Deliverable D4.1.2 Concerted Approach V2
Appendix 3: SUDPLAN Service Interface Layer

Revision History			
Version	**Date**	**Modified by**	**Comments**
0.1	2011-11-14	Martin Scholl	Initial document template
0.2	2011-11-12	Peter Kutschera	SOS, SPS, TS-Toolbox
0.3	2011-12-14	Mihai Bartha	Small changes.
0.4	2011-12-20	Lars Örtegren	WMS additions.
0.5	2012-01-13	Sascha Schlobinski	Major revision

Copyright © SUDPLAN

Table of Contents

Table of Figures

Copyright © SUDPLAN

Glossary

Information product	Raw data, such as the results of mathematical modelling, and the analysis thereof, will often need to be packaged in such a way as to be accessible to the various stakeholders of an analysis. The medium can be one of a wide variety, such as print, photo, video, slides, or web pages. The term *information product* refers to such an entity.
Model	A *model* is a simplified representation of a system, usually intended to facilitate analysis of the system through manipulation of the model. In the SUDPLAN context the term can be used to refer to mathematical models of processes or spatial models of geographical entities.
Profile	Within SUDPLAN a *profile* is a set of configuration parameters which are associated with an individual or group, and which are remembered in order to facilitate repeated use of the system.
Report	A *report* is a particular type of information product which is usually static and might integrate still images, static data representations, mathematical expressions, and narrative to communicate an analytical result to others.
Scenario	A *scenario* is a set of parameters, variables and other conditions which represent a hypothetical situation, and which can be analysed through the use of models in order to produce hypothetical outcomes.
Scenario Management System	*Scenario Management System* is synonymous with SUDPLAN platform
SUDPLAN application	A *SUDPLAN application* is a decision support system crafted by using the SUDPLAN platform and integrating models, data, sensors, and other services to meet the requirements of the particular application.
SUDPLAN platform	The *SUDPLAN platform* is an ensemble of software components which support the development of SUDPLAN applications.
SUDPLAN system	*SUDPLAN system* is synonymous with SUDPLAN application

User	The term *user* refers to people who have a more or less direct involvement with a system. Primary users are directly and frequently involved, while secondary users may interact with the system only occasionally or through an intermediary. Tertiary users may not interact with the system but have a direct interest in the performance of the system.
Web-based	Computer applications are said to be *web-based* if they rely on or take advantage of data and/or services which are accessible via the World Wide Web using the Internet.

1. Management Summary

This document describes the Service Interface layer of the SUDPLAN System architecture and serves as an abstract (implementation independent) specification for SUDPLAN data retrieval and model integration. It describes the common concepts and conventions used for all SUDPLAN services. Service-specific documentation like for example service endpoints is available in separate documents. The document focuses on OGC Services used at the SUDPLAN service layer.

This is a living document; the current version is available from www.sudplan.eu.

This version reflects the implementation at the end of 2011 and concentrates on the usage of the services using Timeseries-Toolbox. The OGC-related part will be completed in 2012.

This document targets all SUDPLAN partners. Partners involved in the implementation of the Scenario Management System and the Common Services.

Copyright © SUDPLAN

2. Service layer architectural overview

The Service side part of the SUDPLAN architecture is realized according to OGC service specifications. The service interfaces used are Web Map Service (WMS), Sensor Observation Service (Transactional profile) SOS(-T) and the Sensor Planning Service (SPS) as described later.

It is essential to understand, that these service definitions only specify interfaces and say nothing about the used implementation, the internal data model or the number of different service implementations. In fact SUDPLAN uses a lot of service implementations, some of them are using existing open source implementations, and some implementations have been developed within the project.

