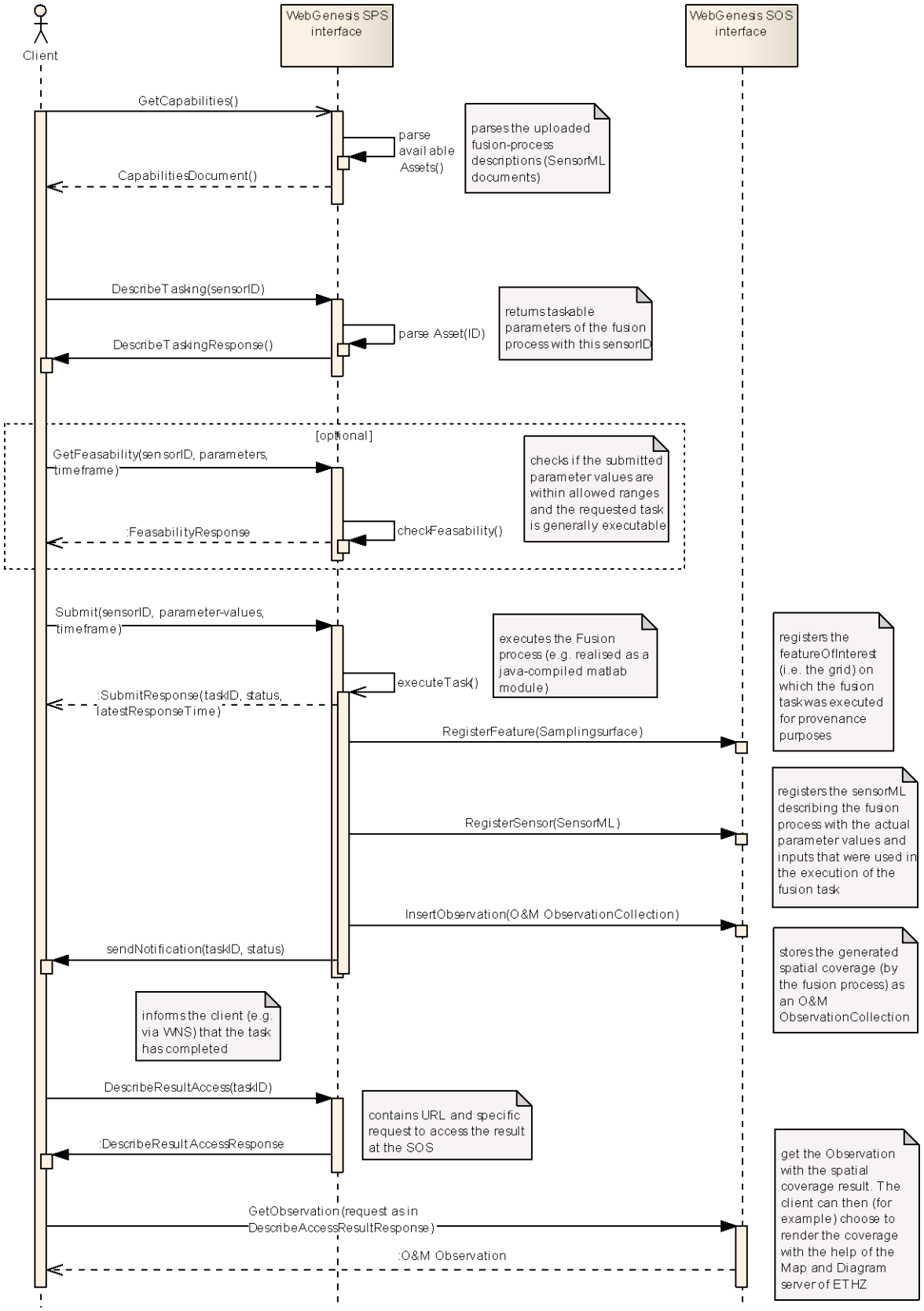
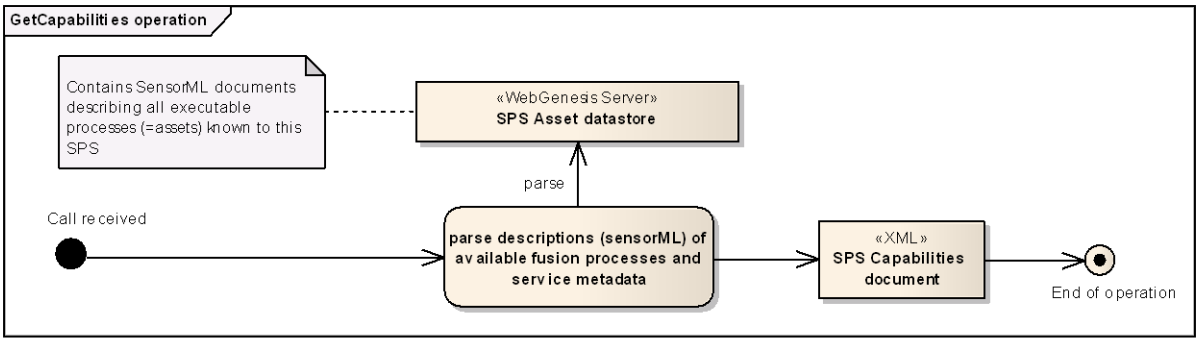
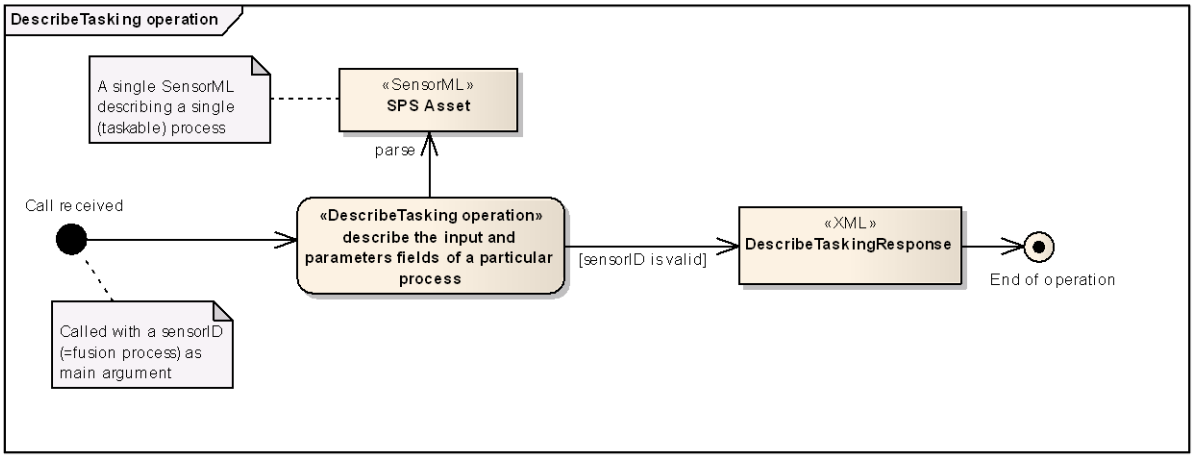
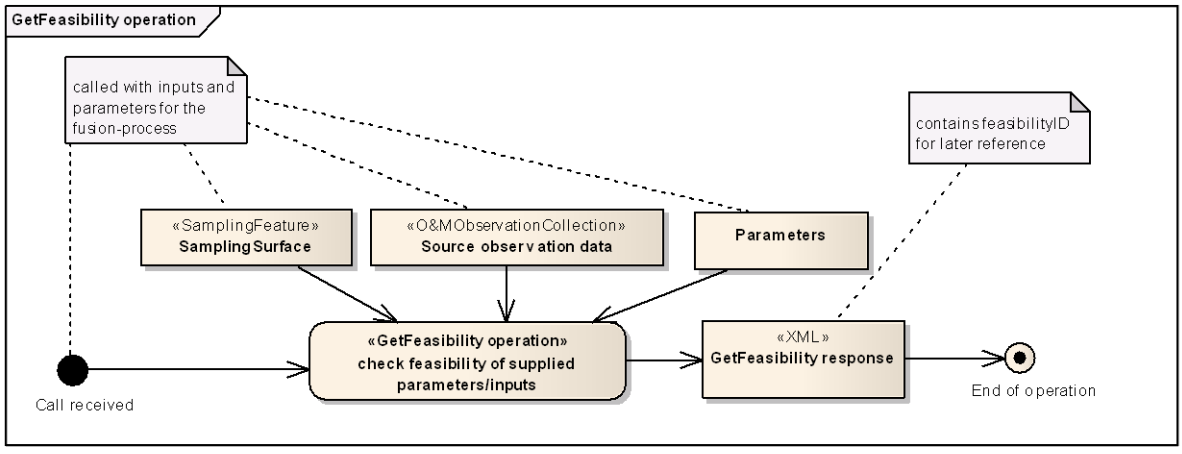


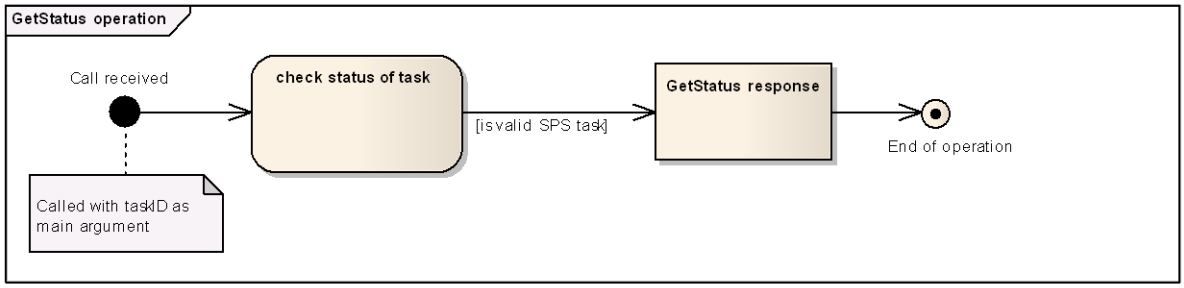

*Figure 3.15. Message exchange pattern Fusion SOS/SPS*

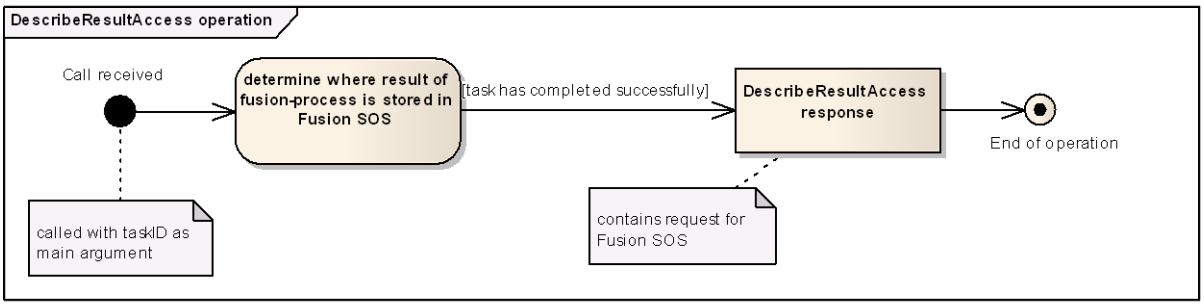
The following activity diagrams provide a more detailed overview about the Fusion SPS operation interactions as well as their inputs and outputs:

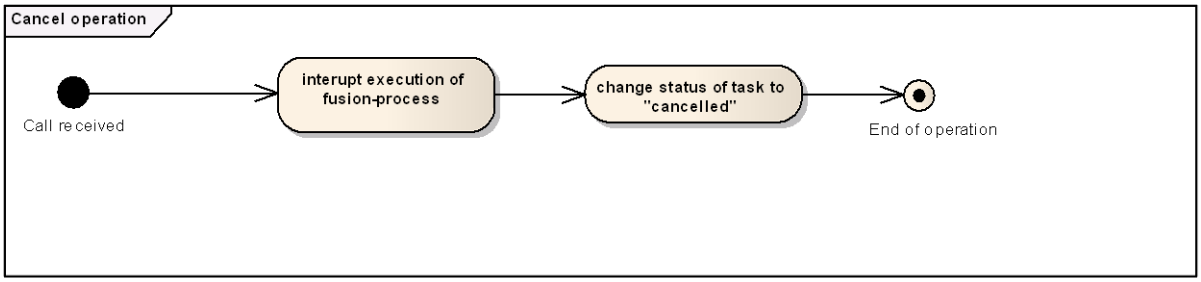



DescribeResultAccess operation
Call received
determine where result of fusion-process is stored in Fusion SOS
[task has completed successfully]
DescribeResultAccess response
End of operation
called with taskID as main argument
contains request for Fusion SOS


Cancel operation
Call received
interupt execution of fusion-process
change status of task to "cancelled"
End of operation

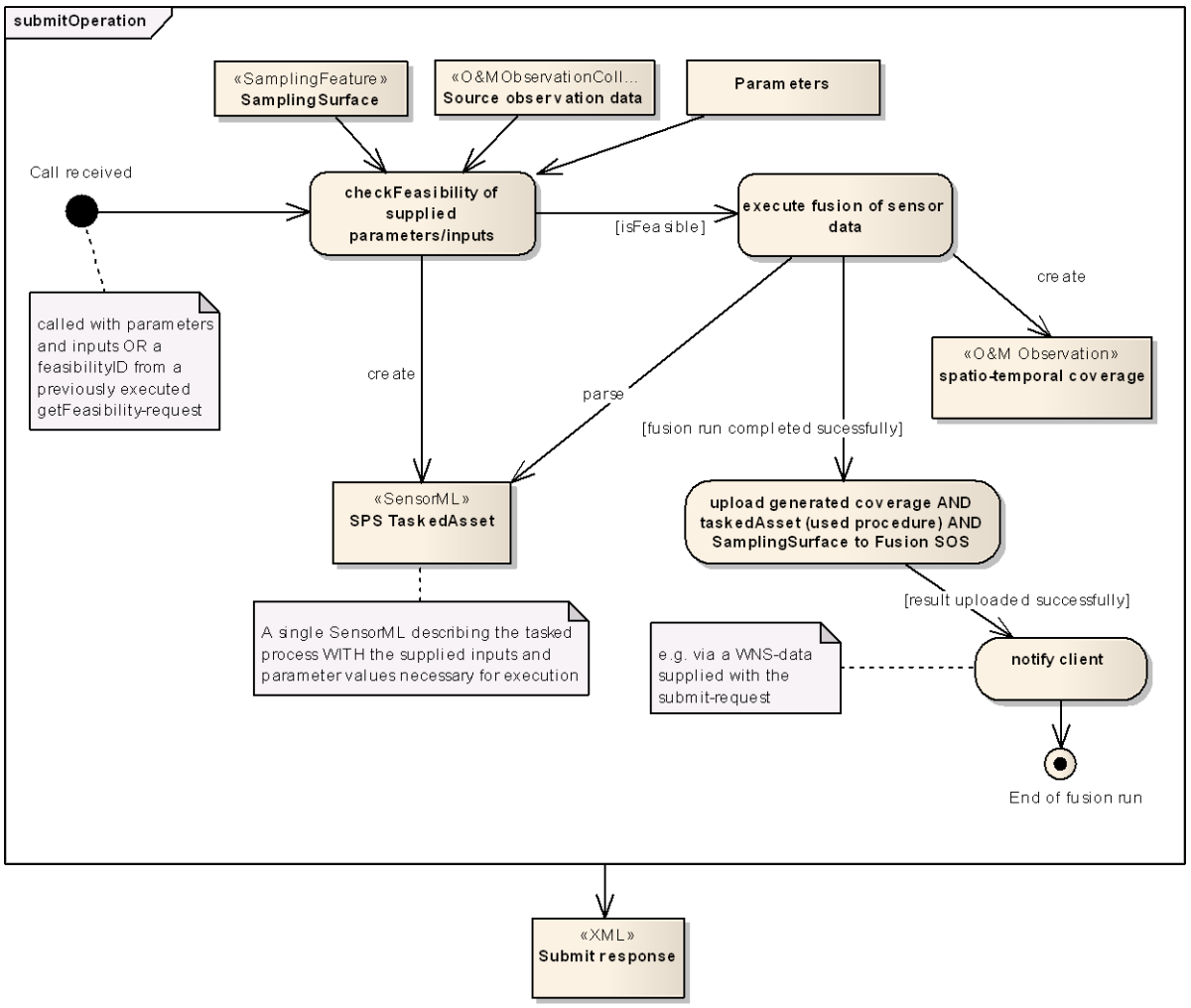



*Figure 3.16. Activity diagrams showing basic operations of Fusion SPS*

## 3.5. Algorithm development

### 3.5.1 Spatial algorithm: Kriging [IT-INNOV]

We have developed a generic ordinary kriging algorithm that is domain agnostic, not restricted to a particular phenomenon or set of dataset characteristics. This algorithm has been applied to wind and ground displacement sensor measurements. The algorithm accepts a set of point sensor values and returns a spatial grid of interpolated points based on these point values.

*Elevation correction*

In our wind dataset the sensors are located at different locations with different elevations above the ground. To improve accuracy we perform wind speed profiling from the observation's elevation to a desired reference elevation for all observations. This is not required for our ground displacement data case study.

*Periodic variable support*

Wind direction is reported as a meteorological angle, which is periodic. Most statistical estimation techniques are not directly fit for estimating periodic variables. For wind direction spatial interpolation we tackle the problem of estimating a periodic random variable with a simple but effective solution based on vector rotation and Cartesian transformation with stochastic simulation. We observed in our test datasets that about 80% of the wind direction angles spanned less then 180 degrees. In cases like these, before creating a variogram we will only need to rotate the wind directions in a way that the period onset doesn't intersect the wind directions span. This will eradicate the periodicity problem as the numerical distance between the wind direction angles will correspond to the distance between the wind directions and will enable variogram building and kriging.

*Automated variogram selection*

Our ordinary kriging procedure is combined with automated variogram model selection (AVMS). Background information describing the phenomenon characteristics can be reflected by the variogram model used for kriging. For the ordinary kriging with AVMS, along with the sensor data, metadata is supplied that imposes constraints to the variogram model to be selected in a way that reflects the phenomenology of the interpolated phenomenon.

The most critical part of creating an experimental variogram is the selection of lags. Lags need to be selected so they contain an optimal number of points so that physical phenomenon characteristics are not smoothed out and lost. Generally the initial slope of the variogram needs to be well estimated so the first few lags shall contain smaller number of points. If no hole-effect is expected the lags may contain a large number of points, but if hole-effect is expected the lags shall contain lower number of points so the effect is not smoothed out. The relative number of points in a lag is specified in the metadata supplied to the interpolation procedure. This relative number can be set by a phenomenon expert or pulled from an expert system listing known phenomena.

*Ordinary kriging algorithm*

For theoretical-variogram model selection we currently have eight models implemented: spherical, exponential, Gaussian, linear, power, generalised Bessel, sine hole-effect and cosine hole-effect. The model shape is governed by a subset of the following parameters: nugget, range, power, hole and sill. We use least-squares fitting method to select a model that best fits the experimental variogram. We can introduce background information about the phenomenon by constraining the fitted model types and the parameter values and then, in effect, a variogram model reflecting the characteristics of the phenomenon of interest will be selected.

Algorithm parameters include nugget, range, sil and hole. Nugget reflects the magnitude of rapid fluctuations (assuming that the measuring error is negligible in comparison to these fluctuations). Range reflects the scale of the phenomenon (for wind possibly reflects the surface topology too) in comparison to the size of the observed area. Sill depends on the phenomenon scale in relation to the size of the observed area. The hole effect exists or not depending on the essence of the phenomenon and the characteristics of the environment (for wind hole-effect may be caused by the surface topology).

After selecting the theoretical-variogram model, model parameters optimisation is performed in order to improve the internal consistency of the model. We use two statistics, termed Q1 and Q2, that need to be as close to their expected values as possible in order for the model to be consistent with the ordinary kriging inductive bias. We use quadratic- sequential programming to tune the model parameters, subject to the parameter constraints discussed above, with a loss function proportional to the squared differences between Q1 and Q2 and their respective expectations. Additionally, we try to keep the nugget of the model minimal but as a secondary objective (i.e. it is included in the loss function with a smaller weight). Unrealistically high nugget will over-smooth the estimates. Kriging over-smoothing will cause a very high standard deviation in an estimate.

After the variogram model optimisation stage standard ordinary kriging is performed using the optimised variogram model and estimate's mean and standard deviation are computed.

*Multi-region variogram support*

The ordinary kriging algorithm has been enhanced to take into account multi-region descriptions of areas of particular spatial cohesion. Such areas might be buildings, for ground displacement measurement, or ground of a similar height profile for wind measurement. A separate variogram is computed per region, and inter-region borders taken into account when performing the kriging of each interpolated point. The degree of contribution from sensor measurements outside each interpolated point's principle region can be configured based on the type of property being estimated (e.g. wind speed, ground displacement, temperature).

### 3.5.2 Spatial-social correlation: SP6 [IT-INNOV]

A spatial-correlation service has been created to fuse information from an economic impact database about buildings, cracking and the potential economic impact of this cracking with ground displacement sensor data from the SOLDATA SOS in the Barcelona region. The economic data used for this test is provided via an FTP site. The output of the test is a list of buildings, spatially correlated displacement values for each building of the nearest sensor, economic information such as tax and rental income and a 'likely cracking' warning flag based on the ground displacement threshold limit.

### 3.5.3 Spatial-temporal correlation: SP5 [IT-INNOV]

We have asked Triskel, on a single day when bouy is being cleaned, to measure salinity and turbidity at various points in the Falmouth towing the sensor in the water. Two runs will be made over the day and the results used to analyse how we can pre-plan mobile sensor routes to incrementally improve the overall spatial interpolation results in a spatial region. This work by IT-INNOV is an extra work on the Triskel data to compliment the principle work by BMT in SP5 on this data. Our objective is to examine aspects of sensor mobility in the context of data fusion in a lightweight study.

### 3.5.4 Domain model: PECK [IT-INNOV]

The PECK model [PECK1, PECK2, PECK3] describes the soil settlement during tunnel excavation and the final settlement expected at every point near the tunnel route after the end of

the work. The engineering bureau makes studies about the influence of the tunnel excavation on the soil and structures along the tunnel route and produces theoretical reference curves. The theoretical reference curves show the maximum settlement admissible during the works; they are defined by finite elements model and experience. The real measurements will be different from those theoretical reference curves and must not show higher displacements than predicted by the theoretical reference curves.

The PECK model extrapolates future settlement measurements for a given number of hours and compares the results with the theoretical reference curves. The output of the PECK model is a prediction of settlement and estimate of how close this prediction is to each alarm threshold (based on predefined theoretical reference curves). If several measurements are given for one tunnel position the median of the settlement values is used.

### 3.5.5 Spatial-temporal algorithm: Bayesian maximum entropy [IITB]

The fusion procedures developed to date are variants of the Bayesian Maximum Entropy method that is able to consider soft sensor data (e.g. the sensor value lies in an interval) and additional phenomenological knowledge in the form of models. The results are statistics encompassing the uncertainty of the spatial/temporal interpolation given the uncertainty of the available information.

The overall BME fusion method is structured [CHRISTAKOS1, CHRISTAKOS2] in 3 stages:

1. *prior stage:* Consideration of general physical and scientific knowledge **G** about the spatio-temporal properties of the phenomenon of interest. This knowledge may, for example, be expressed in the form of spatio-temporal differential equations derived from physical laws or as covariance models. It is what is known before experience with the specific situation is applied. The prior probability distribution of the so-called random field of the phenomenon is determined using the maximum entropy (ME) principle, i.e. it is the most uninformative (unbiased) probability distribution given only **G**.

2. *meta-prior stage:* Consideration of case-specific hard and soft data of the phenomenon of interest. This information is denoted by **S** (for specific knowledge) and is based on observations and measurements. Hard data refers to values believed to be accurate. Soft data is accompanied by uncertainty information such as a probability distribution for the value range.