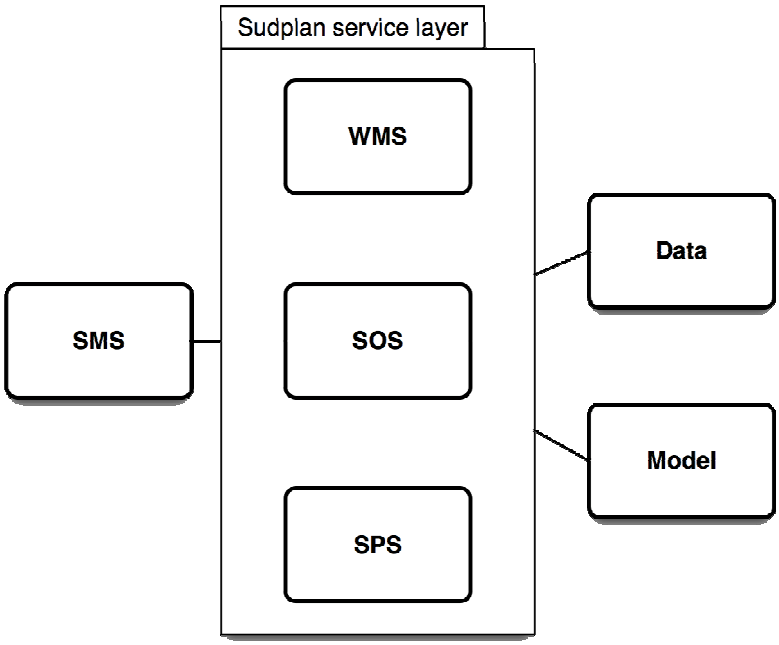

Fig. 1 SUDPLAN OGC service layer

WMS (Web Map Service) is used to deliver maps as images. This service is used within SUDPLAN to deliver overview pictures for the visualisation of data on a background map (which is also provided by one or more WMS). The WMS is good in providing visualisations of extensive data, showing the location of something and providing background maps. But since all this is provided as pictures only the data cannot be used for further processing.

SOS (Sensor Observation Service) is used to deliver time series of data for further processing by a client. This service is used to transfer existing data and model results. For this purpose the data is encoded using the OGC O&M (Observation and Management) data encoding standard.

SOS-T is the transactional profile of the SOS, providing the possibility to transfer data from the client to the server (insertObservation()). This service is used to transfer model input data to a location that is directly accessible from the model system machine.

SPS (Sensor Planning Service) is used to control model execution. SPS is preferred over WPS (Web Processing Service) since SPS provides a better control of the model execution. (See [Kuts2011b]).SUDPLAN Service Call sequences

2.1. WMS data retrieval overview

In SUDPLAN, the WMS service is used to retrieve rendered images (overview) of different climate scenarios. The published layers are organized in a hierarchy consisting of environmental factors, scenarios and variables. Each variable has a time series of images, each one representing a 10-year period from 1960 to 2100.

The workflow to retrieve images from the WMS is:

1. Get the capabilities document; containing among other things the list of available layers.

2. Get an image in the requested format and resolution.

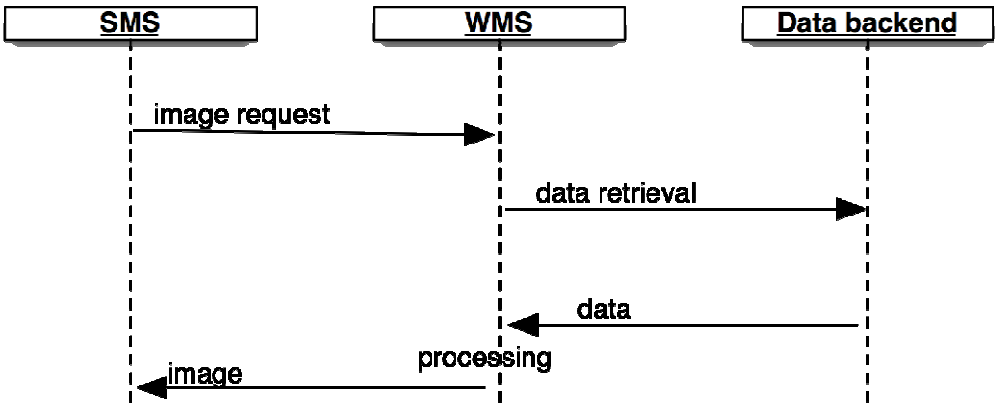

Fig. 2 WMS data retrieval sequence

2.2. SOS data flow

In SUDPLAN a number of SOS services have been implemented, configured and deployed. But the general workflow is the same for all implementations; whereas the main differences lie in the respective communication with the service-backend. The data organisation from the SOS point of view is a list of sensors, each providing a time series of values. Sensors are identified by a filter and contain a description, including a description of the provided data.

The workflow to get data from a SOS is:

1. Get the capabilities document containing the list of sensors available
2. Ask for one sensor description
3. Get data from this sensor

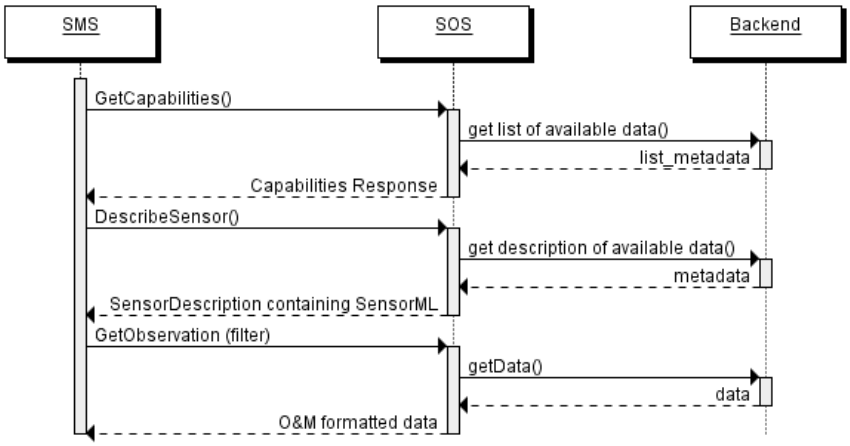

Fig. 3 SOS data retrieval sequence

If the SOS implements the transactional profile (SOS-T) there is an additional workflow to upload data (see Fig. 4).

1. Register a new sensor (to be skipped when adding data to an existing sensor)
2. Insert data for this sensor

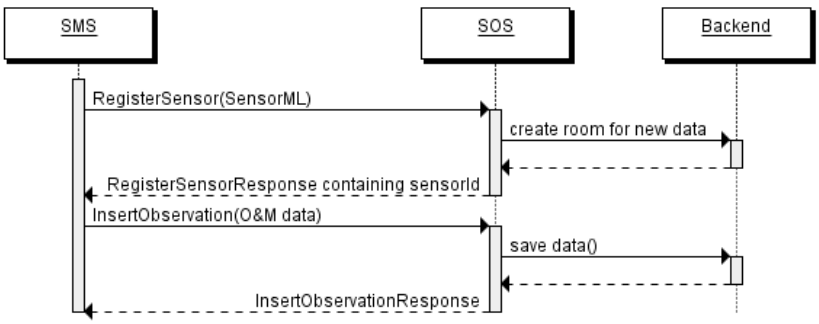

Fig. 4 SOS submit data sequence

2.3. SWE service workflow for model execution

As depicted in Fig. 5 in SUDPLAN pre-existing models are made available by providing a SPS (Sensor Planning Service) interface to control the model.

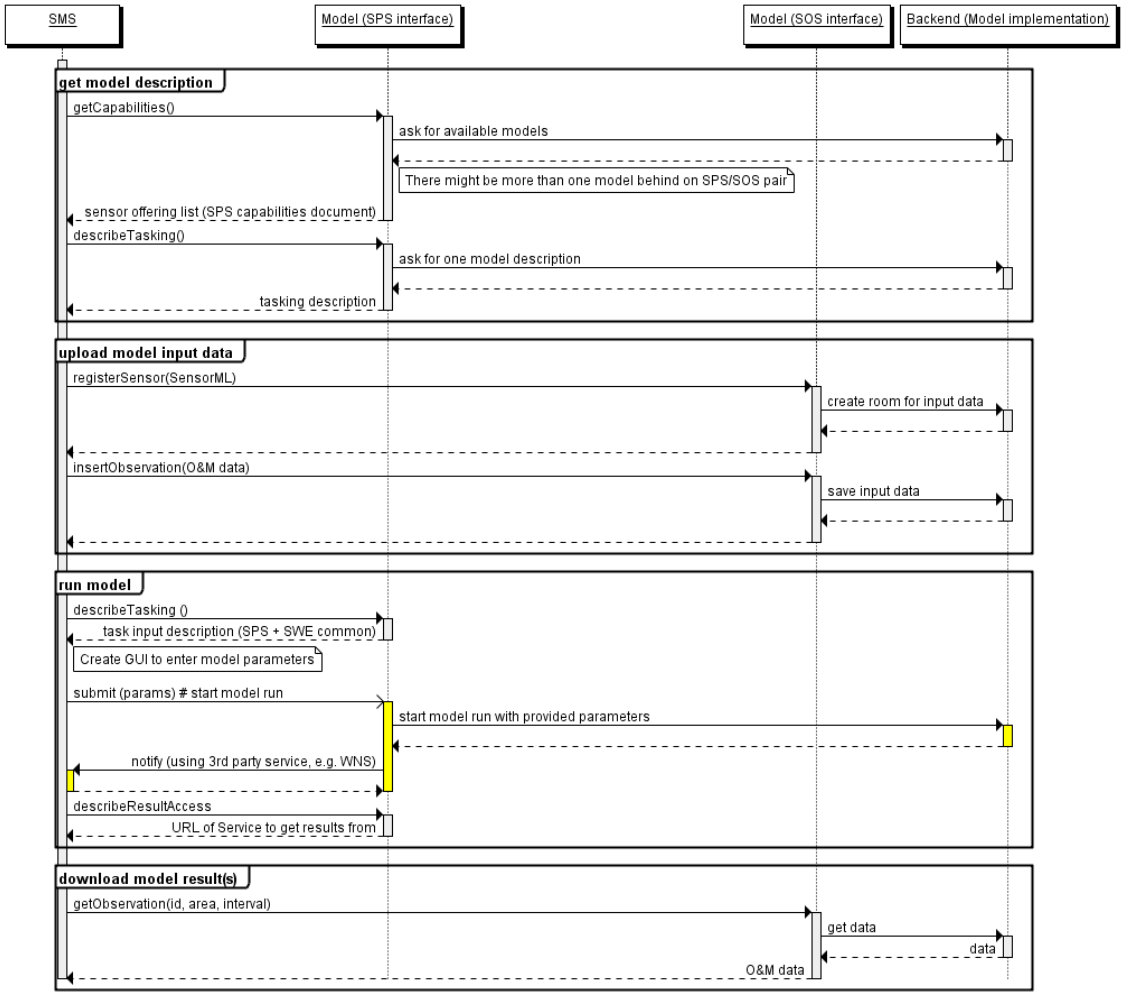

Fig. 5 Model control with SWE[1] services

Since the SPS is meant to transport only relatively small model parameters an additional SOS-T is used to transport larger datasets used as input or output of the model. So in general there are two OGC service interfaces used to control one model. The workflow in general is:

1. Get a model description
2. Upload input data using the SOS-T interface of the model
3. Run the model controlled by the SPS interface of the model. This includes asking for needed parameters, there types and feasible value ranges. This information e.g. can be used to generate a GUI for parameter input. The model also provides a description how to access the model results.
4. Download the results using the SOS interface of the model. In fact here the additional operations coming from the transactional profile are not used.

[1] OGC Sensor Web Enablement

3. SUDLAN Service Layer

3.1. Web Map Service (WMS)

WMS are generally used to produce images from vector or raster data. SUDPLAN uses WMS to provide climate scenario data as images and extends the WMS by keywords to gain linkage between WMS and SOS data.

WMS extension

SUDPLAN extends the WMS capabilities with keywords for the layers to make the linkage between different WMS layers and the access to data in all available time resolutions as rasters or time series from a specified point. This data is accessed using the information from the keywords when accessing SOS. There is also an extension to mark a layer as an “extended layer”.

Array mark convention

The layer that contains the layers for each point in time for a specific environmental factor, scenario and variable, can be considered to be an array. The name of this layer must end with an empty bracket enclosure “[]”, e.g. “Temperature[]”. This is used by the SMS to identify the next level as a time series of fields.

Keyword extension