3. *posterior stage:* Processing (fusion) of the available knowledge **G** and **S** of the prior and meta-prior stages respectively to make a probabilistic map of the phenomenon for a given set of spatio-temporal points (typically a grid). The map is a statement of the general knowledge **G** relative to the case-specific knowledge **S** and is derived using Bayesian conditional probabilities.

If the general knowledge **G** comprises the mean and covariance and if **S** includes only hard data, then the BME estimate coincides with the simple kriging estimate [CHRISTAKOS 1, proposition 12.2]. Similarly, if **G** is limited to the variogram and if **S** includes only hard data, then the BME estimate coincides with the ordinary kriging estimate [CHRISTAKOS 1, proposition 12.3].

In SANY, the knowledge **S** is represented as an observation collection described with the O&M model and including uncertainty information in UncertML. The map resulting from the posterior stage is represented as a coverage with associated uncertainty information. The Fraunhofer Fusion SOS is equipped with several fusion procedures that can be configured and tasked by an integrated SPS. The procedures execute the posterior stage for selected **G** and **S.**

### 3.5.6 Temporal algorithm: Regression [BMT]

Multiple linear regression models the dependent variable as linear combination of explanatory variables. The unknown weights and their statistical significance are estimated by least squares which minimize the squared error between the observed values and the predicted value. The probability of the predicted value above the guide line threshold, e.g., probability of exceedance, is determined through a student t-distribution.

The distribution of the dependent variable is examined before multiple linear regression. In SP5 application, logarithm of E-coli value is used to correct the skewness in the data. Seasonal cycle is removed from the dependent variable. The explanatory variables are selected through stepwise regression process which controls the inclusion of variables through F-tests.

### 3.5.7 Temporal algorithm: Artificial Neural Network [BMT]

Neural networks are mathematical structures analogous to biological neural networks. The radial basis network consists of three separate layers. The input layer is the environmental and hydrological variables. The second layer is a hidden layer of high dimensional nonlinear processing units. The output layer is the response of the network. The network topology is determined by the number of hidden units. Parameters of the neural network are chosen by minimizing the sum of squared error over the given training set.

### 3.5.8 Temporal algorithm: Probability Index [BMT]

Probability index represents the joint conditional probability on independent variables. The conditional probability on each input data are learned through the training data. Independent variables are then selected through searching for the maximum true predictions of events in training data. Prediction of future events are based on the selected variables and conditional probabilities associated with them. A threshold index is provided as a guideline. The output of probability index model is an index that indicates the likelihood that an event will occur.

### 3.5.9 Temporal algorithm: Auto-regression [BMT]

The time series data in SP4 is stationary over short-term period, e.g. one to two month. Time series analysis shows an autoregressive model is capable of modelling this data. An autoregressive model predicts future values of the time series in terms of its own past. The model complexity is chosen by Akaike information criterion.

The historical observation is used to estimate the model parameters. Since the past observations contain missing values, the autoregressive model is formulated in terms of a state-space model. The time series model outputs the predicted values and the associated confidence intervals.

### 3.5.10 State space model algorithm: Kalman filter [BMT]

State-space modelling is a powerful tool capable of handling a wide range of time series and dynamic models, The essential characteristics of data including trend, cycle, seasonal effect, and other latent variables can be expressed explicitly in the model and hence is easy to interpret. In SP4 the autoregressive process and time-varying principal factor are formulated into a state-space form, and in SP6, stochastic trend and daily cycles are included in the state model.

The air quality data in SP4 depends on a small number of latent factors. Factor analysis is used to extract the daily principal factors. The resulting factors are further rotated using a promax procedure to obtain a simple structure.

Once the model is put in a state space form, a Kalman filter is used to predict and update the state estimation. Model parameters are estimated by expectation-maximization (EM) algorithm. The non-stationary elements are initiated with exact Kalman filter. The problem of missing observation is handled by replacing corresponding elements and design matrix to zero.

# 4. Conclusions

This report defines 9 best practice statements that represent our considered view on how best to develop fusion and modelling services within an OGC service infrastructure.

We describe architectural designs, including sequence diagrams showing how these modules interact, for following case-study software modules:

- Fusion WPS server [IT-INNOV, BMT]
- Fusion SPS [IITB, BMT]
- Fusion SOS server [IITB]
- Fusion algorithms [IT-INNOV, BMT, IITB]
- Fusion assessment toolkit [IT-INNOV]

Of the 31 requirements in D3.3.1.3 all 28 are satisfied via SP3 service & algorithm implementations and 3 are satisfied via written reports. The traceability sections in this document explain in more detail exactly how and how much each requirement is satisfied.

Finally the appendix A contains a full set of concrete OGC request and response examples that is intended to help developers who are considering either using a SANY processing service, or writing one of their own.

# 5. References

## Deliverables

**D3.3.1.3** SANY deliverable D3.3.1.3 'Fusion and Modelling Requirements', 2009

**D2.3.3** SANY Deliverable D2.3.3 'Specification of the Sensor Service Architecture V2', 2009

**D2.4.3** SANY Deliverable D2.4.3 'Sensor Service Specification V2', 2009

## Other

**PECK 1** Foundation Engineering, Ralph B. Peck, Walter E. Hanson, and Thomas H. Thornburn, 1974

**PECK 2** Soil Mechanics in Engineering Practice, Karl Terzaghi, Ralph B. Peck, and Gholamreza Mesri, 1996

**PECK 3** Tunnels et ouvrages souterrains, recommandations de l'AFTES, October 1999

**INTAMAP** INTAMAP (2007), uncertML. http://www.intamap.org/uncertml/uncertml.php

**CHRISTAKOS1** Modern spatiotemporal geostatistics, Christakos, G., Oxford University Press, New York, 2nd edition (2001)

**CHRISTAKOS2** Temporal GIS: advanced functions for field-based applications, Christakos, G., Bogaert, P., L. and Serre, M.L., Springer, 2002, ISBN 3540414762, 9783540414766

**ORCHESTRA1** Implementation Specification of the Catalogue Service (Version 0.8), ORCHESTRA Consortium, Editor: IITB, available from: http://www.eu-orchestra.org/publications.shtml#OAImplspecs

**ORCHESTRA2** Service Specification of the Catalogue Service (Version 1.1), ORCHESTRA Consortium, Editor: IITB, available from: http://www.eu-orchestra.org/publications.shtml#OAspecs

**WEBGENESIS** IITB's *WebGenesis* information management system http://www.iitb.fraunhofer.de/servlet/is/2223/WebGenesis-Produktblatt-English.pdf?command=downloadContent&filename= WebGenesis-Produktblatt-English.pdf

## OGC documents

**O&M Part 1, 2007** Observation & Measurements - Part 1: Observation Schema. OGC Document 07-022r1, approved as OpenGIS® implementation standard,

http://www.opengeospatial.org/standards/om, http://portal.opengeospatial.org/files/?artifact_id=22466&version=2, 2007-12-08

**O&M Part 2, 2007** Observation & Measurements - Part 2: Sampling Features. OGC Document 07-002r3, approved as OpenGIS® implementation standard, http://www.opengeospatial.org/standards/om, http://portal.opengeospatial.org/files/?artifact_id=22467&version=2, 2007-12-08

**SensorML, 2007** Sensor Model Language Version 1.0.0. OGC Document 07-000, http://portal.opengeospatial.org/files/?artifact_id=21273, 2007

**SOS, 2007** Sensor Observation Service. OGC Document 06-009r5, approved as OpenGIS® implementation standard, http://portal.opengeospatial.org/files/?artifact_id=20994, 2007-02-28

**SPS, 2007** Sensor Planning Service Implementation Specification. OGC Document 07-014r3, approved as OpenGIS® implementation standard, http://portal.opengeospatial.org/files/?artifact_id=23180, 2007-08-02

**Common, 2007** OGC Web Services Common Specification. OGC Document 06-121r3, approved as OpenGIS® Implementation Specification, http://www.opengeospatial.org/standards/common, 2007-02-09

**RFC 5165** Uniform Resource Name (URN) Namespace for the Open Geospatial Consortium (OGC)" (RFC 5165)

**WPS, 2007** Web Processing Service. OGC Document 05-007r7, approved as OpenGIS® implementation standard, http://portal.opengeospatial.org/files/?artifact_id=24151, 2007-06-08

# 6. Appendix A: I/O specification

## 6.1. IT-INNOV I/O specification

### 6.1.1 IT-INNOV WPS GetCapabilities

**GetCapabilities [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:GetCapabilities xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" language="en-CA"
service="WPS">
 <wps:AcceptVersions>
  <ows:Version>1.0.0</ows:Version>
 </wps:AcceptVersions>
</wps:GetCapabilities>
```

**GetCapabilities [response]**

```
<?xml version="1.0" encoding="utf-8"?>
<Capabilities version="1.0.0" xml:lang="en-CA"
xmlns="http://www.opengis.net/wps/1.0.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ows="http://www.opengeospatial.net/ows"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

 <ows:ServiceIdentification>
  <ows:Title/>
  <ows:Abstract/>
  <ows:Keywords>
   <ows:Keyword>WPS</ows:Keyword>
   <ows:Keyword>Fusion</ows:Keyword>
  </ows:Keywords>
  <ows:ServiceType>WPS</ows:ServiceType>
  <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
  <ows:Fees>NONE</ows:Fees>
  <ows:AccessConstraints>

   … warranty text on terms of use of the service …

  </ows:AccessConstraints>
 </ows:ServiceIdentification>

 <ows:ServiceProvider>
  <ows:ProviderName/>
  <ows:ProviderSite xlink:href="IT-INNOV website URL"/>
  <ows:ServiceContact>
   <ows:IndividualName/>
   <ows:PositionName/>
   <ows:ContactInfo>
    <ows:Phone>
     <ows:Voice/>
     <ows:Facsimile/>
    </ows:Phone>
    <ows:Address>
     <ows:DeliveryPoint/>
     <ows:City/>
     <ows:AdministrativeArea/>
     <ows:PostalCode/>
     <ows:Country/>
     <ows:ElectronicMailAddress/>
    </ows:Address>
   </ows:ContactInfo>
  </ows:ServiceContact>
 </ows:ServiceProvider>

 <ows:OperationsMetadata>
  <ows:Operation name="GetCapabilities">
   <ows:DCP>
    <ows:HTTP>
     <ows:Get xlink:href="http://152.78.239.173/FusionService.asmx?"/>
    </ows:HTTP>
   </ows:DCP>
  </ows:Operation>

  <ows:Operation name="DescribeProcess">
```

```
    <ows:DCP>
     <ows:HTTP>
      <ows:Post xlink:href="http://152.78.239.173/FusionService.asmx"/>
     </ows:HTTP>
    </ows:DCP>
   </ows:Operation>

   <ows:Operation name="Execute">
    <ows:DCP>
     <ows:HTTP>
      <ows:Post xlink:href="http://152.78.239.173/FusionService.asmx"/>
     </ows:HTTP>
    </ows:DCP>
   </ows:Operation>
  </ows:OperationsMetadata>

  <ProcessOfferings>
   <Process processVersion="1.0">
    <ows:Identifier>process_name</ows:Identifier>
    <ows:Title/>
    <ows:Abstract/>
    <ows:Metadata xlink:title="SensorML">discovery_sensorML</ows:Metadata>
   </Process>
  </ProcessOfferings>

  <Languages>
   <Default>
    <ows:Language>en-CA</ows:Language>
   </Default>
   <Supported>
    <ows:Language>en-CA</ows:Language>
   </Supported>
  </Languages>
</Capabilities>

process_name = KrigingProcess | PECK (TBD)
discovery_sensorML = FTL URL to a SensorML file associated with this process
that describes the process capabilities. This file can be used for process
discovery purposes.
```

The metadata child element under each process element will contain a SensorML attribute that defines a FTP URL to a SensorML description of that process. This allows a client (e.g. PCS) to access a richer machine readable description of each fusion process, describing its capabilities in a way that would facilitate service discovery and selection.

### 6.1.2 IT-INNOV WPS DescribeProcess

**DescribeProcess [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeProcess xmlns="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" service="WPS"
version="1.0.0" language="en-CA">
 <ows:Identifier>process_name</ows:Identifier>
</DescribeProcess>

process_name = KrigingProcess | PECK (TBD)
```

The process names should be discovered from a previous GetCapabilities request to ensure they are correct.

**DescribeProcess [response]**

```
<?xml version="1.0" encoding="utf-8"?>
<ProcessDescriptions service="WPS" version="1.0.0" xml:lang="en-CA"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml" xmlns="">

 <ProcessDescription wps:processVersion="1.0" storeSupported="true"
statusSupported="true">

  <ows:Identifier>process_name</ows:Identifier>
  <ows:Title/>
  <ows:Abstract/>
  <ows:Metadata xlink:title="SensorML">discovery_sensorML</ows:Metadata>

  <DataInputs>
   <Input minOccurs="1" maxOccurs="1000">
    <ows:Identifier>SOSDataset</ows:Identifier>
    <ows:Title>Unique identifier to a SOS</ows:Title>
    <ows:Abstract/>
    <LiteralData>
     <ows:DataType>String</ows:DataType>
     <ows:AllowedValues>
      <ows:Value>IITB-NOAA</ows:Value>
      <ows:Value>SOLDATA-ETELE-TERI-ALL</ows:Value>
      ... one for each dataset supported ...
     </ows:AllowedValues>
    </LiteralData>
   </Input>

   <Input minOccurs="0" maxOccurs="1">
    <ows:Identifier>SaveResultInSos</ows:Identifier>
    <ows:Title>Save the results in the SOS</ows:Title>
    <ows:Abstract/>
    <LiteralData>
     <ows:DataType>Boolean</ows:DataType>
     <ows:AnyValue/>
    </LiteralData>
   </Input>

   <Input minOccurs="1" maxOccurs="1000">
    <ows:Identifier>SamplingTime</ows:Identifier>
    <ows:Title>Sampling time</ows:Title>
    <ows:Abstract/>
    <LiteralData>
     <ows:DataType>Time</ows:DataType>
     <ows:AnyValue/>
    </LiteralData>
   </Input>

   <Input minOccurs="0" maxOccurs="1">
    <ows:Identifier>SamplingWindow</ows:Identifier>
    <ows:Title>Sampling period (s)</ows:Title>
    <ows:Abstract/>
    <LiteralData>
     <ows:DataType>Duration</ows:DataType>
```

```
      <ows:AnyValue/>
     </LiteralData>
    </Input>

    <Input minOccurs="0" maxOccurs="1">
     <ows:Identifier>CalcDelta</ows:Identifier>
     <ows:Title>Calculate Delta</ows:Title>
     <ows:Abstract/>
     <LiteralData>
      <ows:DataType>Boolean</ows:DataType>
      <ows:AnyValue/>
     </LiteralData>
    </Input>

    <Input minOccurs="1" maxOccurs="1000">
     <ows:Identifier>ObservedProperty</ows:Identifier>
     <ows:Title>SOS observed property</ows:Title>
     <ows:Abstract/>
     <LiteralData>
      <ows:DataType>String</ows:DataType>
      <ows:AllowedValues>
       <ows:Value>obs_property</ows:Value>
       … full list of obs properties allowed …
      </ows:AllowedValues>
     </LiteralData>
    </Input>

    <Input minOccurs="0" maxOccurs="1">
     <ows:Identifier>SpatialRegion</ows:Identifier>
     <ows:Title>Spatial region</ows:Title>
     <ows:Abstract/>
     <BoundingBoxData>
      <Default>
       <CRS/>
      </Default>
      <Supported>
       <CRS/>
      </Supported>
     </BoundingBoxData>
    </Input>

    <Input minOccurs="1" maxOccurs="1">
     <ows:Identifier>GridResolutionX</ows:Identifier>
     <ows:Title>Grid points on X axis</ows:Title>
     <ows:Abstract/>
     <LiteralData>
      <ows:DataType>Integer</ows:DataType>
      <ows:AnyValue/>
     </LiteralData>
    </Input>

    <Input minOccurs="1" maxOccurs="1">
     <ows:Identifier>GridResolutionY</ows:Identifier>
     <ows:Title>Grid points on Y axis</ows:Title>
     <ows:Abstract/>
     <LiteralData>
      <ows:DataType>Integer</ows:DataType>
      <ows:AnyValue/>
     </LiteralData>
    </Input>
```

```
   <Input minOccurs="0" maxOccurs="1">
    <ows:Identifier>ExtraPoints</ows:Identifier>
    <ows:Title>Extra spatial points</ows:Title>
    <ows:Abstract/>
    <ComplexData>
     <Default>
      <Format>
       <MimeType>text/xml</MimeType>

<Schema>defaultSchema="http://schemas.opengis.net/gml/3.1.1/base/geometryBasi
c0d1d.xsd"</Schema>
      </Format>
     </Default>
     <Supported>
      <Format>
       <MimeType>text/xml</MimeType>

<Schema>defaultSchema="http://schemas.opengis.net/gml/3.1.1/base/geometryBasi
c0d1d.xsd"</Schema>
      </Format>
     </Supported>
    </ComplexData>
   </Input>

   <Input minOccurs="0" maxOccurs="1">
    <ows:Identifier>FTPFileList</ows:Identifier>
    <ows:Title>FTP Input file list</ows:Title>
    <ows:Abstract/>
    <LiteralData>
     <ows:DataType>URL</ows:DataType>
     <ows:AnyValue/>
    </LiteralData>
   </Input>

  </DataInputs>

  <ProcessOutputs>
   <Output>
    <ows:Identifier>CSV_Result_Data</ows:Identifier>
    <ows:Title>CSV result data</ows:Title>
    <ows:Abstract/>
    <LiteralOutput>
     <ows:DataType>URL</ows:DataType>
    </LiteralOutput>
   </Output>
   <Output>
    <ows:Identifier>CSV_Result_Metadata</ows:Identifier>
    <ows:Title>CSV result metadata</ows:Title>
    <ows:Abstract/>
    <LiteralOutput>
     <ows:DataType>URL</ows:DataType>
    </LiteralOutput>
   </Output>
   <Output>
    <ows:Identifier>OM_Result_List</ows:Identifier>
    <ows:Title>OM result list</ows:Title>
    <ows:Abstract/>
    <LiteralOutput>
     <ows:DataType>URL</ows:DataType>
```

```
    </LiteralOutput>
   </Output>
   <Output>
    <ows:Identifier>KML_Result_Data</ows:Identifier>
    <ows:Title>KML result data</ows:Title>
    <ows:Abstract/>
    <LiteralOutput>
     <ows:DataType>URL</ows:DataType>
    </LiteralOutput>
   </Output>

   Note: There will be one set of outputs (above) per result set generated.
The number of result sets generated will depend on the number of observed
properties requested, and the nature of the fusion process chosen. Example is
for the KrigingProcess.

  </ProcessOutputs>

 </ProcessDescription>

</ProcessDescriptions>

process_name = KrigingProcess | PECK (TBD)
discovery_sensorML = FTL URL to a SensorML file associated with this process
that describes the process capabilities. This file can be used for process
discovery purposes. (see section 3.3.7)
obs_property = observed property URN's supported (SOS defined)
```

The metadata child element under each process element will contain a SensorML attribute that defines a FTP URL to a SensorML description of that process. This allows a client (e.g. PCS) to access a richer machine readable description of each fusion process, describing its capabilities in a way that would facilitate service discovery and selection.

The input parameters are already described in the Execute request section of this document.

The ProcessOutputs define a number of reference parameters to the possible outputs from the fusion process; there can be any number of outputs depending on how many observed properties are requested by the client and the type of the fusion process. They are all complex data types which will appear as references in the ExecuteResponse (see section 6.1.3).

The principle output from any fusion process (e.g. Kriging) is a set of 2 CSV files per result set. There will be a CSV file with the fused data values themselves (e.g. interpolated spatial point values for an observed property) and a metadata description that includes the statistical associated values. Examples of these files are in later sections of this document. A fusion process can create multiple result sets depending on the input parameters and nature of the fusion being performed.

Additional outputs are KML (Google Earth visualizations of the CSV data) and XML O&M result files. The O&M result files (SensorML, SamplingSurface, GetObservationResponse) are provided to allow service chaining, and provide the results in an OGC SWE format.

**Discovery information [SensorML]**

```
<?xml version="1.0" encoding="utf-8"?>
<sml:SensorML xmlns:sml="http://www.opengis.net/sensorML/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0">

 <sml:member>
  <sml:ProcessModel>

   <sml:identification>
    <sml:IdentifierList>
     <sml:identifier name="uniqueID">
      <sml:Term definition="urn:ogc:def:identifier:OGC:uniqueID">
       <sml:value>fusion_process_id</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="manufacturerName">
      <sml:Term
       definition="urn:ogc:def:identifier:OGC:1.0.1:manufacturerName">
       <sml:value>fusion_owner</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="longName">
      <sml:Term definition="urn:ogc:def:identifier:OGC:1.0.1:longname">
       <sml:value>fusion_process_long_desc</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="shortName">
      <sml:Term
       definition="urn:ogc:def:identifier:OGC:1.0.1:shortName">
       <sml:value>fusion_process_short_desc</sml:value>
      </sml:Term>
     </sml:identifier>
    </sml:IdentifierList>
   </sml:identification>

   <sml:inputs>
    <sml:InputList>
     <sml:input name="fusion_param_id">
      <swe:Text>
       <gml:description>fusion_param_desc</gml:description>
      </swe:Text>
     </sml:input>
     <sml:input name="fusion_param_id">
      <swe:Quantity>
       <gml:description>fusion_param_desc</gml:description>
       <swe:uom code="unit_urn"/>

     Note: unit element can be empty string if no unit is appropriate

      </swe:Quantity>
     </sml:input>
     <sml:input name="fusion_param_id">
      <swe:Time>
       <gml:description>fusion_param_desc</gml:description>
      </swe:Time>
```

```
    </sml:input>

Note: there will be many parameter entries

  </sml:InputList>
 </sml:inputs>

 <sml:outputs>
  <sml:OutputList>
   <sml:output name="OutputFile">
    <swe:Text>
     <gml:description>fusion_output_desc</gml:description>
    </swe:Text>
   </sml:output>

Note: there will be many output URL entries

  </sml:OutputList>
 </sml:outputs>

 <sml:parameters>
  <sml:ParameterList>
   <sml:parameter name="fusion_param_id">
    <swe:Text>
     <gml:description>fusion_param_desc</gml:description>
    </swe:Text>
   </sml:parameter>
   <sml:parameter name="fusion_param_id">
    <swe:Quantity>
     <gml:description>fusion_param_desc</gml:description>
     <swe:uom code="unit_urn"/>

   Note: unit element can be empty string if no unit is appropriate

    </swe:Quantity>
   </sml:parameter>
   <sml:parameter name="fusion_param_id">
    <swe:Time>
     <gml:description>fusion_param_desc</gml:description>
    </swe:Time>
   </sml:parameter>

Note: there will be many parameter entries

  </sml:ParameterList>
 </sml:parameters>

 <sml:method>
  <sml:ProcessMethod>
   <gml:description>fusion_script_desc</gml:description>
   <sml:rules>
    <sml:RulesDefinition>
     <gml:description>
      fusion_script_io_info
     </gml:description>
    </sml:RulesDefinition>
   </sml:rules>
   <sml:algorithm>
    <sml:AlgorithmDefinition>
     <gml:description>
```

```
        fusion_algorithm_desc
       </gml:description>
      </sml:AlgorithmDefinition>
     </sml:algorithm>
     <sml:implementation>
      <sml:ImplementationCode
       language="fusion_impl_lang"
       version="fusion_impl_version">
       <gml:description>
        Fusion_plugin_desc
       </gml:description>
      </sml:ImplementationCode>
     </sml:implementation>

  Note: There will be an implementation entry per fusion c# plug-in used

    </sml:ProcessMethod>
   </sml:method>

  </sml:ProcessModel>
 </sml:member>
</sml:SensorML>

fusion_process_id = unique name of the fusion process (e.g. KrigingProcess)
fusion_owner = WPS owner (e.g. IT Innovation Centre)
fusion_param_id = id of parameter passed by client or statically defined
(e.g. EventTime)
fusion_param_desc = description of parameter (e.g. Sampling time used to
request sensor data from SOS)
fusion_param_value_text = textual values (e.g. true)
fusion_param_value_numeric = numeric values (e.g. 54.2)
fusion_param_value_time = temporal values (e.g. 2005-04-05T12:00:00+01)
fusion_output_desc = description of output param
fusion_output_URL = URL
fusion_process_long_desc = long name of fusion process (e.g. Ordinary Kriging
with AVMS)
fusion_process_short_desc = short name of fusion process (e.g. Kriging)
fusion_input_id = id of input parameter passed by client (e.g. EventTime)
fusion_input_desc = description of input parameter (e.g. Sampling time used
to request sensor data from SOS)
fusion_name = internal name of the fusion process (e.g. Ordinary Kriging)
fusion_master_script_id = internal name of fusion script (e.g.
script_master_kriging)
fusion_script_desc = internal description of fusion script (e.g. Correlation,
Syntax checking, Normalization, Elevation correction, Ordinary Kriging, AVMS)
fusion_script_io_info = internal description of any input / output
restrictions (e.g. Number of sensors must be > 10 to achieve reasonable
results)
fusion_algorithm_desc = internal description of the algorithm components used
(e.g. OrdinaryKriging, ElevationCorrection, AVMS)
fusion_impl_lang = language of algorithm plugin (e.g. c#)
fusion_impl_version = version of plugin (e.g. 1.0)
Fusion_plugin_desc = name of fusion plugin (e.g. OrdinaryKriging)
```

There is a single SensorML file per WPS fusion process. This file describes the type of fusion that could be executed and the input parameters that would be needed. The purpose of this SensorML is to provide a more machine readable description of a process than is available in the DescribeProcess response.