The layer that contains the layers for each point in time for a specific environmental factor should also contain a list of keywords with the necessary information to provide the linkage to the data via SOS.

Parameter	Value restriction	Restriction description
KeywordList	Keyword objects	• Any array marked layer shall contain at least the seven mandatory keywords.
Keyword	ts:observed_property=<the observed property>	• The observed property value that points to the raw data of which the marked layer was constructed of. The observed property value is used to access time series data from the SOS service.
	ts:procedure=<the procedure>	• The procedure value that points to the raw data of which the marked layer was constructed of. This procedure value is used to access time series data from the SOS service.

	Ts:feature_of_interest=<the feature of interest>	• The feature of interest value is used to access time series data from the SOS service.
	Ts:offering=<the offering>	• The offering is used to access time series data from the SOS service.
	Sos_url=<the sos endpoint url>	• The URL of the SOS service endpoint. Requests are formulated using the values from the 4 rows above.
	Ts:available_data_min=<time of first available field>	• The date/time of the first available value in the time series. Can be used in the GUI without the need to access the SOS first.
	Ts:available_data_max=<time of last available field>	• The date/time of the last available field value in the time series. Can be used in the GUI without the need to access the SOS first.

Example

```
<WMT_MS_Capabilities version="1.1.1" updateSequence="214">
   […]
   <Layer>
      <Name>Temperature[]</Name>
      <Title>Temperature[]</Title>
      <KeywordList>
         <Keyword>ts:observed_property=urn:ogc:def:property:OGC:temp</Keyword>
         <Keyword>ts:procedure=urn:ogc:object:hadleya1b:temp:10Y</Keyword>
         <Keyword>ts:feature_of_interest=urn:SMHI:feature:Europe</Keyword>
         <Keyword>ts:offering=climate_hadleya1b_temp_10Y</Keyword>
         <Keyword>sos_url=http://sudplan.ait.ac.at:8083</Keyword>
         <Keyword>ts:available_data_min=19650101T000000</Keyword>
         <Keyword>ts:available_data_max=20950101T000000</Keyword>
      </KeywordList>
   […]
</WMT_MS_Capabilities>
```

3.2. Sensor Observation Service (SOS-T)

SUDPLAN SOS/SOS-T implementation provides a standard OGC interface. All mandatory operations are implemented.

Since the data storage resides in an existing system a set of scripts (back-back end) have been provided by the owner of the existing system to allow the SOS/SOS-T implementation access to the actual data. The downside of this is that the execution of scripts is more time consuming than a direct access to the data. But as there is the need to split the service implementation from the data storage to allow the service to be accessible from the internet and have the data in a more secure place behind a firewall this separation is needed anyway.

While the realisation of a Service by just wrapping scripts sounds simple the implementation in reality is not since there is no one to one mapping between SOS operations and scripts and the data encodings are completely different.

SOS and SOS-T extension

There are no extensions to the SOS / SOS-T interface specification.

Encoding Conventions

The following conventions should be known by the user:

SUDPLAN's implementation uses the convention, that there is a one to one mapping between offerings and sensors. So the list of available sensors can be obtained very easily.

SOS defines the encoding of timeseries of simple types very well. But for more complex data structures (e.g. timeseries of data fields) a special encoding is implemented. To be standard compliant the optional SOS operation DescribeFeatureType is used to get a XML schema of a new data type extending the usual "sampledFeature" data type. This is described in more detail in the next chapter.

Encoding

The encoding of simple time series data (one float per time stamp) is well described within O&M. The O&M conform encoding of complex data types is not that well defined. Observations and Measurements - Sampling Features (O&M-SF[OGC2007b] specification provides means to encode discrete coverages and provides dedicated elements to do so, but it lacks dedicated elements necessary to describe continuous coverages.

In SUDPLAN the following method has been considered for describing continuous coverages. The optional SOS method sos:DescribeFeatureType allows for retrieval of the feature type description (xml schema) for a given feature of interest. This allows for the introduction of a new namespace containing additional SamplingFeature types. The new SamplingGrid type inherits

from the sa:SpatiallyExtensiveSamplingFeatureType defined in O&M-SF and contains a gml:RectifiedGrid element that describes the rectified grid on which the sampling takes place. The advantage of the mentioned inheritance is that the sampledFeature relation is retained.

This is described in detail in [Bartha2011a], chapter 3.3.

Limitations

While the response time to moderate requests is quite short the encoding and decoding of large amounts of data is problematic:

- The encoded data uses a large amount of memory, so you need to allocate enough heap space for your java virtual machine (java –x parameters).

- The second problem is that the process of encoding of data is quite fast, but the decoding of data is slow since the float values are encoded as strings within O&M and need to be parsed.
  While the implementation was tested with more than 2 million values a single request should be limited to about 100000 values. This will provide short answer time, low memory consumption and the possibility to provide a progress bar to the user.

As a work around for these limitations the current SOS implementation will only deliver some data.

The SUDPLAN SPS implementation provides a standard OGC interface. All mandatory operations have been implemented.

Since the models are available within existing systems a set of scripts has been provided by the owner of the existing system to allow the SPS implementation access to the models.

The models within SUDPLAN differ in many ways:

- Type and amount of data: from a small table up to more than 100GByte

- Execution time between minutes and weeks

- Some models provide their own job management, others don’t.

All this is hidden behind a standard SPS interface.

SPS extension

There are no extensions to the SPS interface specification.

Conventions

None

Encoding

Parameter descriptions are provided as XML-Schemas

Limitations

Notification is not implemented in yet.

4. Timeseries Toolbox as Service Backend Adapter

4.1. Role

The TimeSeries Toolbox [TSTB] is a set of java tools to the work with time series. The toolbox is used in SUDPLAN since nearly all the models input and output data are time series of data.

In SUDPLAN the toolbox is primary used in for two purposes:

1. In the SOS / SPS implementations to access data on the pre-existing systems
2. In the SMS to access the SOS / SPS servers.

While the first point is only interesting for users wanting to provide their own legacy systems as SOS / SPS services the second point is relevant for everyone who wants to write a client application to access the SUDPLAN services.This section concentrates to the second case.

4.2. Concepts

A TimeSeries in the sense of the toolbox is a sorted list of values, each with a time stamp. The values can be of any types from simple numbers to data fields or structured data types.

Timestamp	Value1	Value2
2011-11-01 00:00:00	0.0	30
2011-11-01 00:00:30	1.1	45

Without additional information this data structure is useless, so each TimeSeries has a set of metadata attached. This includes units, geo-reference and descriptions.

To access data, a wide variety of so called DataHandlers is available. They provide an abstraction of data sources and sinks. They even can be connected together to automate the dataflow between them.

Moreover the toolbox contains components for data processing, storage and retrieval, building large and complex systems for data management and also some applications like for example a viewer application.

4.3. Usage of Timeseries Toolbox

The most common usage of the Timeseries Toolbox is to get data from somewhere, e.g. a remote service or a data base. The principal steps required are:

Copyright © SUDPLAN

1. Get a DataHandler for the required protocol. Doing a lookup is the preferred way. The DataHandler acts as a proxy for the whole background system, which might be a bunch of files, a database or, as in our case, an OGC Service.