The parameter elements provide a detailed record of the input parameters that must be provided by the client. The ProcessMethod element provides a detailed record of the processing steps that would be executed to handle an Execute request.

sml:parameters is a list of the algorithm config parameters.

sml:inputs is a list of WPS input parameters.

sml:outputs is a list of WPS output files created at the end of the processing act.

### 6.1.3 IT-INNOV WPS Execute

**Execute [request for input data located in an SOS]**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Execute service="WPS" version="1.0.0"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wps/1.0.0">

 <ows:Identifier>process_name</ows:Identifier>

 <DataInputs>
  <Input>
   <ows:Identifier>SOSDataset</ows:Identifier>
   <Data>
    <LiteralData dataType="String">dataset_name</LiteralData>
   </Data>
  </Input>

  Note: WPS will accept multiple SOSDataset entries

  <Input>
   <ows:Identifier>SaveResultToSos</ows:Identifier>
   <Data>
    <LiteralData dataType="Boolean">save_to_sos</LiteralData>
   </Data>
  </Input>

  <Input>
   <ows:Identifier>ObservedProperty</ows:Identifier>
   <Data>
    <LiteralData dataType="String">obs_property</LiteralData>
   </Data>
  </Input>

  Note: WPS will accept multiple ObservedProperty entries

  <Input>
   <ows:Identifier>SamplingTime</ows:Identifier>
   <Data>
    <LiteralData dataType="Time">sampling_time</LiteralData>
   </Data>
  </Input>

  Note: WPS will accept multiple sampling times (as required)

  <Input>
   <ows:Identifier>SamplingWindow</ows:Identifier>
   <Data>
    <LiteralData dataType="Duration">sampling_window</LiteralData>
   </Data>
  </Input>
  <Input>
   <ows:Identifier>CalcDelta</ows:Identifier>
```

```
    <Data>
     <LiteralData dataType="Boolean">calc_delta</LiteralData>
    </Data>
   </Input>
   <Input>
    <ows:Identifier>SpatialRegion</ows:Identifier>
    <Data>
     <BoundingBoxData>
      <ows:LowerCorner>deg_lat deg_long</ows:LowerCorner>
      <ows:UpperCorner>deg_lat deg_long</ows:UpperCorner>
     </BoundingBoxData>
    </Data>
   </Input>
   <Input>
    <ows:Identifier>GridResolutionX</ows:Identifier>
    <Data>
     <LiteralData dataType="Integer">resolution</LiteralData>
    </Data>
   </Input>
   <Input>
    <ows:Identifier>GridResolutionY</ows:Identifier>
    <Data>
     <LiteralData dataType="Integer">resolution</LiteralData>
    </Data>
   </Input>
   <Input>
    <ows:Identifier>ExtraPoints</ows:Identifier>
    <Data>
     <ComplexData>gml_pos_list</ComplexData>
    </Data>
   </Input>
  </DataInputs>

  <ResponseForm>
   <ResponseDocument status="true" storeExecuteResponse="true" lineage="true">

       Note: WPS supports lineage values of true and false
       Note: WPS only supports storeExecuteResponse values of true
       Note: WPS only supports status values of true

    <Output asReference=”true”>
     <ows:Identifier>CSV_Result_Data</ows:Identifier>
    </Output>
    <Output asReference=”true”>
     <ows:Identifier>CSV_Result_Metadata</ows:Identifier>
    </Output>
    <Output asReference=”true”>
     <ows:Identifier>KML_Result_Data</ows:Identifier>
    </Output>
    <Output asReference=”true”>
     <ows:Identifier>OM_Result_Data</ows:Identifier>
    </Output>

    Note: There can be many instances of these Output’s

   </ResponseDocument>
  </ResponseForm>
</Execute>

process_name = KrigingProcess | PECK (TBD)
```

```
dataset_name = IITB-NOAA | SOLDATA-ETELE-TERI-ALL
save_to_sos = true if results are to be uploaded to fusion SOS
obs_property = property URN e.g. urn:ogc:def:phenomenon:OGC:1.0:temperature
sampling_time = e.g. 2005-04-05T12:00:00+01 there can be multiple if more
than one time is required, or a delta calculation is required
sampling_window = duration in seconds [integer]
calc_delta = true if a delta (difference) calculation is needed between the
first and second sampling time
deg_lat = degrees latitude [double]
deg_long = degrees latitude [double]
resolution = resolution in number of points (min 1) [integer]
gml_pos_list = GML posList entry for a set of (lat,long) points that will be
interpolated in addition to the grid points. An example of a posList with 3
points is <gml:posList count="3" srsDimension="2">18.810 54.597 18.47000
54.38083 18.81194 54.60361</gml:posList>
```

The normal Execute request does not contain an FTP input parameter and expects the WPS service itself to contact the SOS's and retrieve input data from them.

The 'SOSDataset' input parameter must be one of the allowed dataset profile names returned in the DescribeProcess allowedValues response. An SOSDataset profile must be specified for each required SOS that will be asked to provide input data. Each SOS will be asked to return the full set of observed properties, even if some properties are not present (in which case only the properties the SOS can return are returned).

The SaveResultInSos input parameters are Booleans that control how the fusion SOS is used. If the SaveResultsInSos is true then the results of the fusion run are uploaded into the fusion SOS in addition to being made available on the FTP for download.

The ObservedProperty parameter must contain a SOS property URN. Where multiple SOS's are used this property must be present in at least one of the SOS's. Each property specified will be used as input into the fusion process.

The SamplingTime input parameter is needed for spatial fusion to specify the event time that will be used in SOS requests (i.e. sampling time). If an SamplingWindow input parameter is present the SOS request will be for all sensor values between the time window from (SamplingTime) to (SamplingTime + SamplingWindow). If needed multiple sampling times can be provided depending on what is needed for the algorithm.

The CalcDelta flag instructs the first sampling time to be subtracted from the second, computing a delta value that is passed to the algorithm. This is useful for processing temporal displacements.

The SpatialRegion determines the area spatial fusion should work on. The GridResolutionX and GridResolutionY values determine the number of spatial interpolation points within the grid specified by the SpatialRegion. The ExtraPoints value can be used to add specific points to the spatial grid where interpolation should occur. The output might therefore not be a true Grid but just a collection of points.

**Execute [request where input data is located on an FTP site]**

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<Execute service="WPS" version="1.0.0"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns="http://www.opengis.net/wps/1.0.0">

 <ows:Identifier>process_name</ows:Identifier>

 <DataInputs>

  Note: same input parameters as SOS execute request apply

  <Input>
   <ows:Identifier>FTPFileList</ows:Identifier>
   <LiteralValue dataType="URL">filelist_ftp_url</LiteralValue>
  </Input>
 </DataInputs>

 <ResponseForm>
  <ResponseDocument status="true" storeExecuteResponse="true" lineage="true">

  Note: same output parameters as SOS execute request apply

  </ResponseDocument>
 </ResponseForm>
</Execute>

process_name = KrigingProcess | PECK (TBD)
filelist_ftp_url = FTP URL to a FTP file list configuration file
```

The FTPFileList input parameter can be used in addition to all the parameters of a normal Execute request. It will cause a FTP file configuration file to be downloaded from the URL provided and all SOS response files referenced by this FTP file configuration to be downloaded into the algorithm's work directory; downloaded files will be renamed to avoid possible naming conflicts.

Whenever input data is requested from a SOSDataset the downloaded FTP response files will first be checked to see if the SOS response has already been performed and is thus available cached. This allows the client (e.g. PCS) to access multiple SOS's directly and provide an efficient place where the responses can be found, potentially avoiding redundant multiple access to the same SOS by different processing services.

**FTP file list configuration file**

```
<?xml version="1.0" encoding="UTF-8"?>
<FTPList>
 <FTPEntry>
  <DatasetProfile>dataset_name</DatasetProfile>
  <FTPURL>ftp_url</FTPURL>
  <Type>file_format</Type>
  <Metadata>
   <tag>value</tag>
      … optional for use by DescribeSensor and GetFeatureOfInterestId …
  </Metadata>
 </FTPEntry>

 e.g. below example entry, there can be many FTPEntry elements

 <FTPEntry>
  <DatasetProfile>IITB-NOAA</DatasetProfile>
  <FTPURL>ftp://localhost/job/SP5_IITB_SOS/DescribeSensor/Meteorological
Station Elblag.xml</FTPURL>
  <Type>DescribeSensor</Type>
  <Metadata>
   <procedure>Meteorological Station Elblag</procedure>
  </Metadata>
 </FTPEntry>

</FTPList>

dataset_name = IITB-NOAA | SOLDATA-ETELE-TERI-ALL
ftp_url = FTP URL to response file
file_format = GetCapabilities | DescribeSensor | GetFeatureOfInterestId |
GetObservation
tag = procedure | feature
value = value for the corresponding tag element
```

The FTP file list is a single configuration file that contains details of any number of FTP input files to the WPS process. A URL to the FTP file list should be passed in an Execute request. The file itself should be accessible via an external FTP site.

Each FTP file is a type of SOS response. Whenever the WPS needs to issue a SOS request it will first check the files downloaded from the FTP site to see if the call to the SOS has already been made; if a suitable FTP file exists it will be used in place of a request to the SOS.

Note: It is the client's responsibility to ensure an up-to-date SOS response is provided. If an old SOS response is stored on the FTP site, with old SOS results values, then it will be used and treated as the latest values; this might cause synchronization problems if a mixture of cached SOS responses and live responses are used. The principle is "clients beware".

The FTP file list contains metadata that allows the WPS to link the SOS response file to a specific dataset (needed in case multiple SOS input sources are provided) and for DescribeSensor and GetFeatureOfInterestId responses also links them to their procedure and feature respectively.

A client can obtain dataset names from the allowedValues field in the DescribeProcess response of the WPS.

**Execute [response for job started]**

```
<?xml version="1.0" encoding="utf-8"?>
<ExecuteResponse service="WPS" version="1.0.0" xml:lang="en-CA"
serviceInstance="" statusLocation="ftp_url"
xmlns="http://www.opengis.net/wps/1.0.0"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

 <wps:Process wps:processVersion="1">
  <ows:Identifier>process_name</ows:Identifier>
 </wps:Process>

 <wps:Status creationTime="creation_time">
  <wps:ProcessAccepted>Fusion process is about to start</wps:ProcessAccepted>
 </wps:Status>

 <DataInputs>

  Note: Identical to the execute request. If the request 'lineage' attribute
is set to false this section will be missing completely.

 </DataInputs>

 <OutputDefinitions>
   <Output asReference="true">
    <ows:Identifier>CSV_Result_Data</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>CSV_Result_Metadata</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>KML_Result_Data</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>OM_Result_Data</ows:Identifier>
   </Output>
 </OutputDefinitions>

</ExecuteResponse>

ftp_url = FTP URL for this status file
process_name = KrigingProcess | PECK (TBD)
creation_time = timestamp when fusion job was started, e.g. 2009-02-
17T15:11:36 (i.e. OM Result time). This is result time, not sample time.
```

The immediate WPS execute response is saved to a FTP site (as per the WPS specification for requests where status="true" storeExecuteResponse="true"). The statusLocation attribute contains the URL to the file that must be polled by the client. It will be an FTP URL.

**Execute [response for job running]**

```
<?xml version="1.0" encoding="utf-8"?>
<ExecuteResponse service="WPS" version="1.0.0" xml:lang="en-CA"
serviceInstance="" statusLocation="ftp_url"
xmlns="http://www.opengis.net/wps/1.0.0"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

 <wps:Process wps:processVersion="1">
  <ows:Identifier>process_name</ows:Identifier>
 </wps:Process>

 <wps:Status creationTime="creation_time">
  <wps:ProcessStarted percentageComplete="percentage_done">Fusion process is
currently running OK</wps:ProcessStarted>
 </wps:Status>

 <DataInputs>

  Note: Identical to the execute request. If the request 'lineage' attribute
is set to false this section will be missing completely.

 </DataInputs>

 <OutputDefinitions>
   <Output asReference="true">
    <ows:Identifier>CSV_Result_Data</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>CSV_Result_Metadata</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>KML_Result_Data</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>OM_Result_Data</ows:Identifier>
   </Output>
 </OutputDefinitions>

</ExecuteResponse>

ftp_url = FTP URL for this status file
process_name = KrigingProcess | PECK (TBD)
creation_time = timestamp when fusion job was started, e.g. 2009-02-
17T15:11:36 (i.e. OM Result time). This is result time, not sample time.
percentage_done = a crude percentage complete that cannot be used to estimate
how long a job will take to finish due to non-linear nature of most fusion
jobs [Integer]
```

The status report whilst a job is running will contain a percentageComplete field to report the progress of the job execution. This percentage is not something that a client can use to generate time estimates for WPS completion time since fusion jobs are rarely linear. However it is useful for a client UI to report as it shows the job is alive and running and with some previous experience of a job type users can still get a 'feel' for how much time a job has left to run.

**Execute [response for job finished]**

```
<?xml version="1.0" encoding="utf-8"?>
<ExecuteResponse service="WPS" version="1.0.0" xml:lang="en-CA"
serviceInstance="" statusLocation="ftp_url"
xmlns="http://www.opengis.net/wps/1.0.0"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

 <wps:Process wps:processVersion="1">
  <ows:Identifier>process_name</ows:Identifier>
 </wps:Process>

 <wps:Status creationTime="creation_time">
  <wps:ProcessSucceeded>Fusion process script completed ok, output files
ready for download</wps:ProcessSucceeded>
 </wps:Status>

 <DataInputs>

  Note: Identical to the execute request. If the request 'lineage' attribute
is set to false this section will be missing completely.

 </DataInputs>

 <OutputDefinitions>
   <Output asReference="true">
    <ows:Identifier>CSV_Result_Data</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>CSV_Result_Metadata</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>KML_Result_Data</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>OM_Result_Data</ows:Identifier>
   </Output>
 </OutputDefinitions>

 <ProcessOutputs>
  <Output>
   <Identifier>CSV_Result_Data</Identifier>
   <Reference href="ftp_url"/>
  </Output>
  <Output>
   <Identifier>CSV_Result_Metadata</Identifier>
   <Reference href="ftp_url"/>
  </Output>
  <Output>
   <Identifier>KML_Result_Metadata</Identifier>
   <Reference href="ftp_url"/>
  </Output>
  <Output>
   <Identifier>OM_Result_Data</Identifier>
   <Reference href="ftp_url"/>
  </Output>
```

```
  Note: There will be one set of outputs (above) per result set generated.
The number of result sets generated will depend on the number of observed
properties requested, and the nature of the fusion process chosen. Example is
for the KrigingProcess.

 </ProcessOutputs>
</ExecuteResponse>

process_name = KrigingProcess | PECK (TBD)
creation_time = timestamp when fusion job was started, e.g. 2009-02-
17T15:11:36 (i.e. OM Result time). This is result time, not sample time.
ftp_url = FTP URL to a result file or status file on the IT-INNOV FTP server.
It will be either a CSV, KML or XML file depending on the Output / Status
type.
```

When a job finishes the ProcessOuputs element is added to the final WPS status report. This will contain reference entries for each output file generated by the fusion process.

In Kriging there will be a CSV data and metadata file generated for each observed property that was kriged. There will also be an XML file in the O&M format per result set that was kriged.

The CSV data files are the raw data in its 'pure' form. These are the principle output created by the kriging process. There can be several properties per result set, depending on the fusion process; kriging will probably output a single property per result set however.

The ETHZ mapping service can read the O&M files. KML files are for display in Google Earth.

The O&M result XML files are for input into the process chaining service to allow workflow driven WPS execution. These files will also be used to upload results into a Fusion SOS based on the IITB Web Genesis solution. XML formats will be either GetObservationResponse, SamplingSurface or SensorML.

**Execute [response job failed]**

```
<?xml version="1.0" encoding="utf-8"?>
<ExecuteResponse service="WPS" version="1.0.0" xml:lang="en-CA"
serviceInstance="" statusLocation="ftp_url"
xmlns="http://www.opengis.net/wps/1.0.0"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

 <wps:Process wps:processVersion="1">
  <ows:Identifier>process_name</ows:Identifier>
 </wps:Process>

 <wps:Status creationTime="creation_time">
  <wps:ProcessFailed>
   <ExceptionReport>ogc_exception_report</ExceptionReport>
  </wps:ProcessStarted>
 </wps:Status>

 <DataInputs>

  Note: Identical to the execute request. If the request 'lineage' attribute
is set to false this section will be missing completely.

 </DataInputs>

 <OutputDefinitions>
   <Output asReference="true">
    <ows:Identifier>CSV_Result_Data</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>CSV_Result_Metadata</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>KML_Result_Data</ows:Identifier>
   </Output>
   <Output asReference="true">
    <ows:Identifier>OM_Result_Data</ows:Identifier>
   </Output>
 </OutputDefinitions>

</ExecuteResponse>

ftp_url = FTP URL for this status file
process_name = KrigingProcess | PECK (TBD)
creation_time = timestamp when fusion job was started, e.g. 2009-02-
17T15:11:36 (i.e. Result time)
ogc_exception_report = OGC exception report detailing some information about
why the fusion job failed
```

If a job fails an OGC exception will be generated with some information about the reason for this failure. If this text description is not sufficient the WPS provider will need to be contacted and the job creation time and job number quoted to investigate server side log files.

### 6.1.4 IT-INNOV WPS Result set formats

**Result set format [CSV]**

```
CSV_Result_Metadata = grid1-metadata.csv

# col index, phen URN, col URN, unit, dist URN, percentile
0,,urn:ogc:def:phenomenon:latitude,deg,,
1,,urn:ogc:def:phenomenon:longitude,deg,,
2,,urn:ogc:def:phenomenon:altitude,m,,
3,,urn:sany:def:phenomenon:windspeed,m/s,,
4,,urn:sany:def:phenomenon:winddirection,deg,,
5,urn:sany:def:phenomenon:windspeed,urn:ogc:def:statistic:SANY:2009.05:mean,m
/s,urn:ogc:def:distribution:SANY:2009.05:gaussian,
6,urn:sany:def:phenomenon:windspeed,urn:ogc:def:statistic:SANY:2009.05:standa
rd-deviation,m/s,urn:ogc:def:distribution:SANY:2009.05:gaussian,
7,urn:sany:def:phenomenon:winddirection,urn:ogc:def:statistic:SANY:2009.05:mo
de,deg,#dist1,
8,urn:sany:def:phenomenon:winddirection,urn:ogc:def:statistic:SANY:2009.05:pe
rcentile,deg,#dist1,0.025
9,urn:sany:def:phenomenon:winddirection,urn:ogc:def:statistic:SANY:2009.05:pe
rcentile,deg,#dist1,0.250
10,urn:sany:def:phenomenon:winddirection,urn:ogc:def:statistic:SANY:2009.05:p
ercentile,deg,#dist1,0.500
11,urn:sany:def:phenomenon:winddirection,urn:ogc:def:statistic:SANY:2009.05:p
ercentile,deg,#dist1,0.750
12,urn:sany:def:phenomenon:winddirection,urn:ogc:def:statistic:SANY:2009.05:p
ercentile,deg,#dist1,0.975

Issue: number of rows needed for easy client parsing

CSV_Result_Data = grid1-data.csv

54.1,18.5,10,6.5,90,6.5,1.5,90,5,60,80,95,105
54.5,18.7,10,6.5,90,6.5,1.5,90,6,64,81,94,101
55,19,10,6.5,90,6.5,1.5,90,5,63,84,94,101
…
```

Depending on the type of processing performed there can be a single or multiple properties per result set. The CSV metadata file will fully describe the result block. A client parsing this data must first read the CSV metadata to acquire the column labels and then read the CSV data file.

CSV_Result_Metadata: This is the first file that must be read in since it defines the column indexes for the data file. Each row defines the labels for a range of rows in the data file and the associated statistics information available.

CSV_Result_Data: Each row contains a fusion result with any associated information. In the case of spatial interpolation this is an interpolated point (lat,long) and its interpolated value (windspeed, motionX etc).

**Result set format [SamplingSurface]**

```
<?xml version="1.0" encoding="utf-8"?>
<sa:SamplingSurface
 xmlns:sa="http://www.opengis.net/sampling/1.0"
 xmlns:gml="http://www.opengis.net/gml"
 xmlns:xlink="http://www.w3.org/1999/xlink"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 gml:id="sampling_feature_id">

 Note: envelope for entire surface

 <gml:boundedBy>
  <gml:Envelope srsName="sany_srs">
   <gml:lowerCorner>grid_lat_bottom grid_long_left</gml:lowerCorner>
   <gml:upperCorner>grid_lat_top grid_long_right</gml:upperCorner>
  </gml:Envelope>
 </gml:boundedBy>

 <sa:sampledFeature>
  <gml:MultiPointCoverage>

   Note: envelope for this feature

   <gml:boundedBy>
    <gml:Envelope srsName="sany_srs">
     <gml:lowerCorner>grid_lat_bottom grid_long_left</gml:lowerCorner>
     <gml:upperCorner>grid_lat_top grid_long_right</gml:upperCorner>
    </gml:Envelope>
   </gml:boundedBy>

   <gml:multiPointDomain>
    <gml:MultiPoint srsName="sany_srs" gml:id="point_array_id">
     <gml:pointMembers>
      <gml:Point gml:id="point_id">
       <gml:pos>point_lat point_long</gml:pos>
      </gml:Point>

      Note: there will be many points in a coverage

     </gml:pointMembers>
    </gml:MultiPoint>
   </gml:multiPointDomain>

   <gml:rangeSet>
    <gml:File>
     <gml:rangeParameters/>
     <gml:fileName>obs_response_file</gml:fileName>
     <gml:fileStructure>Record Interleaved</gml:fileStructure>
    </gml:File>
   </gml:rangeSet>

  </gml:MultiPointCoverage>
 </sa:sampledFeature>

 <sa:shape/>

 Note: shape is empty since we only need to define the sampledFeature
```

```
</sa:SamplingSurface>

sampling_feature_id = unique ID of feature (e.g. job_1_sampling_surface)
point_array_id = unique ID for point array [mandatory, meaningless]
point_id = unique ID for point [mandatory, meaningless]
point_lat = latitude of an interpolated point
point_long = longitude of an interpolated point
sany_srs = urn:ogc:def:crs:EPSG:4326
grid_lat_bottom = latitude of bottom of spatial grid (e.g. ~53 for gdansk)
grid_long_left = longitude of left of spatial grid (e.g. ~18 for gdansk)
grid_lat_top = latitude of top of spatial grid
grid_long_right = longitude of right of spatial grid
obs_response_file = FTP URL of the observation response file with result data
```

Each fusion result set (i.e. observation) refers to a sampling surface. If several result sets refer to the same sampling surface this surface definition must be cloned as the SA schema do not allow multiple File references. The sampling surface will be stored on the IT-INNOV FTP site.