2. Configure the DataHandler to access the concrete data storage. For the OGC services the service endpoint is needed.

3. Open the DataHandler. This does some initialization.

4. Ask the DataHandler about available data. Data in a DataHandler is organized in DataPoints, each acting as an access point to get or set timeseries data. There can be an arbitrary number of "DataPoints", each defines as a source or a sink (or both) of TimeSeries data. The DataPoints can be selected by using filter properties (Name/value pairs).

5. Get a TimeSeries from the DataPoint providing a time interval parameter. A TimeSeries is a Java object containing the actual values and their time stamps.

The same procedure is used to write time series data to some sink, just the step 5 is replaced by putting data to the DataPoint:

6. Write a TimeSeries to the DataPoint providing the TimeSeries as parameter.

There is also a notification mechanism available to get informed when new data can be fetched. This mechanism can be used to connect one DataPoint acting as source to one or more DataPoints acting as sinks. This is called a "Pipe". Now new (or changed) data is automatically transferred. There can even be processing elements within a pipe.

Obviously, not every DataPoint can act as source and sink (e.g. Temperature sensor and line printer), but some can (e.g. Databases).

4.4. Data retrieval workflow

As the Timeseries-Toolbox is a tool for accessing and manipulation time series data from any source the workflow differs somehow from the usage of a plain OGC SOS service.

This introduces some limitations but on the other side provides a uniform way to access many different systems.

The main steps are:

1. Get a DataHandler object connected to the background system.
   In the shown example the background system is a SOS, but this is completely invisible to the user.
   To get a new DataHandler there are two possibilities. Either by directly instantiating with *new()* or using lookup.

   a. Using *new* is simpler but introduces a compile-time dependency to the specific DataHandler class. In particular this method is much simpler because the

DataHandler specific properties can be set directly since the class is known at compile time.

b. Using lookup avoids this dependency by looking for the appropriate class at runtime somewhere in the classpath. This allows the creation of generic clients; the actual DataHandler classes can be provided by the user at runtime and will be usable without re-compilation of the client. In this case Java reflection needs to be used to find out which properties are available and needs to be set. In the case of an interactive application this includes the requirement to create a GUI dynamically.

By using the open() method the DataHandler will be set in action using the actual provided properties. This might create some DataPoint objects in the background depending of the data available in the background system (The SOS in our example).

2. Ask the DataHandler about available filters and their values.
   To select DataPoints a filter in the form of a (Name/Value) list is provided. But since these are dependent of the background system they have to been retrieved from the DataHandler at runtime.

3. Get a DataPoint
   Since SUDPLAN uses the convention that there is exactly one DataPoint per offering this is a suitable filter to select DataPoints exactly.

4. Download Data
   This is done by invoking the *getTimeseries()* method on the DataPoint, providing a time interval to get data for. The method implementation is dependent on the concrete DataHandler, in our example the SOS is asked for data using the getObservation method. But independent of the implementation *getTimeseries()* returns a Java object of type TimeSeries, holding the requested data.

5. Upload data
   This is done in a similar way; by calling the method *putTimeseries()* providing a TimeSeries object as parameter. There is no TimeInterval parameter, the whole TimeSeries is uploaded.