A result set can be any shape, hence the use of a gml:MultiPointCoverage element. In kriging the basic result is a true uniform grid but the client can optionally request additional points for that grid. It is therefore not suggested that Grid or RectifiedGrid to be used to define the sampledFeature for kriging. For this same reason the shape is defined as a simple envelope that covers the smallest area to completely contain all the sampled points.

Temporal results will need to use a different SamplingFeatureType element.

**Result set format [SensorML provenance record]**

```
<?xml version="1.0" encoding="utf-8"?>
<sml:SensorML xmlns:sml="http://www.opengis.net/sensorML/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1.0">

 <sml:member>
  <sml:ProcessModel>

   <sml:identification>
    <sml:IdentifierList>
     <sml:identifier name="uniqueID">
      <sml:Term definition="urn:ogc:def:identifier:OGC:uniqueID">
       <sml:value>fusion_job_id</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="manufacturerName">
      <sml:Term
       definition="urn:ogc:def:identifier:OGC:1.0.1:manufacturerName">
       <sml:value>fusion_owner</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="longName">
      <sml:Term definition="urn:ogc:def:identifier:OGC:1.0.1:longname">
       <sml:value>fusion_job_long_desc</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="shortName">
      <sml:Term
       definition="urn:ogc:def:identifier:OGC:1.0.1:shortName">
       <sml:value>fusion_job_short_desc</sml:value>
      </sml:Term>
     </sml:identifier>
    </sml:IdentifierList>
   </sml:identification>

   <sml:inputs>
    <sml:InputList>
     <sml:input name="fusion_param_id">
      <swe:Text>
       <gml:description>fusion_param_desc</gml:description>
       <swe:value>fusion_param_value_text</swe:value>
      </swe:Text>
     </sml:input>
     <sml:input name="fusion_param_id">
      <swe:Quantity>
       <gml:description>fusion_param_desc</gml:description>
       <swe:uom code="unit_urn"/>

     Note: unit element can be empty string if no unit is appropriate

       <swe:value>fusion_param_value_numeric</swe:value>
      </swe:Quantity>
     </sml:input>
     <sml:input name="fusion_param_id">
      <swe:Time>
```

```
      <gml:description>fusion_param_desc</gml:description>
      <swe:value>fusion_param_value_time</swe:value>
     </swe:Time>
    </sml:input>

  Note: there will be many parameter entries

   </sml:InputList>
  </sml:inputs>

  <sml:outputs>
   <sml:OutputList>
    <sml:output name="OutputFile">
     <swe:Text>
      <gml:description>fusion_output_desc</gml:description>
      <swe:value>fusion_output_URL</swe:value>
     </swe:Text>
    </sml:output>

  Note: there will be many output URL entries

   </sml:OutputList>
  </sml:outputs>

  <sml:parameters>
   <sml:ParameterList>
    <sml:parameter name="fusion_param_id">
     <swe:Text>
      <gml:description>fusion_param_desc</gml:description>
      <swe:value>fusion_param_value_text</swe:value>
     </swe:Text>
    </sml:parameter>
    <sml:parameter name="fusion_param_id">
     <swe:Quantity>
      <gml:description>fusion_param_desc</gml:description>
      <swe:uom code="unit_urn"/>

    Note: unit element can be empty string if no unit is appropriate

      <swe:value>fusion_param_value_numeric</swe:value>
     </swe:Quantity>
    </sml:parameter>
    <sml:parameter name="fusion_param_id">
     <swe:Time>
      <gml:description>fusion_param_desc</gml:description>
      <swe:value>fusion_param_value_time</swe:value>
     </swe:Time>
    </sml:parameter>

  Note: there will be many parameter entries

   </sml:ParameterList>
  </sml:parameters>

  <sml:method>
   <sml:ProcessMethod>
    <gml:description>fusion_script_desc</gml:description>
    <sml:rules>
     <sml:RulesDefinition>
      <gml:description>
```

```
        fusion_script_io_info
       </gml:description>
      </sml:RulesDefinition>
     </sml:rules>
     <sml:algorithm>
      <sml:AlgorithmDefinition>
       <gml:description>
        fusion_algorithm_desc
       </gml:description>
      </sml:AlgorithmDefinition>
     </sml:algorithm>
     <sml:implementation>
      <sml:ImplementationCode
       language="fusion_impl_lang"
       version="fusion_impl_version">
       <gml:description>
        Fusion_plugin_desc
       </gml:description>
      </sml:ImplementationCode>
     </sml:implementation>

  Note: There will be an implementation entry per fusion c# plug-in used

    </sml:ProcessMethod>
   </sml:method>

  </sml:ProcessModel>
 </sml:member>
</sml:SensorML>

fusion_job_id = unique name of the fusion run instance (e.g. job_1)
fusion_owner = WPS owner (e.g. IT Innovation Centre)
fusion_job_long_desc = long name of fusion job (e.g. Ordinary Kriging with
AVMS - job 1)
fusion_job_short_desc = short name of fusion job (e.g. Kriging - job 1)
fusion_param_id = id of parameter passed by client or statically defined
(e.g. EventTime)
fusion_param_desc = description of parameter (e.g. Sampling time used to
request sensor data from SOS)
fusion_param_value_text = textual values (e.g. true)
fusion_param_value_numeric = numeric values (e.g. 54.2)
fusion_param_value_time = temporal values (e.g. 2005-04-05T12:00:00+01)
fusion_output_desc = description of output param
fusion_output_URL = URL
fusion_name = internal name of the fusion process (e.g. Ordinary Kriging)
fusion_master_script_id = internal name of fusion script (e.g.
script_master_kriging)
fusion_script_desc = internal description of fusion script (e.g. Correlation,
Syntax checking, Normalization, Elevation correction, Ordinary Kriging, AVMS)
fusion_script_io_info = internal description of any input / output
restrictions (e.g. Number of sensors must be > 10 to achieve reasonable
results)
fusion_algorithm_desc = internal description of the algorithm components used
(e.g. OrdinaryKriging, ElevationCorrection, AVMS)
fusion_impl_lang = language of algorithm plugin (e.g. c#)
fusion_impl_version = version of plugin (e.g. 1.0)
Fusion_plugin_desc = name of fusion plugin (e.g. OrdinaryKriging)
```

There is a unique SensorML file per successful fusion Execute request. This file describes the type of fusion executed and the input parameters provided. The purpose of this SensorML is to provide a provenance record to allow a fusion experiment to be reproduced. In principle identical fusion runs should use the same SensorML file, but this caching behaviour (and associated synchronization functionality) is probably beyond the scope of what we will achieve within SANY; there is also the difficulty between the schema enforced one to one mapping between sampled surface and observation response file.

sml:parameters is a list of the algorithm config parameters.

sml:inputs is a list of WPS input parameters.

sml:outputs is a list of WPS output files created at the end of the processing act.

It has been discussed that all SOS responses downloaded during the processing of an Execute request must be cached to ensure perfect provenance; this is because the sensor data in any SOS might change after the processing has occurred. If we have time the SOS response files will be uploaded to the IITB SOS to provide a perfect provenance record of SANY processing.

**Result set format [GetObservationResponse]**

```
<?xml version="1.0" encoding="utf-8"?>
<om:Observation xmlns:om="http://www.opengis.net/om/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:swe="http://www.opengis.net/swe/1.0.1"
xmlns:un="http://www.uncertml.org"
xmlns:swe2="http://www.opengis.net/swe/1.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi=http://www.w3.org/2001/XMLSchema-instance gml:id="fusion_job_id">

   <gml:boundedBy>
    <gml:Envelope srsName="sany_srs">
     <gml:lowerCorner>grid_lat_bottom grid_long_left</gml:lowerCorner>
     <gml:upperCorner>grid_lat_top grid_long_right</gml:upperCorner>
    </gml:Envelope>
   </gml:boundedBy>

   <om:samplingTime>
    <gml:TimePeriod>
     <gml:beginPosition>sample_time_begin</gml:beginPosition>
     <gml:endPosition>sample_time_end</gml:endPosition>
    </gml:TimePeriod>
   </om:samplingTime>

   <om:resultTime>
    <gml:TimeInstant>
     <gml:timePosition>result_time</gml:timePosition>
    </gml:TimeInstant>
   </om:resultTime>

   <om:procedure xlink:href="fusion_job_sensorML_url"/>

   <om:observedProperty>
    <swe:CompositePhenomenon
     gml:id="composite_id" dimension="number_of_phen">
     <gml:name>result set composite phenomenon</gml:name>
     <swe:component xlink:href="fusion_property_urn"/>

     Note: there can be many fusion result properties (i.e. components)

    </swe:CompositePhenomenon>
   </om:observedProperty>

   <om:featureOfInterest xlink:href="fusion_job_sampling_surface_url"/>

   <om:result>
    <swe:DataArray>
     <swe:elementCount>
      <swe:Count>
       <swe:value>number_of_results</swe:value>
      </swe:Count>
     </swe:elementCount>
     <swe:elementType name="Fusion result set description">
      <swe:SimpleDataRecord gml:id="result_set_id">
       <swe:field name="result_property_urn" xlink:href="stats_xpointer">
        <swe:Quantity>
         <swe:uom code="unit_urn"/>
        </swe:Quantity>
```

```
      </swe:field>

  Note: there can be many result_property entries, each with a field
  Note: the result_property's must include all fusion_property's

     </swe:SimpleDataRecord>
    </swe:elementType>

    <swe:encoding>
     <swe:TextBlock
      blockSeparator="@@"
      decimalSeparator="."
      tokenSeparator=","/>
    </swe:encoding>
    <swe:values>
     … result data in format described in simple data record
     e.g. 49.0156,8.4252,15.2053,77.324,1004.61@@ …
    </swe:values>
   </swe:DataArray>

   <un:StatisticsArray definition="stats_definition">
    <un:elementType>
     <un:StatisticsRecord>
      <un:field>
       <un:Statistic definition="stat_definition">
        <un:parameters>
         <un:Parameter definition="dist_definition"/>
        </un:parameters>
       </un:Statistic>
      </un:field>
     </un:StatisticsRecord>
    </un:elementType>
    <un:elementCount>number_of_results</un:elementCount>
    <swe2:encoding>
     <swe2:TextBlock
      blockSeparator="@@"
      decimalSeparator="."
      tokenSeparator=","/>
    </swe2:encoding>
    <swe2:values>
     … stats data in format described in stats record
     e.g. 49.6372@@ …
    </swe2:values>
   </un:StatisticsArray>

   <un:DistributionArray definition="dist_definition">
    <un:elementType>
     <un:Distribution
       definition="urn:sany:def:distribution:non-analytical">
      <un:parameters>
       <un:Parameter definition="urn:sany:def:statistic:percentile">
        <un:value>percentile_value</un:value>
       </un:Parameter>

       Note: there can be many percentiles defined

      </un:parameters>
     </un:Distribution>
    </un:elementType>
    <un:elementCount>number_of_results</un:elementCount>
```

```
      <swe2:encoding>
       <swe2:TextBlock
        blockSeparator="@@" decimalSeparator="." tokenSeparator=","/>
      </swe2:encoding>
      <swe2:values>
       … dist data in format described in dist record
       e.g. 49.6372@@ …
      </swe2:values>
     </un:DistributionArray>

 Note: there can be many un:StatisticsArray and un:DistributionArray entries

   </om:result>
</om:Observation>

sany_srs = urn:ogc:def:crs:EPSG:4326
grid_lat_bottom = latitude of bottom of spatial grid (e.g. ~53 for gdansk)
grid_long_left = longitude of left of spatial grid (e.g. ~18 for gdansk)
grid_lat_top = latitude of top of spatial grid
grid_long_right = longitude of right of spatial grid
fusion_job_id = unique name of the fusion run instance (e.g. job_1)
sample_time_begin = input data sample time start
sample_time_end = input data sample time end
result_time = start time of fusion processing
fusion_job_sensorML_url = URL to SensorML description of fusion job (see
section 3.3.4)
number_of_phen = number of result phenomenon
composite_id = gml:id for composite [mandatory, meaningless]
fusion_property_urn = URN to observed property that will be focus of fusion
fusion_job_sampling_surface_url = URL to the shape of the result set (see
section 3.3.3)
number_of_results = number of individual points in result shape feature
result_set_id = id for data record [mandatory in schema, meaningless]
result_property_urn = URN to properties in result set. The
fusion_property_urn's must appear within the set of result_property_urn's
unless they have been transformed into a new result type. (e.g. spatial
interpolation point will report windspeed, longitude and latitude)
stats_urn = URN to a set of stats for result set (e.g. stats1)
stat_xpointer = XPointer to a StatisticsArray element inthis document
stat_definition = URL of a specific statistic definition (e.g. mean)
dist_ definition = URL to a distribution definition (e.g. guassian)
unit_urn = URN for the unit of this result value (e.g. m/s)
```

There can be either one GetObservationResponse file per result set (i.e. observation) or a single aggregated GetObservationResponse file for all result sets, depending on what type of fusion process is executed.

Some information about the input parameters is returned in the observation itself but the main reference source is the SensorML link to the fusion_job_sensorML_url (section 6.1.4) and fusion_job_sampling_surface_url (section 6.1.4).

The result set values are returned in the usual SWE SimpleDataRecord and associated encoded data block. The result definition may contain href links to a separate uncertML value block where specific values have uncertainty/error information associated with them. In this case the uncertML block will have a statistics values (e.g. variance) for each value that appears in the result block. This is why the count for uncertML and SimpleDataRecord is the same.

Other complex uncertML entries are possible (e.g. un:DistributionArray) but these have been left unspecified until needed.

Unit URN's will be taken where possible from the SOS providing input data. Where new properties are reported well known URN's will be defined for the units (e.g. variance of wind speed has a unit of $(m/s)^2$) or left undefined. Statistics values often leave units undefined as they can be derived from the variable they are associated with.

**Result set format [KML]**

```
Defined in Google KML specification
http://code.google.com/apis/kml/documentation/
```

IT-INNOV create KML output for easy visualization of spatial result sets in Google earth. This KML output can be used in addition to the SANY mapping service accesed via the SSE portal.

## 6.2. IITB I/O specification

### 6.2.1 IITB Fusion SOS GetCapabilities

**GetCapabilities [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCapabilities xmlns="http://www.opengis.net/sos/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.iitb.fraunhofer.de/sos"
xmlns:ows="http://www.opengis.net/ows/1.1" service="SOS"/>
```

**GetCapabilities [response]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:Capabilities version="1.0.0"
xsi:schemaLocation="http://www.opengis.net/sos/1.0
http://schemas.opengis.net/sos/1.0.0/sosAll.xsd"
xmlns:sos="http://www.opengis.net/sos/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink">
 <ows:ServiceIdentification xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:om="http://www.opengis.net/om/1.0">
  <ows:Title>Testbed FUSION-SOS 1.0.0 Fraunhofer IITB</ows:Title>
  <ows:Abstract>Testbed Deployment of the FUSION SOS 1.0.0 (realised with
Webgenesis) at Fraunhofer IITB</ows:Abstract>
  <ows:Keywords>
   <ows:Keyword>fusion, sos, fusionsos, webgenesis</ows:Keyword>
  </ows:Keywords>
  <ows:ServiceType
codeSpace="http://opengeospatial.net">OGC:SOS</ows:ServiceType>
   <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
   <ows:Fees>NONE</ows:Fees>
   <ows:AccessConstraints>NONE</ows:AccessConstraints>
  </ows:ServiceIdentification>
  <ows:ServiceProvider xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:om="http://www.opengis.net/om/1.0">
   <ows:ProviderName>Fraunhofer Institute for Information Technology and Data
Processing</ows:ProviderName>
   <ows:ProviderSite xlink:href="http://www.iitb.fraunhofer.de"/>
   <ows:ServiceContact>
    <ows:IndividualName>Dr. Kym Watson</ows:IndividualName>
    <ows:PositionName>Research Associate</ows:PositionName>
    <ows:ContactInfo>
     <ows:Phone>
      <ows:Voice>+49-721-6091</ows:Voice>
     </ows:Phone>
     <ows:Address>
      <ows:DeliveryPoint>Fraunhoferstr. 1</ows:DeliveryPoint>
      <ows:City>Karlsruhe</ows:City>
      <ows:AdministrativeArea>BW</ows:AdministrativeArea>
      <ows:PostalCode>76131</ows:PostalCode>
      <ows:Country>Germany</ows:Country>
      <ows:ElectronicMailAddress>
        kym.watson@iitb.fraunhofer.de
      </ows:ElectronicMailAddress>
     </ows:Address>
    </ows:ContactInfo>
    <ows:Role/>
   </ows:ServiceContact>
  </ows:ServiceProvider>
  <ows:OperationsMetadata xmlns:ows="http://www.opengis.net/ows/1.1">
   <ows:Operation name="GetCapabilities">
    <ows:DCP>
     <ows:HTTP>
      <ows:Get xlink:href="http://fusion-sos-
test.iitb.fhg.de/servlet/is/Entry..SOSRequest/?"/>
```

```
      <ows:Post xlink:href="http://fusion-sos-
test.iitb.fhg.de/servlet/is/Entry..SOSRequest/"/>
     </ows:HTTP>
    </ows:DCP>
    <ows:Parameter name="service">
     <ows:AllowedValues>
      <ows:Value>SOS</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="AcceptVersions">
     <ows:AllowedValues>
      <ows:Value>1.0.0</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="Sections">
     <ows:AllowedValues>
      <ows:Value>All</ows:Value>
      <ows:Value>ServiceIdentification</ows:Value>
      <ows:Value>ServiceProvider</ows:Value>
      <ows:Value>OperationsMetadata</ows:Value>
      <ows:Value>Contents</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
   </ows:Operation>
   <ows:Operation name="DescribeSensor">
    <ows:DCP>
     <ows:HTTP>
      <ows:Post xlink:href="http://fusion-sos-
test.iitb.fhg.de/servlet/is/Entry..SOSRequest/"/>
     </ows:HTTP>
    </ows:DCP>
    <ows:Parameter name="service">
     <ows:AllowedValues>
      <ows:Value>SOS</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="version">
     <ows:AllowedValues>
      <ows:Value>1.0.0</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="outputFormat">
     <ows:AllowedValues>
      <ows:Value>text/xml;subtype="sensorML/1.0.1"</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="procedure">
     <ows:AllowedValues>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:160</ows:Value>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:161</ows:Value>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:162</ows:Value>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:180</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
   </ows:Operation>
   <ows:Operation name="GetObservation">
    <ows:DCP>
     <ows:HTTP>
      <ows:Post xlink:href="http://fusion-sos-
test.iitb.fhg.de/servlet/is/Entry..SOSRequest/"/>
```

```
      </ows:HTTP>
    </ows:DCP>
    <ows:Parameter name="service">
     <ows:AllowedValues>
      <ows:Value>SOS</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="version">
     <ows:AllowedValues>
      <ows:Value>1.0.0</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="srsName">
     <ows:AllowedValues>
      <ows:Value>urn:ogc:def:crs:EPSG:4326</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="offering">
     <ows:AllowedValues>
      <ows:Value>DefaultOffering</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="eventTime">
     <ows:AllowedValues>
      <ows:Range>
       <ows:MinimumValue>2006-10-16T01:00:00+02:00</ows:MinimumValue>
       <ows:MaximumValue>2006-11-13T00:00:00+02:00</ows:MaximumValue>
      </ows:Range>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="procedure">
     <ows:AllowedValues>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:160</ows:Value>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:161</ows:Value>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:162</ows:Value>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:180</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="observedProperty">
     <ows:AllowedValues>
      <ows:Value>urn:ogc:def:phenomenon:OGC:1.0:wind-spd</ows:Value>
      <ows:Value>urn:ogc:def:phenomenon:OGC:1.0:wind-dir</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="featureOfInterest">
     <ows:AllowedValues>
      <ows:Value>
        urn:ogc:def:featureType:OGC:1.0:SamplingSurface:163
      </ows:Value>
      <ows:Value>
        urn:ogc:def:featureType:OGC:1.0:SamplingSurface:164
      </ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="responseFormat">
     <ows:AllowedValues>
      <ows:Value>text/xml;subtype="om/1.0.0"</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Parameter name="resultModel">
```

```
      <ows:AllowedValues>
       <ows:Value>Observation</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="responseMode">
      <ows:AllowedValues>
       <ows:Value>inline</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="GetObservationById">
     <ows:DCP>
      <ows:HTTP>
       <ows:Post xlink:href="http://fusion-sos-
test.iitb.fhg.de/servlet/is/Entry..SOSRequest/"/>
      </ows:HTTP>
     </ows:DCP>
     <ows:Parameter name="service">
      <ows:AllowedValues>
       <ows:Value>SOS</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="version">
      <ows:AllowedValues>
       <ows:Value>1.0.0</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="srsName">
      <ows:AllowedValues>
       <ows:Value>urn:ogc:def:crs:EPSG:4326</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="ObservationId">
      <ows:AllowedValues>
       <ows:Value>urn:ogc:def:featureType:OGC:1.0:Observation:174</ows:Value>
       <ows:Value>urn:ogc:def:featureType:OGC:1.0:Observation:173</ows:Value>
       <ows:Value>urn:ogc:def:featureType:OGC:1.0:Observation:172</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="responseFormat">
      <ows:AllowedValues>
       <ows:Value>text/xml;subtype="om/1.0.0"</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="resultModel">
      <ows:AllowedValues>
       <ows:Value>Observation</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="responseMode">
      <ows:AllowedValues>
       <ows:Value>inline</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="GetFeatureOfInterest">
     <ows:DCP>
      <ows:HTTP>
       <ows:Post xlink:href="http://fusion-sos-
test.iitb.fhg.de/servlet/is/Entry..SOSRequest/"/>
```

```
      </ows:HTTP>
     </ows:DCP>
     <ows:Parameter name="service">
      <ows:AllowedValues>
       <ows:Value>SOS</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="version">
      <ows:AllowedValues>
       <ows:Value>1.0.0</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="featureOfInterestId">
      <ows:AllowedValues>
       <ows:Value>
         urn:ogc:def:featureType:OGC:1.0:SamplingSurface:163
       </ows:Value>
       <ows:Value>
         urn:ogc:def:featureType:OGC:1.0:SamplingSurface:164
       </ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="location">
      <ows:AllowedValues>
       <ows:Value>ogc:BBOX</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="GetFeatureOfInterestTime">
     <ows:DCP>
      <ows:HTTP>
       <ows:Post xlink:href="http://fusion-sos-
test.iitb.fhg.de/servlet/is/Entry..SOSRequest/"/>
      </ows:HTTP>
     </ows:DCP>
     <ows:Parameter name="service">
      <ows:AllowedValues>
       <ows:Value>SOS</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="version">
      <ows:AllowedValues>
       <ows:Value>1.0.0</ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
     <ows:Parameter name="featureOfInterestId">
      <ows:AllowedValues>
       <ows:Value>
         urn:ogc:def:featureType:OGC:1.0:SamplingSurface:163
       </ows:Value>
       <ows:Value>
         urn:ogc:def:featureType:OGC:1.0:SamplingSurface:164
       </ows:Value>
      </ows:AllowedValues>
     </ows:Parameter>
    </ows:Operation>
    <ows:Operation name="RegisterSensor">
     <ows:DCP>
      <ows:HTTP>
```

```
      <ows:Post xlink:href="http://fusion-sos-
test.iitb.fhg.de/servlet/is/Entry..SOSRequest/"/>
     </ows:HTTP>
    </ows:DCP>
    <ows:Constraint name="supportedSensorDescription">
     <ows:AllowedValues>
      <ows:Value>sml:SensorML</ows:Value>
     </ows:AllowedValues>
     <ows:Meaning
ows:reference="urn:ogc:def:swe:SupportedSensorDescription">The service will
only accept sensor descriptions that comply with the listed
ones</ows:Meaning>
    </ows:Constraint>
   </ows:Operation>
   <ows:Operation name="InsertObservation">
    <ows:DCP>
     <ows:HTTP>
      <ows:Post xlink:href="http://fusion-sos-
test.iitb.fhg.de/servlet/is/Entry..SOSRequest/"/>
     </ows:HTTP>
    </ows:DCP>
    <ows:Parameter name="AssignedSensorId">
     <ows:AllowedValues>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:160</ows:Value>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:161</ows:Value>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:162</ows:Value>
      <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:180</ows:Value>
     </ows:AllowedValues>
    </ows:Parameter>
    <ows:Constraint name="supportedObservationDescription">
     <ows:AllowedValues>
      <ows:Value>om:Observation</ows:Value>
     </ows:AllowedValues>
     <ows:Meaning
ows:reference="urn:ogc:def:swe:SupportedObservationDescription">The service
will only accept Observations that comply with the listed ones</ows:Meaning>
    </ows:Constraint>
   </ows:Operation>
  </ows:OperationsMetadata>
  <sos:Contents>
   <sos:ObservationOfferingList>
    <sos:ObservationOffering gml:id="DefaultOffering">
     <gml:name>Default Observation Offering</gml:name>
     <gml:boundedBy>
      <gml:Envelope srsName="urn:ogc:def:crs:EPSG:4326">
       <gml:lowerCorner>54.02694 18.4699895191325</gml:lowerCorner>
       <gml:upperCorner>54.57972 19.1341686645625</gml:upperCorner>
      </gml:Envelope>
     </gml:boundedBy>
     <sos:time>
      <gml:TimePeriod xsi:type="gml:TimePeriodType">
       <gml:beginPosition>2006-10-16T01:00:00+02:00</gml:beginPosition>
       <gml:endPosition>2006-11-13T00:00:00+02:00</gml:endPosition>
      </gml:TimePeriod>
     </sos:time>
     <sos:procedure
       xlink:href="urn:ogc:def:featureType:OGC:1.0.1:Process:160"/>
     <sos:procedure
       xlink:href="urn:ogc:def:featureType:OGC:1.0.1:Process:161"/>
     <sos:procedure
```

```
       xlink:href="urn:ogc:def:featureType:OGC:1.0.1:Process:162"/>
     <sos:procedure
       xlink:href="urn:ogc:def:featureType:OGC:1.0.1:Process:180"/>
     <sos:observedProperty
       xlink:href="urn:ogc:def:phenomenon:OGC:1.0:wind-spd"/>
     <sos:observedProperty
       xlink:href="urn:ogc:def:phenomenon:OGC:1.0:wind-dir"/>
     <sos:featureOfInterest
       xlink:href="urn:ogc:def:featureType:OGC:1.0:SamplingSurface:163"/>
    <sos:featureOfInterest
       xlink:href="urn:ogc:def:featureType:OGC:1.0:SamplingSurface:164"/>
    <sos:responseFormat>text/xml;subtype="om/1.0.0"</sos:responseFormat>
    <sos:resultModel
xmlns:om="http://www.opengis.net/om/1.0">om:Observation</sos:resultModel>
    <sos:responseMode>inline</sos:responseMode>
   </sos:ObservationOffering>
  </sos:ObservationOfferingList>
 </sos:Contents>
</sos:Capabilities>
```

### 6.2.2 IITB Fusion SOS DescribeSensor

**DescribeSensor [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:DescribeSensor
outputFormat="text/xml;subtype=&quot;sensorML/1.0.1&quot;" service="SOS"
version="1.0.0" xmlns:sos="http://www.opengis.net/sos/1.0">
 <sos:procedure>urn:ogc:def:featureType:OGC:1.0.1:Process:197</sos:procedure>
</sos:DescribeSensor>
```

**DescribeSensor [response fusion processing provenance information]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sml:SensorML version="1.0.1"
xsi:schemaLocation="http://www.opengis.net/sensorML/1.0.1
http://schemas.opengis.net/sensorML/1.0.1/sensorML.xsd"
xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
xmlns="http://www.opengis.net/sos/1.0" xmlns:gml="http://www.opengis.net/gml"
xmlns:om="http://www.opengis.net/om/1.0"
xmlns:swe="http://www.opengis.net/swe/1.0.1"
xmlns:tml="http://www.opengis.net/tml"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <sml:member>
  <sml:ProcessModel gml:id="urn_ogc_def_featureType_OGC_1.0.1_Process_4728">
   <gml:methodName>BME interval</gml:methodName>
   <!--System Identifiers-->
   <!--====================================================-->
   <!--sml:identification element must contain ID of the Fusion process-->
   <sml:identification>
    <sml:IdentifierList>
     <sml:identifier name="uniqueID">
      <sml:Term definition="urn:ogc:def:identifier:OGC:uniqueID">
       <sml:value>Watson test 1</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="manufacturerName">
      <sml:Term
        definition="urn:ogc:def:identifier:OGC:1.0.1:manufacturerName">
       <sml:value>Fraunhofer IITB</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="longName">
      <sml:Term definition="urn:ogc:def:identifier:OGC:1.0.1:longname">
       <sml:value>IITB-F010 BME Fusion for interval data</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="shortName">
      <sml:Term definition="urn:ogc:def:identifier:OGC:1.0.1:shortName">
       <sml:value>IITB BME Interval</sml:value>
      </sml:Term>
     </sml:identifier>
    </sml:IdentifierList>
   </sml:identification>
   <!--====================================================-->
   <!--====================================================-->
   <!--System Characteristics-->
   <!--====================================================-->
   <!--sml:characteristics element may be empty-->
   <sml:characteristics/>
   <!--====================================================-->
   <!--System capabilities-->
   <!--====================================================-->
   <!--sml:capabilities element may be empty-->
   <sml:capabilities/>
   <!--list containing the input and output phenomena, may be empty-->
   <sml:inputs>
    <sml:InputList>
     <sml:input name="ObservationData"
xlink:href="http://imtfusos.pc.iitb.fhg.de/servlet/is/2165/assembled-
```

```
observation.xml?command=downloadContent&amp;filename=assembled-
observation.xml" xlink:role="urn:ogc:def:featureType:OGC:1.0:Observation"
xlink:title=""/>
     <sml:input name="SamplingFeature"
xlink:href="http://imtfusos.pc.iitb.fhg.de/servlet/is/2165/SamplingFeature_21
7.xml?command=downloadContent&amp;filename=SamplingFeature_217.xml"
xlink:role="urn:ogc:def:featureType:OGC:1.0:SamplingSurface" xlink:title=""/>
    </sml:InputList>
   </sml:inputs>
   <!--output section;-->
   <sml:outputs>
    <sml:OutputList>
     <sml:output name="kriged_temperature">
      <!--This is name of 'phenomenon' that appears in SPS capabilities-->
      <swe:DataArray>
       <swe:elementCount>
        <swe:Count>
         <!--<swe:value>100</swe:value>-->
        </swe:Count>
       </swe:elementCount>
       <swe:elementType name="Interpolation Fields">
        <swe:DataRecord gml:id="Format-for-Interpolation-Fields_1">
         <swe:field name="latitude">
          <swe:Quantity
            definition="urn:ogc:property:location:EPSG:4326:latitude">
           <swe:uom code="deg"/>
          </swe:Quantity>
         </swe:field>
         <swe:field name="longitude">
          <swe:Quantity
            definition="urn:ogc:property:location:EPSG:4326:longitude">
           <swe:uom code="deg"/>
          </swe:Quantity>
         </swe:field>
         <swe:field name="temperature">
          <swe:Quantity
            definition="urn:ogc:def:property:OGC:1.0.1:temperature">
           <swe:uom code="Cel"/>
          </swe:Quantity>
         </swe:field>
        </swe:DataRecord>
       </swe:elementType>
      </swe:DataArray>
     </sml:output>
    </sml:OutputList>
   </sml:outputs>
   <!--parameter section;-->
   <sml:parameters>
    <sml:ParameterList>
     <sml:parameter name="covmodel">
      <swe:Text>
       <gml:description>name of covariance model, one of nuggetC, sphericalC,
exponentialC, gaussianC, holecosC, holesinC</gml:description>
       <swe:value>exponentialC</swe:value>
      </swe:Text>
     </sml:parameter>
     <sml:parameter name="covparam">
      <swe:Text>
       <gml:description>variance and range of covariance
model</gml:description>
```

```
       <swe:value>20 0.0001</swe:value>
      </swe:Text>
     </sml:parameter>
     <sml:parameter name="nhmax">
      <swe:Quantity>
       <gml:description>max number of hard observations considered in a
neighborhood</gml:description>
       <swe:value>10</swe:value>
      </swe:Quantity>
     </sml:parameter>
     <sml:parameter name="nsmax">
      <swe:Quantity>
       <gml:description>max number of soft observations considered in a
neighborhood</gml:description>
       <swe:value>5</swe:value>
      </swe:Quantity>
     </sml:parameter>
     <sml:parameter name="dmax">
      <swe:Quantity>
       <gml:description>spatial diameter of a neighborhood around an
estimation location</gml:description>
       <swe:value>0.005</swe:value>
      </swe:Quantity>
     </sml:parameter>
     <sml:parameter name="order">
      <swe:Quantity>
       <gml:description>order polynomial of local
interpolation</gml:description>
       <swe:value>0</swe:value>
      </swe:Quantity>
     </sml:parameter>
     <sml:parameter name="options">
      <swe:Text>
       <gml:description>optional parameters for computation</gml:description>
       <swe:value>0 0 50000 1e-4</swe:value>
      </swe:Text>
     </sml:parameter>
    </sml:ParameterList>
   </sml:parameters>
   <!--method section;-->
   <sml:method>
    <sml:ProcessMethod>
     <!--=== Description ===-->
     <gml:description>This process methode uses an Fusion algorithm to
generate Fusion observation data.</gml:description>
     <!--=== Contact ===-->
     <sml:contact xlink:arcrole="urn:ogc:def:property:OGC:1.0:author"
xlink:href="IITB-SiegbertKunz-KymWatson"/>
     <!--=== Rules Set ===-->
     <sml:rules>
      <sml:RulesDefinition>
       <gml:description>The input must be an observation collection. The
output will be a coverage.</gml:description>
       <!--If a profile would be defined for describing the in- and outputs
and parameters expected / required by the ProcessMethod it could be
referenced here.-->
      </sml:RulesDefinition>
     </sml:rules>
     <!--=== Algorithm ===-->
     <sml:algorithm>
```

```
      <sml:AlgorithmDefinition>
       <gml:description>The Fusion algorithm is based on the Bayesian maximum
entropy method for interval PDFs</gml:description>
      </sml:AlgorithmDefinition>
     </sml:algorithm>
     <sml:implementation>
      <sml:ImplementationCode language="JAVA" version="1.5">
       <sml:binaryRef xlink:href="fusion.sps.process.BMEFusionTask"/>
      </sml:ImplementationCode>
     </sml:implementation>
    </sml:ProcessMethod>
   </sml:method>
  </sml:ProcessModel>
 </sml:member>
</sml:SensorML>
```

As in the IT-INNOV result-set (compare 6.1.4), there is a unique SensorML file per successful fusion run (in this case executed via the SPS *Submit* operation). This file describes the fusion algorithm that was executed and the input parameters provided. The purpose of this SensorML is to provide a provenance record to allow a fusion experiment to be reproduced.

sml:parameters is a list of the algorithm config parameters.

sml:inputs is a list of SPS input parameters.

sml:outputs describes the result fields of the O&M Observation created by the fusion-run (the grid coverage).

### 6.2.3 IITB Fusion SOS GetObservationById

Of course the IITB *Fusion SOS* also supports standard *GetObservation* requests, with several filter parameters, as specified in the SOS 1.0 specification. However, in the Fusion processing workflow, the *GetObservationById* request is the one that is most important: After a fusion run has been successfully completed and the fusion result has been uploaded to the *Fusion SOS*, a client can issue a *DescribeResultAccess* request to the SPS (see also section 6.2.11). This *DescribeResultAccess* request will return a prepared *GetObservationByID* request, along with the *Fusion SOS* URL. This *GetObservationById* request, when send to the Fusion SOS URL, will retrieve exactly the uploaded fusion result.

**GetObservationById [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:GetObservationById version="1.0.0" service="SOS"
xmlns:sos="http://www.opengis.net/sos/1.0">
<sos:ObservationId>urn:ogc:def:featureType:OGC:1.0:Observation:5800</sos:Obse
rvationId>
 <sos:responseFormat>text/xml;subtype="om/1.0.0"</sos:responseFormat>
</sos:GetObservationById>
```

**GetObservationById [response fusion process result]**

This is the result produced by a fusion process run, and uploaded afterwards to the *Fusion SOS*.

As in the IT-INNOV approach (compare 6.1.4), the result set values are returned in the usual SWE SimpleDataRecord and associated encoded data block. The result definition may contain href links to a separate uncertML value block where specific values have uncertainty information associated with them. In this case the uncertML block will have a statistics values (e.g. variance) for each value that appears in the result block. This is why the count for uncertML and SimpleDataRecord is the same.

(Please note that the XML example below is abbreviated in the result section, for clarity purposes).

```
<?xml version="1.0" encoding="UTF-8"?>
<om:ObservationCollection xmlns:om="http://www.opengis.net/om/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/om/1.0
http://schemas.opengis.net/om/1.0.0/om.xsd">
 <om:member>
  <om:Observation xmlns:gml="http://www.opengis.net/gml"
xmlns:swe="http://www.opengis.net/swe/1.0.1"
xmlns:swe2="http://www.opengis.net/swe/1.0"
xmlns:un="http://www.uncertml.org" xmlns:xlink="http://www.w3.org/1999/xlink"
gml:id="urn_ogc_def_featureType_OGC_1.0_Observation_5800">
   <gml:boundedBy>
    <gml:Envelope srsName="urn:ogc:def:crs:EPSG:4326">
     <gml:lowerCorner>49.0146 8.42521</gml:lowerCorner>
     <gml:upperCorner>49.0159 8.42742</gml:upperCorner>
    </gml:Envelope>
   </gml:boundedBy>
   <om:samplingTime>
    <gml:TimePeriod>
     <gml:beginPosition>2008-11-06T13:00:00+01:00</gml:beginPosition>
     <gml:endPosition>2008-11-06T14:00:00+01:00</gml:endPosition>
    </gml:TimePeriod>
   </om:samplingTime>
   <om:resultTime>
    <gml:TimeInstant>
     <gml:timePosition>2009-05-20T18:25:32+02:00</gml:timePosition>
    </gml:TimeInstant>
   </om:resultTime>
   <om:procedure
     xlink:href="urn:ogc:def:featureType:OGC:1.0.1:Process:5799"/>
   <om:observedProperty
     xlink:href="urn:ogc:def:property:OGC:1.0.1:temperature"/>
   <om:featureOfInterest
     xlink:href="urn:ogc:def:featureType:OGC:1.0:SamplingSurface:5798"/>
   <om:result>
    <swe:DataArray>
     <swe:elementCount>
      <swe:Count>
       <swe:value>3111</swe:value>
      </swe:Count>
     </swe:elementCount>
     <swe:elementType name="Interpolation Fields">
      <swe:SimpleDataRecord gml:id="Obs5800_Format-for-Interpolation-Fields">
       <swe:field name="latitude">
        <swe:Quantity
          definition="urn:ogc:property:location:EPSG:4326:latitude">
         <swe:uom code="deg"/>
        </swe:Quantity>
       </swe:field>
       <swe:field name="longitude">
        <swe:Quantity
          definition="urn:ogc:property:location:EPSG:4326:longitude">
         <swe:uom code="deg"/>
        </swe:Quantity>
       </swe:field>
       <swe:field name="temperature"
xlink:href="#xpointer(here()/../../../un:StatisticsArray[1]">
        <swe:Quantity
          definition="urn:ogc:def:property:OGC:1.0.1:temperature">
         <swe:uom code="Cel"/>
```

```
        </swe:Quantity>
       </swe:field>
      </swe:SimpleDataRecord>
     </swe:elementType>
     <swe:encoding>
      <swe:TextBlock
        blockSeparator="@@" decimalSeparator="." tokenSeparator=","/>
     </swe:encoding>
<swe:values>49.0158,8.4252,14.3318@@49.0158,8.42522,14.3318@@49.0158,8.42524,
14.3318@@49.0158,8.42526,14.3318@@49.0158,8.42528,14.3318@@49.0158,8.4253,14.
3318@@49.0158,8.42532,14.3318@@49.0158,8.42534,14.3318@@49.0158,8.42536,14.33
18@@49.0158,8.42538,14.3318@@49.0158,8.4254,14.3318@@49.0158,8.42542,14.3318@
@49.0158,8.42544,14.3318@@49.0158,8.42546,14.3319@@49.0158,8.42548,14.3319@@4
9.0158,8.4255,14.6037@@49.0158,8.42552,14.6038@@49.0158,8.42554,14.604@@49.01
58,8.42556,14.6042@@49.0158,8.42558,14.6046@@49.0158,8.4256,14.6051@@49.0158,
8.42562,14.6059@@49.0158,8.42564,14.6068@@49.0158,8.42566,14.6078@@49.0158,8.
42568,14.6089@@49.0158,8.4257,14.6098@@49.0158,8.42572,14.6103@@49.0158,8.425
74,14.6104@@49.0158,8.42576,14.61@@49.0158,8.42578,14.6093@@49.0158,8.4258,14
.6083@@49.0158,8.42582,14.6073@@49.0158,8.42584,14.6064@@49.0158,8.42586,14.6
056@@49.0158,8.42588,14.6051@@49.0146,8.42594,14.1948@@49.0146,8.42596,14.194
7@@49.0146,8.42598,14.1944@@49.0146,8.426,14.1941@@49.0146,8.42602,14.1934@@4
9.0146,8.42604,14.1922@@49.0146,8.42606,14.1901@@49.0146,8.42608,14.1865@@49.
0146,8.4261,13.8939@@49.0146,8.42612,13.8917@@49.0146,8.42614,13.8884@@49.014
6,8.42616,13.8843@@49.0146,8.42618,13.8807@@49.0146,8.4262,13.88@@###abbrevia
ted###</swe:values>
    </swe:DataArray>
    <un:StatisticsArray>
     <un:elementType>
      <un:StatisticsRecord>
       <un:field>
        <un:Statistic definition="www.stat.berkeley.edu/gloss.htm#variance"
name="variance"/>
       </un:field>
      </un:StatisticsRecord>
     </un:elementType>
     <un:elementCount>3111</un:elementCount>
     <swe2:encoding>
      <swe2:TextBlock blockSeparator="@@" decimalSeparator="."
tokenSeparator=","/>
     </swe2:encoding>
<swe2:values>19.958915@@19.958915@@19.958915@@19.958915@@19.958915@@19.958915
@@19.958915@@19.958915@@19.958915@@19.958915@@19.958915@@19.958915@@19.958914
@@19.958913@@19.958911@@19.949698@@19.949692@@19.949682@@19.949664@@19.949629
@@19.949566@@19.949453@@19.949268@@19.948997@@19.948656@@19.948315@@19.948086
@@19.948061@@19.94825@@19.948573@@19.948916@@19.9492@@19.949399@@19.949524@@1
9.949597@@19.949639@@19.949663@@19.949678@@19.949687@@19.949694@@19.949698@@1
9.9497@@19.946809@@19.94681@@19.946811@@19.946811@@19.946812@@19.946812@@19.9
46812@@19.946812@@19.956677@@19.958915@@19.958915@@19.958915@@19.958915@@19.9
58915@@19.958915@@19.958915@@19.958915@@19.958915@@19.958915@@19.958915@@19.9
58914@@19.958913@@19.958912@@19.958909@@19.949694@@19.949686@@19.949668@@19.9
49633@@19.949562@@19.949417@@19.949135@@19.948629@@19.947817@@19.946717@@19.9
45548@@19.944728@@19.944639@@19.945325@@19.946449@@19.947577@@19.948442@@19.9
48998@@19.949315@@19.949486@@19.949576@@19.949626@@19.949656@@19.949674@@19.9
49685@@19.949693@@19.949697@@19.946808@@19.946809@@19.94681@@19.946811@@19.94
6811@@19.946812@@19.946812@@19.946812@@19.956677@@19.958915@@19.958915@@19.95
8915@@###abbreviated###</swe2:values>
    </un:StatisticsArray>
   </om:result>
  </om:Observation>
 </om:member>
```

```
</om:ObservationCollection>
```

The fusion result data section may also contain further quality metadata as a quality flag, e.g. “NaN” indicating that data matches a Not-A-Number value (due to possible sensor failures), as illustrated in the following example fragment of a fusion result.