6. Cleanup
   To release resources the connection should be closed using the *close()* method of the DataHandler.

Copyright © SUDPLAN

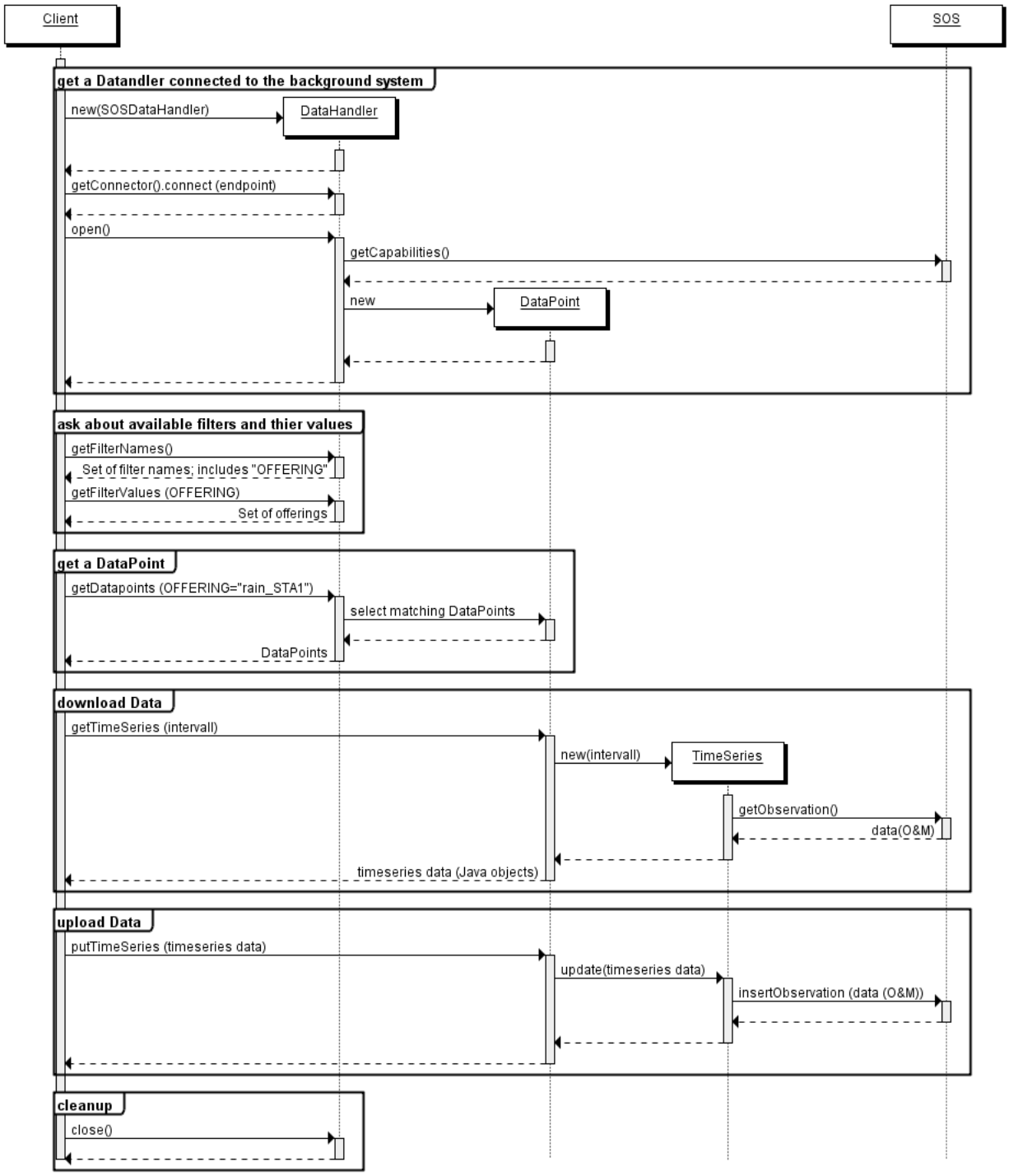

Fig. 6 Data retrieval workflow

4.5. Model execution workflow

There is no special concept of a model execution in the timeseries toolbox needed. The workflow, as implemented in SUDPLAN to control the various models, is simple:

1. Connect to a SOS to upload input data.
   This is only required if the Model needs some timeseries data as input.

2. Connect to the SPS, ask for needed parameters and send parameters and commands.
   Also download status information and where to get the result from (usually an SOS).

3. Connect to the service holding the result(s) and download them.

Points 1 and 3 are already described above. Point 2 consists of the following steps:

1. Get a DataHandler object connected to the background system.
   This is the same as for the SOS access, only another type of Handler is used.

2. Ask the DataHandler about available filters and their values.
   To select DataPoints a filter in the form of a (Name/Value) list is provided. But since these are dependent of the background system they have to been retrieved from the DataHandler at runtime.
   In the case of a SPS the values for the "PROCEDURE" property are of interest. This is the list of models known by this SPS – this can be more than one.

3. Create a new DataPoint
   Each DataPoint stands for on task executing a model. This DataPoint is now used to communicate with the task.

4. Ask Task for needed parameters
   This is done by invoking the *getProperties()* method on the DataPoint. This method returns the list of parameters understood by the model, including their descriptions, types and where appropriate value lists.

5. Set parameter and start model
   This is done by creating a time series holding all parameters and the value "start" for the "action"-value, which is the sent to the DataPoint using *putTimeseries()*.

6. Wait for results
   Now the client can request information about the task by using *getTimeseries()*. The TimeSeries will contain status information and a description where to get the results from.