```
<om:Observation>…
   <om:procedure xlink:href="urn:ogc:def:featureType:OGC:1.0.1:Process:201"/>
   <om:observedProperty
xlink:href="urn:ogc:def:property:OGC:1.0.1:temperature"/>…
   <om:metadata>
  <swe:elementType name="QualityFlag"
href="http://sanyv1.iitb.fraunhofer.de/capext/Testbedgloss.htm#QualityFlags">
   <swe:field name="Not a Number">
    <swe:Text definition="Data matches a Not-A-Number value from the provider
(or error code from an instrument)"/>
    <swe:value>NaN</swe:value>
   </swe:field>
   <swe:field name="Null">
    <swe:Text definition="Data missing or null"/>
     <swe:value>NULL</swe:value>
   </swe:field>
   <swe:field name="out of engineering range">
    <swe:Text definition="Data exceeds range specified for instrument"/>
     <swe:value>OOER</swe:value>
   </swe:field>
   <swe:field name="out of range for datatype">
    <swe:Text definition="Data is out of valid range for this datatype"/>
     <swe:value>OODTR</swe:value>
   </swe:field>
   <swe:field name="out of calculation range">
    <swe:Text definition="Data exceeds range that its calculation allows"/>
     <swe:value>OOCR</swe:value>
   </swe:field>
  </swe:elementType>
 </om:metadata>
 <swe:DataArray>
    …
      <swe:elementType name="Interpolation Fields">
        <swe:SimpleDataRecord gml:id="Format-for-Interpolation-Fields">
         <swe:field name="latitude">…</swe:field>
         <swe:field name="longitude">…</swe:field>
         <swe:field name="temperature"
xlink:href="#xpointer(here()/../../../un:StatisticsArray[1]">
           <swe:Quantity
definition="urn:ogc:def:property:OGC:1.0.1:temperature">
            <swe:uom code="Cel"/>
           </swe:Quantity>
         </swe:field>
        </swe:SimpleDataRecord>
      </swe:elementType>
      <swe:encoding>
        <swe:TextBlock blockSeparator="@@" decimalSeparator="."
tokenSeparator=","/>
      </swe:encoding>
<swe:values>49.0146,8.42612,NaN@@49.0146,8.42614,13.8057@@49.0146,8.42616,NUL
L@@49.0146,8.42618,OOER@@49.0146,8.4262,13.806@@
  </swe:values>
 </swe:DataArray>
 </om:result>
</om:Observation>
```

### 6.2.4 IITB Fusion SOS GetFeatureOfInterest

Each fusion result (= O&M Observation uploaded to the Fusion SOS after successfull fusion execution) references a *SamplingSurface* which describes the shape of the fusion result set. For provenance purposes, this *SamplingSurface* is also uploaded to the *Fusion SOS* and can be retrieved with a GetFeatureOfInterest request. Note: In contrast to the IT-INNOV approach, not *gml:MultiPointCoverage*, but *gml:RectifiedGrid* is used to describe the spatial grid of the result coverage.

This operation supports one or more ID(s) for *SamplingSurface(s)* **or** a spatial filter in the form of a Bounding Box (ogc:BBOX element). If the spatial filter is used, all *SamplingSurfaces* whose Bounding-Box intersects with the requested BoundingBox (i.e. they have a common area) will be returned.

**GetFeatureOfInterest [request by ID]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:GetFeatureOfInterest version="1.0.0" service="SOS"
xmlns:sos="http://www.opengis.net/sos/1.0">

<sos:FeatureOfInterestId>urn:ogc:def:featureType:OGC:1.0:SamplingSurface:5798
</sos:FeatureOfInterestId>
</sos:GetFeatureOfInterest>
```

**GetFeatureOfInterest [request by spatial filter]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sos:GetFeatureOfInterest service="SOS" version="1.0.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:sos="http://www.opengis.net/sos/1.0"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sos/1.0
http://schemas.opengis.net/sos/1.0.0/sosAll.xsd">
 <sos:location>
  <ogc:BBOX>
   <ogc:PropertyName>gml:location</ogc:PropertyName>
   <gml:Envelope>
    <gml:lowerCorner>49.0150 8.426</gml:lowerCorner>
    <gml:upperCorner>49.0153 8.428</gml:upperCorner>
   </gml:Envelope>
  </ogc:BBOX>
 </sos:location>
</sos:GetFeatureOfInterest>
```

**GetFeatureOfInterest [response SamplingSurface describing shape of the result set]**

```
<?xml version="1.0" encoding="utf-8"?>
<sa:SamplingSurface xmlns:sa="http://www.opengis.net/sampling/1.0"
xmlns="http://www.iitb.fraunhofer.de/sos"
xmlns:gml="http://www.opengis.net/gml"
xmlns:om="http://www.opengis.net/om/1.0"
xmlns:sos="http://www.iitb.fraunhofer.de/sos"
```

```
xmlns:swe="http://www.opengis.net/swe/1.0.1"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
gml:id="urn_ogc_def_featureType_OGC_1.0_SamplingSurface_274"
xsi:schemaLocation="http://www.opengis.net/om/1.0
http://schemas.opengis.net/om/1.0.0/om.xsd
http://www.opengis.net/sampling/1.0
http://schemas.opengis.net/sampling/1.0.0/sampling.xsd ">
   <gml:name>Fraunhofer IITB Site</gml:name>
   <gml:srsName>urn:ogc:def:crs:EPSG:4326</gml:srsName>
   <gml:boundedBy>
      <gml:Envelope>
         <gml:lowerCorner>49.014604 8.425207</gml:lowerCorner>
         <gml:upperCorner>49.015893 8.427421</gml:upperCorner>
      </gml:Envelope>
   </gml:boundedBy>
   <sa:sampledFeature>
      <gml:RectifiedGridCoverage>
         <gml:rectifiedGridDomain>
            <gml:RectifiedGrid dimension="2">
               <gml:limits>
                  <gml:GridEnvelope>
                     <gml:low>0 0</gml:low>
                     <gml:high>50 100</gml:high>
                  </gml:GridEnvelope>
               </gml:limits>
               <gml:axisName>x</gml:axisName>
               <gml:axisName>y</gml:axisName>
               <gml:origin>
                  <gml:Point>
                     <gml:pos>49.0147 8.4253</gml:pos>
                  </gml:Point>
               </gml:origin>
               <gml:offsetVector>0 4e-005</gml:offsetVector>
               <gml:offsetVector>2e-005 0</gml:offsetVector>
            </gml:RectifiedGrid>
         </gml:rectifiedGridDomain>
         <gml:rangeSet>
            <gml:ValueArray/>
         </gml:rangeSet>
      </gml:RectifiedGridCoverage>
   </sa:sampledFeature>
   <sa:shape>
      <gml:Polygon>
         <gml:exterior>
            <gml:Ring>
               <gml:curveMember>
                  <gml:LineString>
                     <gml:pos>49.014604 8.425207</gml:pos>
                     <gml:pos>49.014604 8.427421</gml:pos>
                     <gml:pos>49.015893 8.427421</gml:pos>
                     <gml:pos>49.015893 8.425207</gml:pos>
                  </gml:LineString>
               </gml:curveMember>
            </gml:Ring>
         </gml:exterior>
      </gml:Polygon>
   </sa:shape>
</sa:SamplingSurface>
```

### 6.2.5 IITB SPS Fusion Process Description

The following is an example of a SensorML uploaded to the SPS (a so called "Asset"), which describes an available and executable fusion process (i.e. a interpolation algorithm). Most importantly, it describes the expected inputs/parameters, outputs, the algorithm itself and the implementation module of the process.

Please note that the input and parameter sections describe the expected data-types, but no concrete values yet. The values can be supplied later via a SPS *Submit* request to start the execution of the process with concrete input data and configuration parameters. After successful execution of the fusion process, the process SensorML description *with the supplied values*, will then be stored (along with the Fusion result and the SamplingSurface) for provenance purposes in the *Fusion SOS* (see section 6.2).

There is one *SPS Asset* (SensorML) per SPS fusion process.

The SPS *Assets* provide the core data which allow the Fusion SPS to handle/execute *GetCapabilities*, *DescribeTasking*, *GetFeasibility* and *Submit* requests.

The XML example below is a description of the BME algorithm used in the IITB Testbed *Fusion SPS.*

**SPS Asset [SensorML description of IITB BME algorithm]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sml:SensorML version="1.0.1"
xsi:schemaLocation="http://www.opengis.net/sensorML/1.0.1
http://schemas.opengis.net/sensorML/1.0.1/sensorML.xsd"
xmlns:sml="http://www.opengis.net/sensorML/1.0.1"
xmlns="http://www.opengis.net/sos/1.0" xmlns:gml="http://www.opengis.net/gml"
xmlns:om="http://www.opengis.net/om/1.0"
xmlns:swe="http://www.opengis.net/swe/1.0.1"
xmlns:tml="http://www.opengis.net/tml"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <sml:member>
  <sml:ProcessModel gml:id="urn_ogc_def_featureType_OGC_1.0.1_Process_963">
   <gml:methodName>BME interval</gml:methodName>
   <!--System Identifiers-->
   <!--====================================================-->
   <!--sml:identification element must contain ID of the Fusion process-->
   <sml:identification>
    <sml:IdentifierList>
     <sml:identifier name="uniqueID">
      <sml:Term definition="urn:ogc:def:identifier:OGC:uniqueID">
       <sml:value>Watson test 1</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="manufacturerName">
      <sml:Term
        definition="urn:ogc:def:identifier:OGC:1.0.1:manufacturerName">
       <sml:value>Fraunhofer IITB</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="longName">
      <sml:Term definition="urn:ogc:def:identifier:OGC:1.0.1:longname">
       <sml:value>IITB-F010 BME Fusion for interval data</sml:value>
      </sml:Term>
     </sml:identifier>
     <sml:identifier name="shortName">
      <sml:Term definition="urn:ogc:def:identifier:OGC:1.0.1:shortName">
       <sml:value>IITB BME Interval</sml:value>
      </sml:Term>
     </sml:identifier>
    </sml:IdentifierList>
   </sml:identification>
   <!--====================================================-->
   <!--====================================================-->
   <!--System Characteristics-->
   <!--====================================================-->
   <!--sml:characteristics element may be empty-->
   <sml:characteristics/>
   <!--====================================================-->
   <!--System capabilities-->
   <!--====================================================-->
   <!--sml:capabilities element may be empty-->
   <sml:capabilities/>
   <!--list containing the input and output phenomena, may be empty-->
   <sml:inputs>
    <sml:InputList>
     <sml:input name="ObservationData" xlink:href=""
xlink:role="urn:ogc:def:featureType:OGC:1.0:Observation" xlink:title=""/>
```

```
     <sml:input name="SamplingFeature" xlink:href=""
xlink:role="urn:ogc:def:featureType:OGC:1.0:SamplingSurface" xlink:title=""/>
    </sml:InputList>
   </sml:inputs>
   <!--output section;-->
   <sml:outputs>
    <sml:OutputList>
     <sml:output name="kriged_temperature">
      <!-- This is name of 'phenomenon' that appears in SPS capabilities -->
      <swe:DataArray>
       <swe:elementCount>
        <swe:Count>
         <!-- <swe:value>100</swe:value> -->
        </swe:Count>
       </swe:elementCount>
       <swe:elementType name="Interpolation Fields">
        <swe:DataRecord gml:id="Format-for-Interpolation-Fields_1">
         <swe:field name="latitude">
          <swe:Quantity
            definition="urn:ogc:property:location:EPSG:4326:latitude">
           <swe:uom code="deg"/>
          </swe:Quantity>
         </swe:field>
         <swe:field name="longitude">
          <swe:Quantity
            definition="urn:ogc:property:location:EPSG:4326:longitude">
           <swe:uom code="deg"/>
          </swe:Quantity>
         </swe:field>
         <swe:field name="temperature">
          <swe:Quantity
            definition="urn:ogc:def:property:OGC:1.0.1:temperature">
           <swe:uom code="Cel"/>
          </swe:Quantity>
         </swe:field>
        </swe:DataRecord>
       </swe:elementType>
      </swe:DataArray>
     </sml:output>
    </sml:OutputList>
   </sml:outputs>
   <!--parameter section;-->
   <sml:parameters>
    <sml:ParameterList>
     <sml:parameter name="samplingStart">
      <swe:Time>
       <gml:description>Start time of the sampled data</gml:description>
       <swe:value>unknown</swe:value>
       <!-- In iso 8601 format -->
      </swe:Time>
     </sml:parameter>
     <sml:parameter name="samplingEnd">
      <swe:Time>
       <gml:description>End time of the sampled data</gml:description>
       <swe:value>unknown</swe:value>
       <!-- In iso 8601 format -->
      </swe:Time>
     </sml:parameter>
     <sml:parameter name="observedProperty">
      <swe:Category>
```

```
        <swe:constraint>
         <swe:AllowedTokens>
          <swe:valueList>
            urn:ogc:def:property:OGC:1.0.1:temperature
          </swe:valueList>
          <swe:valueList>
            urn:ogc:def:property:OGC:1.0.1:relativeHumidity
          </swe:valueList>
          <swe:valueList>
            urn:ogc:def:phenomenon:SANY:2008:01:pressure
          </swe:valueList>
          <swe:valueList>
            urn:ogc:def:phenomenon:SANY:2008:01:acceleration:X
          </swe:valueList>
          <swe:valueList>
            urn:ogc:def:phenomenon:SANY:2008:01:acceleration:Y
          </swe:valueList>
          <swe:valueList>
            urn:ogc:def:phenomenon:SANY:2008:01:illuminance
          </swe:valueList>
         </swe:AllowedTokens>
        </swe:constraint>
        <swe:value/>
        <!-- One of the above listed -->
       </swe:Category>
      </sml:parameter>
      <sml:parameter name="covmodel">
       <swe:Text>
        <gml:description>name of covariance model, one of nuggetC, sphericalC,
exponentialC, gaussianC, holecosC, holesinC</gml:description>
        <swe:value/>
       </swe:Text>
      </sml:parameter>
      <sml:parameter name="covparam">
       <swe:Text>
        <gml:description>variance and range of covariance
model</gml:description>
        <swe:value/>
       </swe:Text>
      </sml:parameter>
      <sml:parameter name="nhmax">
       <swe:Quantity>
        <gml:description>max number of hard observations considered in a
neighborhood</gml:description>
        <swe:value>NaN</swe:value>
       </swe:Quantity>
      </sml:parameter>
      <sml:parameter name="nsmax">
       <swe:Quantity>
        <gml:description>max number of soft observations considered in a
neighborhood</gml:description>
        <swe:value>NaN</swe:value>
       </swe:Quantity>
      </sml:parameter>
      <sml:parameter name="dmax">
       <swe:Quantity>
        <gml:description>spatial diameter of a neighborhood around an
estimation location</gml:description>
        <swe:value>NaN</swe:value>
       </swe:Quantity>
```

```
      </sml:parameter>
      <sml:parameter name="order">
       <swe:Quantity>
        <gml:description>order polynomial of local
interpolation</gml:description>
        <swe:value>NaN</swe:value>
       </swe:Quantity>
      </sml:parameter>
      <sml:parameter name="options">
       <swe:Text>
        <gml:description>optional parameters for computation</gml:description>
        <swe:value/>
       </swe:Text>
      </sml:parameter>
     </sml:ParameterList>
    </sml:parameters>
    <!--method section;-->
    <sml:method>
     <sml:ProcessMethod>
      <!-=== Description ===-->
      <gml:description>This process methode uses an Fusion algorithm to
generate Fusion observation data.</gml:description>
      <!-=== Contact ===-->
      <sml:contact xlink:arcrole="urn:ogc:def:property:OGC:1.0:author"
xlink:href="IITB-SiegbertKunz-KymWatson"/>
      <!-=== Rules Set ===-->
      <sml:rules>
       <sml:RulesDefinition>
        <gml:description>The input must be an observation collection. The
output will be a coverage.</gml:description>
        <!--If a profile would be defined for describing the in- and outputs
and parameters expected / required by the ProcessMethod it could be
referenced here.-->
       </sml:RulesDefinition>
      </sml:rules>
      <!-=== Algorithm ===-->
      <sml:algorithm>
       <sml:AlgorithmDefinition>
        <gml:description>The Fusion algorithm is based on the Bayesian maximum
entropy method for interval PDFs</gml:description>
       </sml:AlgorithmDefinition>
      </sml:algorithm>
      <sml:implementation>
       <sml:ImplementationCode language="JAVA" version="1.5">
        <sml:binaryRef xlink:href="fusion.sps.process.BMEFusionTask"/>
       </sml:ImplementationCode>
      </sml:implementation>
     </sml:ProcessMethod>
    </sml:method>
   </sml:ProcessModel>
  </sml:member>
</sml:SensorML>
```

### 6.2.6 IITB SPS GetCapabilities

**GetCapabilities [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<GetCapabilities xmlns="http://www.opengis.net/sps/1.0" service="SPS"/>
```

**GetCapabilities [response]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sps:Capabilities version="1.0.0"
xsi:schemaLocation="http://www.opengis.net/sps/1.0
http://schemas.opengis.net/sps/1.0.0/spsAll.xsd"
xmlns:sps="http://www.opengis.net/sps/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:ows="http://www.opengis.net/ows">
 <ows:ServiceIdentification xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:om="http://www.opengis.net/om/1.0">
  <ows:Title>SPS interface for FUSION-SOS 1.0.0 Fraunhofer IITB</ows:Title>
  <ows:Abstract>Sensor Planning Service interface for configuration of the
FUSION SOS 1.0.0 at Fraunhofer IITB</ows:Abstract>
  <ows:Keywords>
   <ows:Keyword>fusion, sos, sps, fusionsos, webgenesis</ows:Keyword>
  </ows:Keywords>
  <ows:ServiceType
codeSpace="http://opengeospatial.net">OGC:SPS</ows:ServiceType>
  <ows:ServiceTypeVersion>1.0.0</ows:ServiceTypeVersion>
  <ows:Fees>NONE</ows:Fees>
  <ows:AccessConstraints>NONE</ows:AccessConstraints>
 </ows:ServiceIdentification>
 <ows:ServiceProvider xmlns:swe="http://www.opengis.net/swe/1.0"
xmlns:ogc="http://www.opengis.net/ogc"
xmlns:om="http://www.opengis.net/om/1.0">
  <ows:ProviderName>Fraunhofer Institute for Information Technology and Data
Processing</ows:ProviderName>
  <ows:ProviderSite xlink:href="http://www.iitb.fraunhofer.de"/>
  <ows:ServiceContact>
   <ows:IndividualName>Dr. Kym Watson</ows:IndividualName>
   <ows:PositionName>Research Associate</ows:PositionName>
   <ows:ContactInfo>
    <ows:Phone>
     <ows:Voice>+49-721-6091</ows:Voice>
    </ows:Phone>
    <ows:Address>
     <ows:DeliveryPoint>Fraunhoferstr. 1</ows:DeliveryPoint>
     <ows:City>Karlsruhe</ows:City>
     <ows:AdministrativeArea>BW</ows:AdministrativeArea>
     <ows:PostalCode>76131</ows:PostalCode>
     <ows:Country>Germany</ows:Country>
     <ows:ElectronicMailAddress>
       kym.watson@iitb.fraunhofer.de
     </ows:ElectronicMailAddress>
    </ows:Address>
   </ows:ContactInfo>
   <ows:Role/>
  </ows:ServiceContact>
 </ows:ServiceProvider>
 <ows:OperationsMetadata>
  <ows:Operation name="GetCapabilities">
   <ows:DCP>
    <ows:HTTP>
     <ows:Get xlink:href="http://imtfusos/servlet/is/Entry..SPSRequest/?"/>
     <ows:Post xlink:href="http://imtfusos/servlet/is/Entry..SPSRequest/"/>
    </ows:HTTP>
```

```
  </ows:DCP>
  <ows:Parameter name="service">
   <ows:Value>SPS</ows:Value>
  </ows:Parameter>
  <ows:Parameter name="AcceptVersions">
   <ows:Value>1.0.0</ows:Value>
  </ows:Parameter>
  <ows:Parameter name="Sections">
   <ows:Value>All</ows:Value>
   <ows:Value>ServiceIdentification</ows:Value>
   <ows:Value>ServiceProvider</ows:Value>
   <ows:Value>OperationsMetadata</ows:Value>
   <ows:Value>Contents</ows:Value>
  </ows:Parameter>
 </ows:Operation>
 <ows:Operation name="DescribeTasking">
  <ows:DCP>
   <ows:HTTP>
    <ows:Post xlink:href="http://imtfusos/servlet/is/Entry..SPSRequest/"/>
   </ows:HTTP>
  </ows:DCP>
  <ows:Parameter name="service">
   <ows:Value>SPS</ows:Value>
  </ows:Parameter>
  <ows:Parameter name="version">
   <ows:Value>1.0.0</ows:Value>
  </ows:Parameter>
  <ows:Parameter name="sensorID">
   <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:963</ows:Value>
  </ows:Parameter>
 </ows:Operation>
 <ows:Operation name="Submit">
  <ows:DCP>
   <ows:HTTP>
    <ows:Post xlink:href="http://imtfusos/servlet/is/Entry..SPSRequest/"/>
   </ows:HTTP>
  </ows:DCP>
  <ows:Parameter name="service">
   <ows:Value>SPS</ows:Value>
  </ows:Parameter>
  <ows:Parameter name="version">
   <ows:Value>1.0.0</ows:Value>
  </ows:Parameter>
  <ows:Parameter name="notificationTarget">
   <ows:Value/>
  </ows:Parameter>
  <ows:Parameter name="sensorParam">
   <ows:Value/>
  </ows:Parameter>
  <ows:Parameter name="timeFrame">
   <ows:Value/>
  </ows:Parameter>
  <ows:Parameter name="feasibilityID">
   <ows:Value/>
  </ows:Parameter>
 </ows:Operation>
 <ows:Operation name="DescribeResultAccess">
  <ows:DCP>
   <ows:HTTP>
    <ows:Post xlink:href="http://imtfusos/servlet/is/Entry..SPSRequest/"/>
```

```
    </ows:HTTP>
   </ows:DCP>
   <ows:Parameter name="service">
    <ows:Value>SPS</ows:Value>
   </ows:Parameter>
   <ows:Parameter name="version">
    <ows:Value>1.0.0</ows:Value>
   </ows:Parameter>
   <ows:Parameter name="taskID">
    <ows:Value/>
   </ows:Parameter>
  </ows:Operation>
  <ows:Operation name="GetFeasibility">
   <ows:DCP>
    <ows:HTTP>
     <ows:Post xlink:href="http://imtfusos/servlet/is/Entry..SPSRequest/"/>
    </ows:HTTP>
   </ows:DCP>
   <ows:Parameter name="service">
    <ows:Value>SPS</ows:Value>
   </ows:Parameter>
   <ows:Parameter name="version">
    <ows:Value>1.0.0</ows:Value>
   </ows:Parameter>
   <ows:Parameter name="notificationTarget">
    <ows:Value/>
   </ows:Parameter>
   <ows:Parameter name="parameters">
    <ows:Value/>
   </ows:Parameter>
   <ows:Parameter name="timeFrame">
    <ows:Value/>
   </ows:Parameter>
   <ows:Parameter name="sensorID">
    <ows:Value>urn:ogc:def:featureType:OGC:1.0.1:Process:963</ows:Value>
   </ows:Parameter>
  </ows:Operation>
  <ows:Operation name="GetStatus">
   <ows:DCP>
    <ows:HTTP>
     <ows:Post xlink:href="http://imtfusos/servlet/is/Entry..SPSRequest/"/>
    </ows:HTTP>
   </ows:DCP>
   <ows:Parameter name="service">
    <ows:Value>SPS</ows:Value>
   </ows:Parameter>
   <ows:Parameter name="version">
    <ows:Value>1.0.0</ows:Value>
   </ows:Parameter>
   <ows:Parameter name="taskID">
    <ows:Value/>
   </ows:Parameter>
  </ows:Operation>
  <ows:Operation name="Cancel">
   <ows:DCP>
    <ows:HTTP>
     <ows:Post xlink:href="http://imtfusos/servlet/is/Entry..SPSRequest/"/>
    </ows:HTTP>
   </ows:DCP>
   <ows:Parameter name="service">
```

```
    <ows:Value>SPS</ows:Value>
   </ows:Parameter>
   <ows:Parameter name="version">
    <ows:Value>1.0.0</ows:Value>
   </ows:Parameter>
   <ows:Parameter name="taskID">
    <ows:Value/>
   </ows:Parameter>
  </ows:Operation>
 </ows:OperationsMetadata>
 <sps:Contents>
  <sps:SensorOfferingList>
   <sps:SensorOffering>
    <sps:AreaOfService>
     <ows:BoundingBox crs="urn:ogc:def:crs:EPSG:4326">
      <ows:LowerCorner>49.014604 8.425207</ows:LowerCorner>
      <ows:UpperCorner>49.015893 8.427421</ows:UpperCorner>
     </ows:BoundingBox>
    </sps:AreaOfService>
    <sps:Phenomenon>kriged_temperature</sps:Phenomenon>

<sps:SensorDefinition>http://IMTFUSOS.pc.iitb.fhg.de/servlet/is/963/?command=
downloadContent&amp;filename=FusionProcess_963.xml</sps:SensorDefinition>

<sps:SensorID>urn:ogc:def:featureType:OGC:1.0.1:Process:963</sps:SensorID>
   </sps:SensorOffering>
  </sps:SensorOfferingList>
  <sps:PhenomenonOfferingList>
   <sps:PhenomenonOffering>
    <sps:Phenomenon>kriged_temperature</sps:Phenomenon>

<sps:SensorID>urn:ogc:def:featureType:OGC:1.0.1:Process:963</sps:SensorID>
   </sps:PhenomenonOffering>
  </sps:PhenomenonOfferingList>
 </sps:Contents>
</sps:Capabilities>
```

### 6.2.7 IITB SPS DescribeTasking

This operation serves to describe the "taskable" parameters and inputs of a given SPS fusion process (the so called SPS *Asset*, described in SensorML, see also section 6.2).

**DescribeTasking [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<DescribeTasking xmlns="http://www.opengis.net/sps/1.0" service="SPS"
version="1.0.0">
 <sensorID>urn:ogc:def:featureType:OGC:1.0.1:Process:963</sensorID>
</DescribeTasking>
```

**DescribeTasking [response]**

For each requested Fusion process <*sps:sensorID*> there are 1..n *InputDescriptors* that describe the expected inputs for the Fusion process as swe data-types (e.g. swe:Time, swe:Quantity).

A client is expected to parse these *InputDescriptors* so and supply actual values for it in preparation for a SPS *Submit* request (see section 6.2.9)

```
<?xml version="1.0" encoding="UTF-8"?>
<sps:DescribeTaskingRequestResponse
xsi:schemaLocation="http://www.opengis.net/sps/1.0
http://schemas.opengis.net/sps/1.0.0/spsDescribeTaskingRequestResponse.xsd"
xmlns:sps="http://www.opengis.net/sps/1.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
 <sps:taskingDescriptor>
  <sps:sensorID>urn:ogc:def:featureType:OGC:1.0.1:Process:963</sps:sensorID>
  <sps:InputDescriptor parameterID="samplingStart" updateable="false"
use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Time xmlns:swe="http://www.opengis.net/swe/1.0">
      <gml:description xmlns:gml="http://www.opengis.net/gml">Start time of
the sampled data</gml:description>
      <swe:value>unknown</swe:value>
      <!--In iso 8601 format-->
     </swe:Time>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="samplingEnd" updateable="false"
use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Time xmlns:swe="http://www.opengis.net/swe/1.0">
      <gml:description xmlns:gml="http://www.opengis.net/gml">End time of the
sampled data</gml:description>
      <swe:value>unknown</swe:value>
      <!--In iso 8601 format-->
     </swe:Time>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="observedProperty" updateable="false"
use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Category xmlns:swe="http://www.opengis.net/swe/1.0">
      <swe:constraint>
       <swe:AllowedTokens>
        <swe:valueList>
          urn:ogc:def:property:OGC:1.0.1:temperature
        </swe:valueList>
        <swe:valueList>
          urn:ogc:def:property:OGC:1.0.1:relativeHumidity
        </swe:valueList>
        <swe:valueList>
          urn:ogc:def:phenomenon:SANY:2008:01:pressure
        </swe:valueList>
        <swe:valueList>
          urn:ogc:def:phenomenon:SANY:2008:01:acceleration:X
        </swe:valueList>
        <swe:valueList>
          urn:ogc:def:phenomenon:SANY:2008:01:acceleration:Y
        </swe:valueList>
        <swe:valueList>
          urn:ogc:def:phenomenon:SANY:2008:01:illuminance
        </swe:valueList>
       </swe:AllowedTokens>
```

```
      </swe:constraint>
      <swe:value/>
      <!--One of the above listed-->
     </swe:Category>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="covmodel" updateable="false"
use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Text xmlns:swe="http://www.opengis.net/swe/1.0">
      <gml:description xmlns:gml="http://www.opengis.net/gml">name of
covariance model, one of nuggetC, sphericalC, exponentialC, gaussianC,
holecosC, holesinC</gml:description>
      <swe:value/>
     </swe:Text>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="covparam" updateable="false"
use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Text xmlns:swe="http://www.opengis.net/swe/1.0">
      <gml:description xmlns:gml="http://www.opengis.net/gml">variance and
range of covariance model</gml:description>
      <swe:value/>
     </swe:Text>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="nhmax" updateable="false" use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Quantity xmlns:swe="http://www.opengis.net/swe/1.0">
      <gml:description xmlns:gml="http://www.opengis.net/gml">max number of
hard observations considered in a neighborhood</gml:description>
      <swe:value>NaN</swe:value>
     </swe:Quantity>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="nsmax" updateable="false" use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Quantity xmlns:swe="http://www.opengis.net/swe/1.0">
      <gml:description xmlns:gml="http://www.opengis.net/gml">max number of
soft observations considered in a neighborhood</gml:description>
      <swe:value>NaN</swe:value>
     </swe:Quantity>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="dmax" updateable="false" use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Quantity xmlns:swe="http://www.opengis.net/swe/1.0">
      <gml:description xmlns:gml="http://www.opengis.net/gml">spatial
diameter of a neighborhood around an estimation location</gml:description>
```

```
      <swe:value>NaN</swe:value>
     </swe:Quantity>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="order" updateable="false" use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Quantity xmlns:swe="http://www.opengis.net/swe/1.0">
      <gml:description xmlns:gml="http://www.opengis.net/gml">order
polynomial of local interpolation</gml:description>
      <swe:value>NaN</swe:value>
     </swe:Quantity>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="options" updateable="false"
use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Text xmlns:swe="http://www.opengis.net/swe/1.0">
      <gml:description xmlns:gml="http://www.opengis.net/gml">optional
parameters for computation</gml:description>
      <swe:value/>
     </swe:Text>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="ObservationData" updateable="false"
use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Text xmlns:swe="http://www.opengis.net/swe/1.0">
      <swe:value/>
     </swe:Text>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
  <sps:InputDescriptor parameterID="SamplingFeature" updateable="false"
use="required">
   <sps:definition>
    <sps:commonData>
     <swe:Text xmlns:swe="http://www.opengis.net/swe/1.0">
      <swe:value/>
     </swe:Text>
    </sps:commonData>
   </sps:definition>
  </sps:InputDescriptor>
 </sps:taskingDescriptor>
</sps:DescribeTaskingRequestResponse>
```

### 6.2.8 IITB SPS GetFeasibility

This is almost equivalent to the Submit request (see section 6.2.9), only that no fusion execution is started, but rather the feasibility of the fusion execution with the given input/parameters is checked. Please note that even if a *GetFeasibility* request responds with "*feasible*", it is still

possible that at actual execution time, the fusion process fails. (e.g. because a runtime error occurred or between the *GetFeasibility* request and the *Submit* request which started the Fusion, some time has passed which changed some conditions). Nevertheless, it is highly recommended to always issue a *GetFeasibility* request to check necessary preconditions, before issuing a *Submit* request that actually starts the Fusion.

**GetFeasibility [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<GetFeasibility xmlns="http://www.opengis.net/sps/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sps/1.0
http://schemas.opengis.net/sps/1.0.0/spsGetFeasibilityRequest.xsd"
xmlns:swe="http://www.opengis.net/swe/1.0" service="SPS" version="1.0.0">
<notificationTarget>
  <notificationID>89</notificationID>
  <notificationURL>http://mail2.spacebel.be:8080/52nWNS/wns</notificationURL>
 </notificationTarget>
 <sensorID>urn:ogc:def:featureType:OGC:1.0.1:Process:963</sensorID>
 <!-- <Alias>ifgicam01</Alias> -->
 <parameters>
     <InputParameter parameterID="samplingStart">
    <value>
     <swe:Time>
      <swe:value>2008-11-06T13:00:00+01:00</swe:value>
     </swe:Time>
    </value>
   </InputParameter>
       <InputParameter parameterID="samplingEnd">
    <value>
     <swe:Time>
      <gml:description>End time of the sampled data</gml:description>
      <swe:value>2008-11-06T14:00:00+01:00</swe:value>
     </swe:Time>
    </value>
   </InputParameter>
      <InputParameter parameterID="observedProperty">
    <value>
     <swe:Text>
      <swe:value>urn:ogc:def:property:OGC:1.0.1:temperature</swe:value>
     </swe:Text>
    </value>
   </InputParameter>
  <InputParameter parameterID="covmodel">
    <value>
     <swe:Text>
      <gml:description>name of covariance model, one of nuggetC, sphericalC,
exponentialC, gaussianC, holecosC, holesinC</gml:description>
      <swe:value>exponentialC</swe:value>
     </swe:Text>
    </value>
   </InputParameter>
   <InputParameter parameterID="covparam">
    <value>
     <swe:Text>
      <gml:description>variance and range of covariance
model</gml:description>
      <swe:value>20 0.0001</swe:value>
     </swe:Text>
    </value>
   </InputParameter>
   <InputParameter parameterID="nhmax">
    <value>
     <swe:Quantity>
```

```
      <gml:description>max number of hard observations considered in a
neighborhood</gml:description>
      <swe:value>10</swe:value>
     </swe:Quantity>
    </value>
   </InputParameter>
   <InputParameter parameterID="nsmax">
    <value>
     <swe:Quantity>
      <gml:description>max number of soft observations considered in a
neighborhood</gml:description>
      <swe:value>5</swe:value>
     </swe:Quantity>
    </value>
   </InputParameter>
   <InputParameter parameterID="dmax">
    <value>
     <swe:Quantity>
      <gml:description>spatial diameter of a neighborhood around an
estimation location</gml:description>
      <swe:value>0.005</swe:value>
     </swe:Quantity>
    </value>
   </InputParameter>
   <InputParameter parameterID="order">
    <value>
     <swe:Quantity>
      <gml:description>order polynomial of local
interpolation</gml:description>
      <swe:value>0</swe:value>
     </swe:Quantity>
    </value>
   </InputParameter>
   <InputParameter parameterID="options">
    <value>
     <swe:Text>
      <gml:description>optional parameters for computation</gml:description>
      <swe:value>0 0 50000 1e-4</swe:value>
     </swe:Text>
    </value>
   </InputParameter>
   <InputParameter parameterID="ObservationData">
    <value>
     <swe:Text>
      <gml:description>Input Observation data for the Fusion
process</gml:description>
      <swe:value>http://imtfusos.pc.iitb.fhg.de/servlet/is/2165/assembled-
observation.xml?command=downloadContent&amp;filename=assembled-
observation.xml</swe:value>
     </swe:Text>
    </value>
   </InputParameter>
   <InputParameter parameterID="SamplingFeature">
    <value>
     <swe:Text>
      <gml:description>SamplingSurface of the sampled
observations</gml:description>

<swe:value>http://imtfusos.pc.iitb.fhg.de/servlet/is/2165/SamplingFeature_217
.xml?command=downloadContent&amp;filename=SamplingFeature_217.xml</swe:value>
```

```
      </swe:Text>
     </value>
    </InputParameter>
  </parameters>
   <timeFrame>
   <gml:TimeInstant>
    <gml:timePosition>2009-10-05T12:00:00Z</gml:timePosition>
   </gml:TimeInstant>
  </timeFrame>
</GetFeasibility>
```

**GetFeasibility [response]**

The response returns information whether a fusion process is feasible with the given parameter/input values and also provides a feasibilityID for later reference (see also section 6.2.9).

```
<?xml version="1.0" encoding="UTF-8"?>
<sps:GetFeasibilityRequestResponse
xmlns:sps="http://www.opengis.net/sps/1.0">
 <sps:feasibilityID>e2ecf1bb-0fe6-494a-a38e-8ef9d17d7a44</sps:feasibilityID>
 <sps:feasibility>feasible</sps:feasibility>
 <gml:description xmlns:gml="http://www.opengis.net/gml">BMEFusion process is
feasible with supplied parameters/inputs</gml:description>
</sps:GetFeasibilityRequestResponse>
```

### 6.2.9 IITB SPS Submit

This starts execution of a fusion process with the given inputs and parameters. There are to principal request types:

- With a feasibilityID from a previously executed *GetFeasibility* request that responded with *"feasible"*. In this case, the inputs and parameters supplied in that *GetFeasibility* request will be reused.

- With the actual inputs and parameters supplied (compare also section 6.2.8)

**Submit [request with feasibilityID]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sps:GetFeasibilityRequestResponse
xmlns:sps="http://www.opengis.net/sps/1.0">
 <sps:feasibilityID>e2ecf1bb-0fe6-494a-a38e-8ef9d17d7a44</sps:feasibilityID>
 <sps:feasibility>feasible</sps:feasibility>
 <gml:description xmlns:gml="http://www.opengis.net/gml">BMEFusion process is
feasible with supplied parameters/inputs</gml:description>
</sps:GetFeasibilityRequestResponse>
```

**Submit [request with inputs and parameters inline]**

```
<?xml version="1.0" encoding="UTF-8"?>
<Submit xmlns="http://www.opengis.net/sps/1.0"
xmlns:gml="http://www.opengis.net/gml"
xmlns:swe="http://www.opengis.net/swe/1.0" service="SPS" version="1.0.0"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sps/1.0
http://schemas.opengis.net/sps/1.0.0/spsSubmitRequest.xsd">
 <notificationTarget>
  <notificationID>89</notificationID>
  <notificationURL>http://mail2.spacebel.be:8080/52nWNS/wns</notificationURL>
 </notificationTarget>
 <sensorParam>
  <sensorID>urn:ogc:def:featureType:OGC:1.0.1:Process:963</sensorID>
  <parameters>
   <!-- REQUIRED PARAMETERS -->
      <InputParameter parameterID="samplingStart">
    <value>
     <swe:Time>
      <swe:value>2006-11-06T12:30:00+02:00</swe:value>
     </swe:Time>
    </value>
   </InputParameter>
       <InputParameter parameterID="samplingEnd">
    <value>
     <swe:Time>
      <gml:description>End time of the sampled data</gml:description>
      <swe:value>2006-11-13T12:00:00+02:00</swe:value>
     </swe:Time>
    </value>
   </InputParameter>
      <InputParameter parameterID="observedProperty">
    <value>
     <swe:Text>
      <swe:value>urn:ogc:def:property:OGC:1.0.1:temperature</swe:value>
     </swe:Text>
    </value>
   </InputParameter>
   <InputParameter parameterID="covmodel">
    <value>
     <swe:Text>
      <gml:description>name of covariance model, one of nuggetC, sphericalC,
exponentialC, gaussianC, holecosC, holesinC</gml:description>
      <swe:value>exponentialC</swe:value>
     </swe:Text>
    </value>
   </InputParameter>
   <InputParameter parameterID="covparam">
    <value>
     <swe:Text>
      <gml:description>variance and range of covariance
model</gml:description>
      <swe:value>20 0.0001</swe:value>
     </swe:Text>
    </value>
   </InputParameter>
   <InputParameter parameterID="nhmax">
    <value>
     <swe:Quantity>
```

```
      <gml:description>max number of hard observations considered in a
neighborhood</gml:description>
      <swe:value>10</swe:value>
     </swe:Quantity>
    </value>
   </InputParameter>
   <InputParameter parameterID="nsmax">
    <value>
     <swe:Quantity>
      <gml:description>max number of soft observations considered in a
neighborhood</gml:description>
      <swe:value>5</swe:value>
     </swe:Quantity>
    </value>
   </InputParameter>
   <InputParameter parameterID="dmax">
    <value>
     <swe:Quantity>
      <gml:description>spatial diameter of a neighborhood around an
estimation location</gml:description>
      <swe:value>0.005</swe:value>
     </swe:Quantity>
    </value>
   </InputParameter>
   <InputParameter parameterID="order">
    <value>
     <swe:Quantity>
      <gml:description>order polynomial of local
interpolation</gml:description>
      <swe:value>0</swe:value>
     </swe:Quantity>
    </value>
   </InputParameter>
   <InputParameter parameterID="options">
    <value>
     <swe:Text>
      <gml:description>optional parameters for computation</gml:description>
      <swe:value>0 0 50000 1e-4</swe:value>
     </swe:Text>
    </value>
   </InputParameter>
   <InputParameter parameterID="ObservationData">
    <value>
     <swe:Text>
      <gml:description>Input Observation data for the Fusion
process</gml:description>
      <swe:value>http://imtfusos.pc.iitb.fhg.de/servlet/is/2165/assembled-
observation.xml?command=downloadContent&amp;filename=assembled-
observation.xml</swe:value>
     </swe:Text>
    </value>
   </InputParameter>
   <InputParameter parameterID="SamplingFeature">
    <value>
     <swe:Text>
      <gml:description>SamplingSurface of the sampled
observations</gml:description>

<swe:value>http://imtfusos.pc.iitb.fhg.de/servlet/is/2165/SamplingFeature_217
.xml?command=downloadContent&amp;filename=SamplingFeature_217.xml</swe:value>
```

```
      </swe:Text>
     </value>
    </InputParameter>
   </parameters>
  </sensorParam>
  <timeFrame>
   <gml:TimeInstant>
    <gml:timePosition>2009-10-05T12:00:00Z</gml:timePosition>
   </gml:TimeInstant>
  </timeFrame>
</Submit>
```

**Submit [response]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sps:SubmitRequestResponse xmlns:sps="http://www.opengis.net/sps/1.0">
 <sps:taskID>5801</sps:taskID>
 <sps:status>confirmed</sps:status>
 <gml:description xmlns:gml="http://www.opengis.net/gml">This is a test
task</gml:description>
 <sps:LatestResponseTime>
  <gml:TimeInstant xmlns:gml="http://www.opengis.net/gml">
   <gml:timePosition>2009-05-28T17:02:24+02:00</gml:timePosition>
  </gml:TimeInstant>
 </sps:LatestResponseTime>
</sps:SubmitRequestResponse>
```

It is noteworthy that the Submit response is returned directly (synchronously) while the Fusion process (that may very well take some time to execute) is asynchronously executed in the background. This allows to execute several time-consuming fusion-tasks in parallel, without one task blocking all others.

To handle this asynchronous nature of SPS Fusion execution, a Submit request MUST include notification data, so the client/user can be notified of changes in the Fusion task (e.g. when it has finished, cancelled, etc).

Specifically, the Fusion SPS currently implements notification via a WNS server, so the notification-target parameter in the Submit-request must contain the URL to a WNS server and a valid (registered) notificationID (tested with email notification via 52°North WNS at SPB, see example above)

### 6.2.10 IITB SPS GetStatus

After a fusion-process task has been submitted and confirmed, status information about the current that task can be requested at any time.

**GetStatus [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sps:GetStatus xmlns:sps="http://www.opengis.net/sps/1.0" service="SPS"
version="1.0.0">
<sps:taskID>5801</sps:taskID>
</sps:GetStatus>
```

**GetStatus [response]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sps:GetStatusRequestResponse xmlns:sps="http://www.opengis.net/sps/1.0">
 <sps:taskID>5801</sps:taskID>
 <sps:status>finished</sps:status>
 <gml:description xmlns:gml="http://www.opengis.net/gml">This is a test
task</gml:description>
</sps:GetStatusRequestResponse>
```

### 6.2.11 IITB SPS DescribeResultAccess

After a fusion process has finished execution, a client/user is a alerted to that fact through a notification (e.g. via WNS/email). A DescribeResultAccess can then be issued for the finished task to get a prepared GetObservationById request for retrieval of the result data (see also section 6.2.3).

**DescribeResultAccess [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sps:DescribeResultAccess xmlns:sps="http://www.opengis.net/sps/1.0"
service="SPS" version="1.0.0">
 <sps:taskID>5801</sps:taskID>
</sps:DescribeResultAccess>
```

**DescribeResultAccess [response]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sps:DescribeResultAccessRequestResponse
xmlns:sps="http://www.opengis.net/sps/1.0">
 <sps:service>
  <sps:ServiceType>SOS</sps:ServiceType>

<sps:ServiceURL>http://imtfusos.pc.iitb.fhg.de/servlet/is/Entry..SOSRequest/<
/sps:ServiceURL>
  <sps:request>
   <sos:GetObservationById version="1.0.0" service="SOS"
xmlns:sos="http://www.opengis.net/sos/1.0">

<sos:ObservationId>urn:ogc:def:featureType:OGC:1.0:Observation:5804</sos:Obse
rvationId>
    <sos:responseFormat>text/xml;subtype="om/1.0.0"</sos:responseFormat>
   </sos:GetObservationById>
  </sps:request>
 </sps:service>
</sps:DescribeResultAccessRequestResponse>
```

### 6.2.12 IITB SPS Cancel

At any time during execution of a fusion-process (task), that task can be interrupted by the user/client with a Cancel request. The server will attempt to stop the executing thread as soon as possible (though no guarantee concerning when it will actually be able to stop the execution is given. For example, if issued shortly before the task is finished, it might finish anyway).

**Cancel [request]**

```
<?xml version="1.0" encoding="UTF-8"?>
<Cancel xmlns="http://www.opengis.net/sps/1.0" service="SPS" version="1.0.0">
<taskID>5564</taskID>
</Cancel>
```

**Cancel [response]**

```
<?xml version="1.0" encoding="UTF-8"?>
<sps:CancelRequestResponse xmlns:sps="http://www.opengis.net/sps/1.0">
 <sps:taskID>5564</sps:taskID>
 <sps:status>confirmed</sps:status>
 <gml:description xmlns:gml="http://www.opengis.net/gml">Task 5564 has been
successfully cancelled</gml:description>
</sps:CancelRequestResponse>
```

Note: It might not be possible to cancel a running task for certain reasons, in which case the *Cancel* response status will be "*rejected"*, along with a description of the cause.

## 6.3. BMT I/O specification

As the services produced by BMT follow the same OGC standards as the services produced by IT-INNOV and IITB, and in the interests of preventing repetition, only the functional differences between the BMT and other services are covered here.

### 6.3.1 BMT WPS GetCapabilities

The GetCapabilities request and response documents are functionally identical to those of the IT-INNOV WPS (see section 6.1).