7. Cleanup
   To release resources the connection should be closed using the *close()* method of the DataHandler.

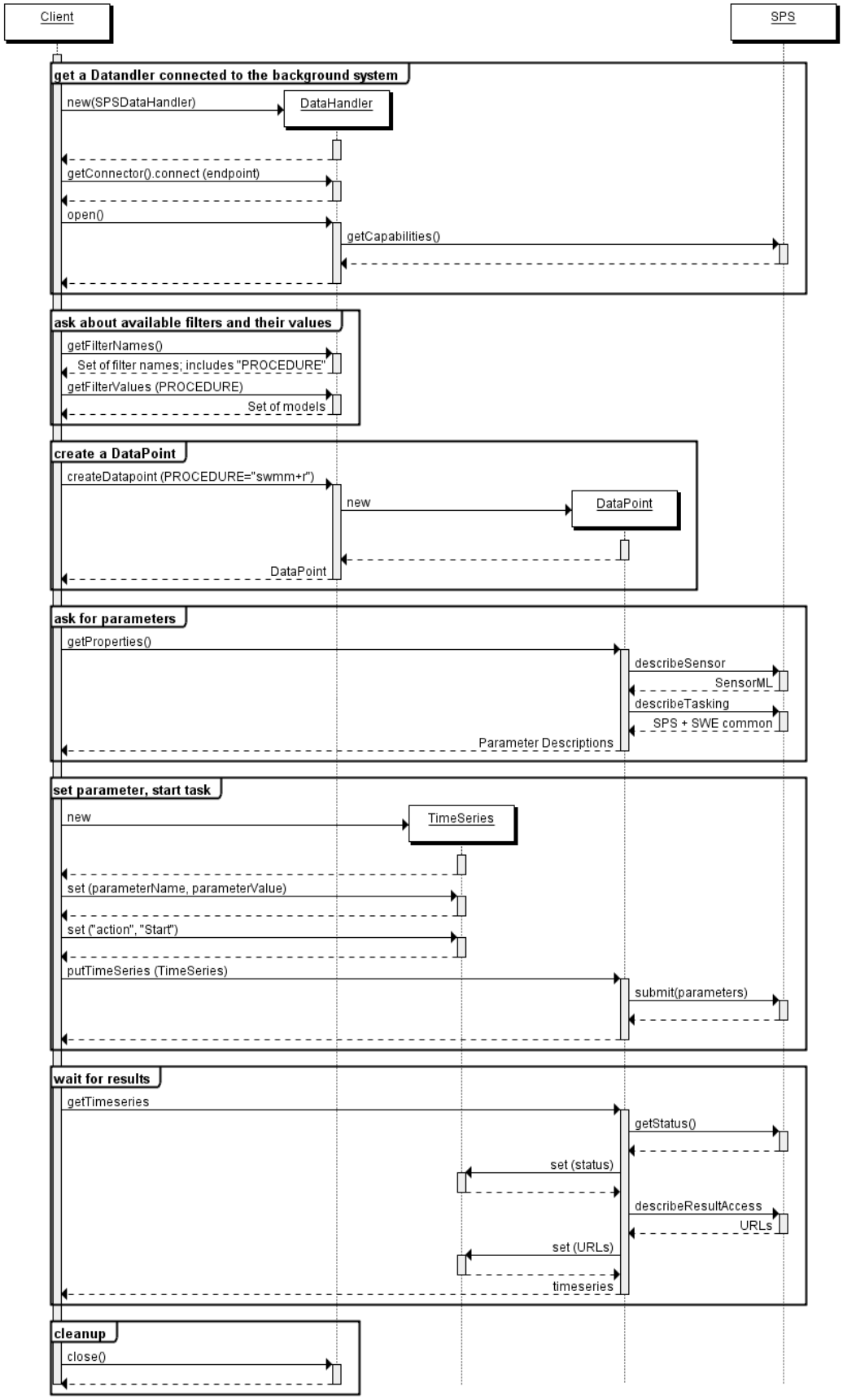

Fig. 7 Model execution workflow

References

- [SOS] OGC Sensor Observation Service specification
  http://www.opengeospatial.org/standards/sos

- [SPS] OGC Sensor Planning Service specification
  http://www.opengeospatial.org/standards/sps

- [O&M] OGC Observations and Measurement
  http://www.opengeospatial.org/standards/om

- [OGC2007b] Opengis observations and measurements part 2 - sampling features,
  http://portal.opengeospatial.org/files/?artifact id=22467

- [Kuts2011b] SUDPLAN's Experiences with the OGC-Based Model Web Services for the Climate Change Usage Area
  P. Kutschera and M. Bartha and D. Havlik
  Environmental Software Systems - Frameworks of eEnvrionment, pages 689 - 604
  ISBN: 978-3-642-22284-9

- [Bartha2011a] Sensor Web Enablement based Model Web Implementation for Climate Change Applications
  Mihai Bartha, Peter Kutschera, Denis Havlik
  Innovations in Sharing Environmental Observations and Information, pages 1-238 to 1-252
  SHAKER, ISBN: 978-3-8440-0451-9

- [TSTB] TS-Toolbox website
  http://ts-toolbox.ait.ac.at/

- SUDPLAN specific TS-Toolbox extensions available from
  http://ts-toolbox.ait.ac.at/

Copyright © SUDPLAN