### 6.3.2 BMT WPS DescribeProcess

The inputs and outputs of the two fusion processes offered in the BMT WPS are identical. Only the method used for the model differs. As such, the response to a DescribeProcess request will look like the following:

```
<?xml version="1.0" encoding="UTF-8"?>
<wps:ProcessDescriptions xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" service="WPS"
version="1.0.0" xml:lang="en-GB"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://213.235.6.202:8081/WPS/SCHEMA/wps/1.0.0/wpsDescribeProcess_response.xs
d http://www.opengis.net/ows/1.1
http://213.235.6.202:8081/WPS/SCHEMA/ows/1.1.0/owsAll.xsd">
 <ProcessDescription wps:processVersion="1.0.0" storeSupported="false"
statusSupported="false">
  <ows:Identifier>BWQMv1</ows:Identifier>
  <ows:Title>Bathing Water Quality Model v1</ows:Title>
  <ows:Abstract>Bathing Water Quality Model v1 developed by BMT Cordah Ltd
for use in the SANY EU research project.</ows:Abstract>
  <ows:Metadata xlink:title="SensorML">discovery_sensorML</ows:Metadata>
  <DataInputs xmlns:wps="http://www.opengis.net/wps/1.0"
xmlns:ows="http://www.opengis.net/ows/1.1">
   <Input minOccurs="1" maxOccurs="1">
    <ows:Identifier>bathing_area_id</ows:Identifier>
    <ows:Title>Bathing area ID</ows:Title>
    <LiteralData>
     <ows:DataType ows:reference="http://www.w3.org/TX/smlchema-2/#string">
       string
     </ows:DataType>
     <ows:AnyValue />
    </LiteralData>
   </Input>
   <Input minOccurs="1" maxOccurs="1">
    <ows:Identifier>bathing_area_wfs_url</ows:Identifier>
    <ows:Title>URL to retrieve bathing area from</ows:Title>
    <LiteralData>
     <ows:DataType ows:reference="http://www.w3.org/TX/smlchema-2/#string">
       string
     </ows:DataType>
     <ows:AnyValue />
    </LiteralData>
   </Input>
   <Input minOccurs="1" maxOccurs="1">
    <ows:Identifier>wind_source</ows:Identifier>
    <ows:Title>Source to use for wind data</ows:Title>
    <ComplexData>
     <Default>
      <Format>
       <MimeType>text/xml</MimeType>
       <Encoding>UTF-8</Encoding>
       <Schema>
         http://213.235.6.202:3666/BenbowWFS/SCHEMA/Sany/DataSource.xsd
       </Schema>
      </Format>
     </Default>
     <Supported>
      <Format>
       <MimeType>text/xml</MimeType>
       <Encoding>UTF-8</Encoding>
       <Schema>
         http://213.235.6.202:3666/BenbowWFS/SCHEMA/Sany/DataSource.xsd
       </Schema>
      </Format>
```

```
    </Supported>
   </ComplexData>
  </Input>
  <Input minOccurs="1" maxOccurs="1">
   <ows:Identifier>rain_source</ows:Identifier>
   <ows:Title>Source to use for rain data</ows:Title>
   <ComplexData>
    <Default>
     <Format>
      <MimeType>text/xml</MimeType>
      <Encoding>UTF-8</Encoding>
      <Schema>
        http://213.235.6.202:3666/BenbowWFS/SCHEMA/Sany/DataSource.xsd
      </Schema>
     </Format>
    </Default>
    <Supported>
     <Format>
      <MimeType>text/xml</MimeType>
      <Encoding>UTF-8</Encoding>
      <Schema>
        http://213.235.6.202:3666/BenbowWFS/SCHEMA/Sany/DataSource.xsd
      </Schema>
     </Format>
    </Supported>
   </ComplexData>
  </Input>
  <Input minOccurs="1" maxOccurs="1">
   <ows:Identifier>cloud_source</ows:Identifier>
   <ows:Title>Source to use for cloud data</ows:Title>
   <ComplexData>
    <Default>
     <Format>
      <MimeType>text/xml</MimeType>
      <Encoding>UTF-8</Encoding>
      <Schema>
        http://213.235.6.202:3666/BenbowWFS/SCHEMA/Sany/DataSource.xsd
      </Schema>
     </Format>
    </Default>
    <Supported>
     <Format>
      <MimeType>text/xml</MimeType>
      <Encoding>UTF-8</Encoding>
      <Schema>
        http://213.235.6.202:3666/BenbowWFS/SCHEMA/Sany/DataSource.xsd
      </Schema>
     </Format>
    </Supported>
   </ComplexData>
  </Input>
  <Input minOccurs="0" maxOccurs="1">
   <ows:Identifier>temperature_source</ows:Identifier>
   <ows:Title>Source to use for temperature data</ows:Title>
   <ComplexData>
    <Default>
     <Format>
      <MimeType>text/xml</MimeType>
      <Encoding>UTF-8</Encoding>
      <Schema>
```

```
          http://213.235.6.202:3666/BenbowWFS/SCHEMA/Sany/DataSource.xsd
        </Schema>
       </Format>
      </Default>
      <Supported>
       <Format>
        <MimeType>text/xml</MimeType>
        <Encoding>UTF-8</Encoding>
        <Schema>
          http://213.235.6.202:3666/BenbowWFS/SCHEMA/Sany/DataSource.xsd
        </Schema>
       </Format>
      </Supported>
     </ComplexData>
    </Input>
    <Input minOccurs="0" maxOccurs="1">
     <ows:Identifier>salinity_source</ows:Identifier>
     <ows:Title>Source to use for salinity data</ows:Title>
     <ComplexData>
      <Default>
       <Format>
        <MimeType>text/xml</MimeType>
        <Encoding>UTF-8</Encoding>
        <Schema>
          http://213.235.6.202:3666/BenbowWFS/SCHEMA/Sany/DataSource.xsd
        </Schema>
       </Format>
      </Default>
      <Supported>
       <Format>
        <MimeType>text/xml</MimeType>
        <Encoding>UTF-8</Encoding>
        <Schema>
          http://213.235.6.202:3666/BenbowWFS/SCHEMA/Sany/DataSource.xsd
        </Schema>
       </Format>
      </Supported>
     </ComplexData>
    </Input>
    <Input minOccurs="1" maxOccurs="1">
     <ows:Identifier>location</ows:Identifier>
     <ows:Title>Location to model</ows:Title>
     <ComplexData>
      <Default>
       <Format>
        <MimeType>text/xml; subtype=gml/3.1.1</MimeType>
        <Encoding>UTF-8</Encoding>
        <Schema>
       http://213.235.6.202:3666/BenbowWFS/SCHEMA/gml/3.1.1/base/gml.xsd#Point
        </Schema>
       </Format>
      </Default>
      <Supported>
       <Format>
        <MimeType>text/xml; subtype=gml/3.1.1</MimeType>
        <Encoding>UTF-8</Encoding>
        <Schema>
       http://213.235.6.202:3666/BenbowWFS/SCHEMA/gml/3.1.1/base/gml.xsd#Point
        </Schema>
       </Format>
```

```
     </Supported>
    </ComplexData>
   </Input>
   <Input minOccurs="1" maxOccurs="1">
    <ows:Identifier>simulation_time</ows:Identifier>
    <ows:Title>Time to model</ows:Title>
    <ComplexData>
     <Default>
      <Format>
       <MimeType>text/xml; subtype=gml/3.1.1</MimeType>
       <Encoding>UTF-8</Encoding>
       <Schema>
http://213.235.6.202:3666/BenbowWFS/SCHEMA/gml/3.1.1/base/temporal.xsd#TimeIn
stant
       </Schema>
      </Format>
     </Default>
     <Supported>
      <Format>
       <MimeType>text/xml; subtype=gml/3.1.1</MimeType>
       <Encoding>UTF-8</Encoding>
       <Schema>
http://213.235.6.202:3666/BenbowWFS/SCHEMA/gml/3.1.1/base/temporal.xsd#TimeIn
stant
       </Schema>
      </Format>
     </Supported>
    </ComplexData>
   </Input>
  </DataInputs>
  <ProcessOutputs xmlns:wps="http://www.opengis.net/wps/1.0"
xmlns:ows="http://www.opengis.net/ows/1.1">
   <Output>
    <ows:Identifier>probability_of_exceedence</ows:Identifier>
    <ows:Title>Probability of Exceedence</ows:Title>
    <LiteralOutput>
     <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#double">
       double
     </ows:DataType>
     <UOMs>
      <Default>
       <ows:UOM>fraction</ows:UOM>
      </Default>
      <Supported>
       <ows:UOM>fraction</ows:UOM>
      </Supported>
     </UOMs>
    </LiteralOutput>
   </Output>
   <Output>
    <ows:Identifier>used_bathing_area</ows:Identifier>
    <ows:Title>Retrieved bathing area</ows:Title>
    <LiteralOutput>
     <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string">
       string
     </ows:DataType>
    </LiteralOutput>
   </Output>
   <Output>
    <ows:Identifier>used_wind_data</ows:Identifier>
```

```
    <ows:Title>Retrieved wind data</ows:Title>
    <LiteralOutput>
     <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string">
       string
     </ows:DataType>
    </LiteralOutput>
   </Output>
   <Output>
    <ows:Identifier>used_rain_data</ows:Identifier>
    <ows:Title>Retrieved rain data</ows:Title>
    <LiteralOutput>
     <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string">
       string
     </ows:DataType>
    </LiteralOutput>
   </Output>
   <Output>
    <ows:Identifier>used_cloud_data</ows:Identifier>
    <ows:Title>Retrieved cloud data</ows:Title>
    <LiteralOutput>
     <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string">
       string
     </ows:DataType>
    </LiteralOutput>
   </Output>
   <Output>
    <ows:Identifier>used_temperature_data</ows:Identifier>
    <ows:Title>Retrieved temperature data</ows:Title>
    <LiteralOutput>
     <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string">
       string
     </ows:DataType>
    </LiteralOutput>
   </Output>
   <Output>
    <ows:Identifier>used_salinity_data</ows:Identifier>
    <ows:Title>Retrieved salinity data</ows:Title>
    <LiteralOutput>
     <ows:DataType ows:reference="http://www.w3.org/TR/xmlschema-2/#string">
       string
     </ows:DataType>
    </LiteralOutput>
   </Output>
  </ProcessOutputs>
 </ProcessDescription>
</wps:ProcessDescriptions>
```

### 6.3.3 BMT WPS Execute

An example execute request is given below:

```
<wps:Execute service="WPS" version="1.0.0"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:ds="http://www.bmtcordah.com/sany/DataSource"
xmlns:gml="http://www.opengis.net/gml/3.1.1">
 <ows:Identifier>BWQMv1</ows:Identifier>
 <wps:DataInputs>
```

```
  <wps:Input>
   <ows:Identifier>bathing_area_id</ows:Identifier>
  <ows:Title>Bathing area ID</ows:Title>
  <wps:Data>
   <wps:LiteralValue>bathing_area.Gdynia_Srodmiescie</wps:LiteralValue>
  </wps:Data>
  </wps:Input>
 <wps:Input>
   <ows:Identifier>bathing_area_wfs_url</ows:Identifier>
   <ows:Title>URL to retrieve bathing area from</ows:Title>
  <wps:Data>

<wps:LiteralValue>http://213.235.6.202:8080/geoserver/wfs</wps:LiteralValue>
  </wps:Data>
 </wps:Input>
 <wps:Input>
  <ows:Identifier>wind_source</ows:Identifier>
   <ows:Title>Source to use for wind data</ows:Title>
  <wps:Data>
    <wps:ComplexData
schema="http://213.235.6.202:8081/WPS/SCHEMA/Sany/DataSource.xsd">
   <ds:SanyWindWps/>
   <!-- Instructs the WPS to attempt to find the IT-INNOV WPS in the
catalogue and use it -->
  </wps:ComplexData>
  </wps:Data>
 </wps:Input>
 <wps:Input>
   <ows:Identifier>rain_source</ows:Identifier>
   <ows:Title>Source to use for rain data</ows:Title>
   <wps:Data>
   <wps:ComplexData
schema="http://213.235.6.202:8081/WPS/SCHEMA/Sany/DataSource.xsd">
   <ds:Sos href="SOS_URL" offering="OFFERING_NAME" operation="OPERATION_NAME"
timePropertyName="TIME_PROPERTY" timeQueryMode="TIME_MODE"
SosNamespace="SOS_NAMESPACE">
    <ds:Property>precipitation</ds:Property>
   </ds:Sos>
  </wps:ComplexData>
  </wps:Data>
 </wps:Input>
 <wps:Input>
   <ows:Identifier>cloud_source</ows:Identifier>
   <ows:Title>Source to use for cloud data</ows:Title>
   <wps:Data>
   <wps:ComplexData
schema="http://213.235.6.202:8081/WPS/SCHEMA/Sany/DataSource.xsd">
   <ds:LiteralData>8</ds:LiteralData>
   </ds:Sos>
  </wps:ComplexData>
  </wps:Data>
 </wps:Input>
 <wps:Input>
  <ows:Identifier>location</ows:Identifier>
  <ows:Title>Location to model</ows:Title>
  <wps:Data>
   <wps:ComplexData
schema="http://213.235.6.202:8081/WPS/SCHEMA/GML/3.1.1/base/gml.xsd#Point">
   <gml:Point srsName="urn:ogc:def:crs:EPSG:4326">
    <gml:pos>54.51545397052842 18.550109319788834</gml:pos>
```

```
    </gml:Point>
   </wps:ComplexData>
   </wps:Data>
  </wps:Input>
  <wps:Input>
   <ows:Identifier>simulation_time</ows:Identifier>
   <ows:Title>Time to model</ows:Title>
   <wps:Data>
    <wps:ComplexData
schema="http://213.235.6.202:8081/WPS/SCHEMA/GML/3.1.1/base/temporal.xsd#Time
Instant">
    <gml:TimeInstant>
     <gml:timePosition>2006-11-26T12:00:00Z</gml:timePosition>
    </gml:TimeInstant>
   </wps:ComplexData>
   </wps:Data>
  </wps:Input>
  </wps:DataInputs>
  <wps:ResponseForm/>
</wps:Execute>
```

The response to this execute request might look like this:

```
<?xml version="1.0" encoding="utf-8"?>
<wps:ExecuteResponse xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:wps="http://www.opengis.net/wps/1.0.0"
xmlns:ows="http://www.opengis.net/ows/1.1"
xmlns:ds="http://www.bmtcordah.com/sany/DataSource" service="WPS"
version="1.0.0"
serviceInstance="http://213.235.6.202:3666/BenbowWFS/OGCWS.ashx?service=WPS&a
mp;request=GetCapabilities" xml:lang="en-GB"
xsi:schemaLocation="http://www.opengis.net/wps/1.0.0
http://213.235.6.202:8081/WPS/SCHEMA/wps/1.0.0/wpsAll.xsd
http://www.bmtcordah.com/sany/DataSource
http://213.235.6.202:8081/WPS/SCHEMA/sany/DataSource.xsd">
 <wps:Process wps:processVersion="1.0.0">
  <ows:Identifier>BWQMv1</ows:Identifier>
  <ows:Title>Bathing Water Quality Model v1</ows:Title>
  <ows:Abstract>Bathing Water Quality Model v1 developed by BMT Cordah Ltd
for use in the SANY EU research project.</ows:Abstract>
 </wps:Process>
 <wps:Status creationTime="2009-05-18T15:28:25.2429986+01:00">
  <wps:ProcessSucceeded />
 </wps:Status>
 <wps:DataInputs>
  <!-- This section copied from the calling Execute document -->
 </wps:DataInputs>
 <wps:OutputDefinitions>
  <wps:Output asReference="false">
   <ows:Identifier>probability_of_exceedence</ows:Identifier>
   <ows:Title>Probability of Exceedence</ows:Title>
  </wps:Output>
  <wps:Output asReference="false">
   <ows:Identifier>used_bathing_area</ows:Identifier>
   <ows:Title>Retrieved bathing area</ows:Title>
  </wps:Output>
  <wps:Output asReference="false">
   <ows:Identifier>used_rain_data</ows:Identifier>
   <ows:Title>Retrieved rain data</ows:Title>
  </wps:Output>
  <wps:Output asReference="false">
   <ows:Identifier>used_wind_data</ows:Identifier>
   <ows:Title>Retrieved wind data</ows:Title>
  </wps:Output>
  <wps:Output asReference="false">
   <ows:Identifier>used_cloud_data</ows:Identifier>
   <ows:Title>Retrieved cloud data</ows:Title>
  </wps:Output>
  <wps:Output asReference="false">
   <ows:Identifier>used_temperature_data</ows:Identifier>
   <ows:Title>Retrieved temperature data</ows:Title>
  </wps:Output>
  <wps:Output asReference="false">
   <ows:Identifier>used_salinity_data</ows:Identifier>
   <ows:Title>Retrieved salinity data</ows:Title>
  </wps:Output>
 </wps:OutputDefinitions>
 <wps:ProcessOutputs>
  <wps:Output>
   <ows:Identifier>probability_of_exceedence</ows:Identifier>
   <ows:Title>Probability of exceedence</ows:Title>
```

```
    <wps:Data>
     <wps:LiteralData>30.5173223418932</wps:LiteralData>
    </wps:Data>
   </wps:Output>
   <wps:Output>
    <ows:Identifier>used_bathing_area</ows:Identifier>
    <ows:Title>Retrieved bathing area</ows:Title>
    <wps:Data>
     <wps:LiteralData>
      <!-- The result of the GetFeature request that was used to retrieve the
bathing area -->
   </wps:LiteralData>
    </wps:Data>
   </wps:Output>
   <wps:Output>
    <ows:Identifier>used_rain_data</ows:Identifier>
    <ows:Title>Retrieved rain data</ows:Title>
    <wps:Data>
     <wps:LiteralData>&lt;LiteralData /&gt;</wps:LiteralData>
    </wps:Data>
   </wps:Output>
   <wps:Output>
    <ows:Identifier>used_wind_data</ows:Identifier>
    <ows:Title>Retrieved rain data</ows:Title>
    <wps:Data>
     <wps:LiteralData>&lt;LiteralData /&gt;</wps:LiteralData>
    </wps:Data>
   </wps:Output>
   <wps:Output>
    <ows:Identifier>used_cloud_data</ows:Identifier>
    <ows:Title>Retrieved cloud data</ows:Title>
    <wps:Data>
     <wps:LiteralData>&lt;LiteralData /&gt;</wps:LiteralData>
    </wps:Data>
   </wps:Output>
   <wps:Output>
    <ows:Identifier>used_temperature_data</ows:Identifier>
    <ows:Title>Retrieved temperature data</ows:Title>
    <wps:Data>
     <wps:LiteralData>&lt;NoneUsed /&gt;</wps:LiteralData>
    </wps:Data>
   </wps:Output>
   <wps:Output>
    <ows:Identifier>used_salinity_data</ows:Identifier>
    <ows:Title>Retrieved salinity data</ows:Title>
    <wps:Data>
     <wps:LiteralData>&lt;NoneUsed /&gt;</wps:LiteralData>
    </wps:Data>
   </wps:Output>
  </wps:ProcessOutputs>
</wps:ExecuteResponse>
```

### 6.3.4 BMT SPS GetCapabilities

Due to the similarities between WPS and SPS capabilities documents, this example will centre around solely the section that is different – the “contents” section of the document.

```
<?xml version="1.0" encoding="utf-8"?>
<GetCapabilities service="SPS" version="1.0.0"
xmlns="http://www.opengis.net/sps">
 <Sections>
  <Section>contents</Section>
 </Sections>
</GetCapabilities>
```

The following capabilities response document contains only the contents section, as requested above. This, hypothetical, SPS contains only a single sensor measuring a single phenomenon. It is worth noting that in these examples, the text encoding has been detected by way of the "accept-encoding" HTTP header.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<sps:Capabilities version="1.0.0" service="SPS" xml:lang="en-GB"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xmlns:xlink="http://www.w3.org/1999/xlink"
xmlns:xi="http://www.w3.org/2003/XInclude"
xmlns:gml="http://www.opengis.net/gml"
xmlns:ows="http://www.opengis.net/ows/1.1" updateSequence="0"
xsi:schemaLocation="http://www.opengis.net/sps
/WPS/SCHEMA/sps/1.0.0/spsGetCapabilities.xsd"
xmlns:sps="http://www.opengis.net/sps">
 <sps:Contents>
  <sps:SensorOfferingList>
   <sps:SensorOffering>
    <sps:AreaOfService>
     <ows:WGS84BoundingBox>
      <ows:LowerCorner>14.314721 48.273333</ows:LowerCorner>
      <ows:UpperCorner>14.314721 48.273333</ows:UpperCorner>
     </ows:WGS84BoundingBox>
    </sps:AreaOfService>
    <sps:Phenomenon>urn:ogc:def:phenomenon:arcs:no2</sps:Phenomenon>
    <sps:SensorDefinition>http://bmt-sps/sensors/bmt_arcs-s416-
no2</sps:SensorDefinition>
    <sps:SensorID>urn:ogc:object:sensor:bmt:arcs-s416-no2</sps:SensorID>
   </sps:SensorOffering>
  </sps:SensorOfferingList>
  <sps:PhenomenonOfferingList>
   <sps:PhenomenonOffering>
    <sps:Phenomenon>urn:ogc:def:phenomenon:arcs:no2</sps:Phenomenon>
    <sps:SensorID>urn:ogc:object:sensor:bmt:arcs-s416-no2</sps:SensorID>
   </sps:PhenomenonOffering>
  </sps:PhenomenonOfferingList>
 </sps:Contents>
</sps:Capabilities>
```

### 6.3.5 BMT SPS DescribeTasking

The DescribeTasking request asks for the inputs required to task a sensor. Following the example from above, here is a DescribeTasking request:

```
<?xml version="1.0" encoding="utf-8"?>
<DescribeTasking xmlns="http://www.opengis.net/sps" service="SPS"
version="1.0.0">
 <sensorID>urn:ogc:object:sensor:bmt:arcs-s416-no2</sensorID>
</DescribeTasking>
```

A matching response follows this paragraph. In the BMT pilot, none of the sensors have parameters.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<DescribeTaskingRequestResponse xmlns="http://www.opengis.net/sps"
xmlns:sps="http://www.opengis.net/sps" xml:lang="en-GB"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sps
/WPS/SCHEMA/sps/1.0.0/spsGetCapabilities.xsd">
 <sps:taskingDescriptor>
  <sps:sensorID>urn:ogc:object:sensor:bmt:arcs-s416-no2</sps:sensorID>
 </sps:taskingDescriptor>
</DescribeTaskingRequestResponse>
```

### 6.3.6 BMT SPS Submit

Following the above example, none of the sensors have any inputs, so submit requests are extremely simple. Also, the notificationTarget element (used for integration with Web Notification Services) is not currently supported.

```
<?xml version="1.0" encoding="utf-8"?>
<Submit service="SPS" version="1.0.0">
 <sps:notificationTarget/>
 <sps:sensorParam>
  <sps:sensorID>urn:ogc:object:sensor:bmt:arcs-s416-no2</sps:sensorID>
 </sps:sensorParam>
</Submit>
```

Providing a successful tasking has taken place, the following response would be expected. Note that the "taskID" is a Windows GUID used as a unique reference to a particular tasking.

```
<?xml version="1.0" encoding="iso-8859-1"?>
<SubmitRequestResponse xmlns="http://www.opengis.net/sps" service="SPS"
version="1.0.0" xml:lang="en-GB" xmlns:xsi="http://www.w3.org/2001/XMLSchema-
instance" xsi:schemaLocation="http://www.opengis.net/sps
/WPS/SCHEMA/sps/1.0.0/spsGetCapabilities.xsd">
 <taskID>A0292A72-7DEF-4a42-B3D0-7A2C0AB7D807</taskID>
 <status>confirmed</status>
</SubmitRequestResponse>
```

### 6.3.7 BMT SPS GetStatus

Once a task is running, the optional "GetStatus" request allows a user to request the current status of the tasking. Following the example task ID given above:

```
<?xml version="1.0" encoding="utf-8"?>
<GetStatus xmlns="http://www.opengis.net/sps" version="1.0.0" service="SPS">
 <taskID>A0292A72-7DEF-4a42-B3D0-7A2C0AB7D807</taskID>
</GetStatus>
```

Assuming the task had been started recently and was currently retrieving data from it's source SOS for training (this would only make a difference to the "gml:description" content), the response might be:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<GetStatusRequestResponse xmlns="http://www.opengis.net/sps"
xmlns:gml="http://www.opengis.net/gml" xml:lang="en-GB"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sps
/WPS/SCHEMA/sps/1.0.0/spsGetCapabilities.xsd">
 <taskID>A0292A72-7DEF-4a42-B3D0-7A2C0AB7D807</taskID>
 <status>in operation</status>
 <gml:description>Retrieving training data</gml:description>
</GetStatusRequestResponse>
```

### 6.3.8 BMT SPS DescribeResultAccess

Part of the SPS specification is that a sensor should give details about where it's results can be retrieved from (the SPS does not return results to the caller itself – it only allows for control of and feedback on the state of the "sensor"). The SPS should be able to respond to this request for either a sensor (assuming the sensor has a fixed result store) or for an individual tasking (assuming the sensors result store can be changed).

Assuming the request was for a given sensor:

```
<?xml version="1.0" encoding="utf-8"?>
<DescribeResultAccess xmlns="http://www.opengis.net/sps"
xmlns:sps="http://www.opengis.net/sps" version="1.0.0" service="SPS">
 <sps:sensorID>urn:ogc:object:sensor:bmt:arcs-s416-no2</sps:sensorID>
</DescribeResultAccess>
```

Or for a specific task:

```
<?xml version="1.0" encoding="utf-8"?>
<DescribeResultAccess xmlns="http://www.opengis.net/sps"
xmlns:sps="http://www.opengis.net/sps" version="1.0.0" service="SPS">
 <sps:taskID>A0292A72-7DEF-4a42-B3D0-7A2C0AB7D807</sps:taskID>
</DescribeResultAccess>
```

In these examples, an output similar to the following would be expected:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<DescribeResultAccess xmlns="http://www.opengis.net/sps"
xmlns:sps="http://www.opengis.net/sps" version="1.0.0" service="SPS"
xml:lang="en-GB" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sps
/WPS/SCHEMA/sps/1.0.0/spsGetCapabilities.xsd">
 <service>
  <ServiceType>SOS</ServiceType>
  <ServiceURL>http://enviro6.arcs.ac.at/SOS-BMT/sos</ServiceURL>
  <request>&lt;GetObservation service=&quot;SOS&quot;
version=&quot;1.0.0&quot; xmlns=&quot;http://www.opengis.net/sos/1.0&quot;
xmlns:gml=&quot;http://www.opengis.net/gml&quot;
xmlns:ogc=&quot;http://www.opengis.net/ogc&quot;&gt;
 &lt;offering&gt;Linz&lt;/offering&gt;

&lt;observedProperty&gt;urn:ogc:def:phenomenon:arcs:no2&lt;/observedProperty&
gt;
 &lt;featureOfInterest&gt;
  &lt;ObjectID&gt;urn:ogc:object:feature:arcs:s416&lt;/ObjectID&gt;
 &lt;/featureOfInterest&gt;
 &lt;responseFormat&gt;
text/xml;subtype=&quot;om/1.0.0&quot;&lt;/responseFormat&gt;
&lt;/GetObservation&gt;</request>
 </service>
</DescribeResultAccess>
```

### 6.3.9 BMT SPS Cancel

A running task can be cancelled (this is especially important in the case of constantly-running tasks such as the virtual sensors in the BMT pilot). The cancel request to match the examples above is given below:

```
<?xml version="1.0" encoding="utf-8"?>
<Cancel xmlns="http://www.opengis.net/sps" version="1.0.0" service="SPS">
 <taskID>A0292A72-7DEF-4a42-B3D0-7A2C0AB7D807</taskID>
</Cancel>
```

Assuming the cancel request was successful, the following response would be given:

```
<?xml version="1.0" encoding="iso-8859-1"?>
<CancelRequestResponse xmlns="http://www.opengis.net/sps"
xmlns:sps="http://www.opengis.net/sps" version="1.0.0" service="SPS"
xml:lang="en-GB" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengis.net/sps
/WPS/SCHEMA/sps/1.0.0/spsGetCapabilities.xsd">
 <sps:taskID>A0292A72-7DEF-4a42-B3D0-7A2C0AB7D807</sps:taskID>
 <sps:status>confirmed</sps:status>
</CancelRequestResponse>
```