WEB-BASED GEOPROCESSING AND WORKFLOW CREATION FOR GENERATING AND PROVIDING REMOTE SENSING PRODUCTS

J. Eberle, Department of Earth Observation, Friedrich-Schiller-University, Jena, Germany
C. Strobl, German Remote Sensing Data Center, German Aerospace Center, Wessling, Germany

With the implementation specification of the Web Processing Service (WPS), a standard to process spatial data on the web is available from the Open Geospatial Consortium (OGC). This new possibility in web-based information systems allows an interoperable processing of different data based on service-oriented architectures. This paper describes requirements and an implementation for web-based processing and workflow creation to generate and provide remote sensing products like fire hotspot points and land surface temperatures based on NASA MODIS data. Therefore, a WPS as well as service chains to process these remote sensing products are implemented. PyWPS is used as WPS and tested with focus on processing remote sensing data. To chain the developed processes, software like 52° North Orchestration Engine, Apache ODE and internal software for executing processing chains and providing a workflow as WPS process are tested. A Web-GIS with open-source software was developed to manage started and finished processes, to search for available satellite data and to create processing chains interactively. An interface to retrieve and process archived data was integrated; the user of this system can search interactively for archived satellite data and process it with the developed processing services and workflows. The Web-GIS also acts as a WPS client, client for visualizing processed data and creator as well as security layer for processing chains. These developments and research work show that there is additional work to improve the WPS specification for example by adding domain-specific profiles for working with large raster data to define standardised rules for data-exchange between the processes. In the field of process orchestration, no standardised data format describing processing chains exists yet.

Grâce aux spécifications d’implantation du Web Processing Service (WPS), une norme pour traiter des données spatiales sur le web est disponible en provenance de l’Open Geospatial Consortium (OGC). Cette nouvelle possibilité dans les systèmes d’information sur le Web permet un traitement interexploitable de différentes données en fonction des architectures axées sur les services. Cet article décrit les exigences et une implémentation pour le traitement sur le Web et la création d’un flux des opérations afin de générer et de fournir des produits de télédétection tels que les points à haut risque d’incendie et les températures à la surface du sol à partir des données MODIS de la NASA. Par conséquent, un WPS de même que des chaînes de services pour traiter ces produits de télédétection sont implémentés. Le PyWPS (WPS en langage Python) est utilisé comme WPS et évalué en mettant l’accent sur le traitement des données de télédétection. Pour enchaîner les processus développés, on évalue des logiciels tels que le 52° North Orchestration Engine, l’Apache ODE et un logiciel interne pour exécuter des chaînes de traitement et fournir un flux des opérations en tant que processus WPS. Un SIG sur le Web avec un logiciel ouvert a été développé pour gérer les processus qui ont débuté et ceux qui sont terminés, pour chercher les données satellitaires disponibles et pour créer des chaînes de traitement de façon interactive. Une interface pour récupérer et traiter les données archivées a été intégrée; l’utilisateur peut chercher de façon interactive les données satellitaires archivées et les traiter avec les services de traitement et les flux des opérations développés. Le SIG sur le Web agit également comme client WPS, client pour visualiser l’auteur et les données traitées de même que la couche de sécurité pour les chaînes de traitement. Ce travail de recherche et de développement indique qu’il faut effectuer un travail supplémentaire pour améliorer les spécifications du WPS, par exemple en ajoutant des profils propres au domaine afin de travailler avec de grands ensembles de données matricielles dans le but de définir des règles normalisées pour l’échange de données entre les processus. Dans le domaine de l’orchestration de processus, il n’existe pas encore de format de données normalisé décrivant les chaînes de traitement.

1. Introduction

Since spatial data are distributed in the web with specifications like Web Map Service (WMS) [OGC 2002], Web Feature Service (WFS) [OGC 2005a] and Web Coverage Service (WCS) [OGC 2005b; 2007a], it is a further step to process these data. Web-based processing of spatial data is more and more on the way since the Open Geospatial Consortium (OGC) published a specification to handle web-based processing, the Web Processing Service (WPS) [OGC 2007b]. This new possibility in web-based information systems allows an interoperable processing of data based on service-oriented architectures.
The main objective of this work was to analyse possibilities of processing remote sensing data with standard compliant web-based services in a web-based geographic information system. Focusing on raster data, especially remote sensing data, issues appear which still have to be investigated like data transfer and exchange mechanisms between processing services as well as performance issues concerning the handling of large data amounts. Therefore the aim of this paper is to research, what requirements for web-based processing of remote sensing data do exist, whether they can be achieved with actual standards, how the combination of OGC standards and processing of remote sensing data can be realised and what requirements need to be solved for the web-based client.

To achieve these aims and answer these questions near real-time and archived remote sensing data from the NASA MODIS sensor was processed. Information like fire hotspots [Justice et al. 2006] and land surface temperature [Wan 1999] was derived and published within a web-based information system. This information system acts as a client for the manual execution and automatic monitoring of available and started processes, an editor for process chains and as an OGC WPS proxy for the provision of created process chains.

The following section shows literature with the usage of OGC WPS and chaining processes based on WPS. The third chapter “Design” describes the defined system architecture and principal methods like service-oriented architectures, the OGC WPS specification and possibilities to orchestrate process services. The technical implementation is shown in chapter four containing used software of the infrastructure to provide web-based processing, process orchestration and the developed web-client with its components. The implementation is then tested with a case study (chapter five), where MODIS data, their products and the developed processes and process chains are described. Finally the scientific findings are summarized and discussed.

2. Literature

Literature work was done in the field of web-based geoprocessing and service-based chaining. Within geoprocessing our main interest was how to bring raster data and algorithms together and how they can be exchanged and processed. For chaining services the main focus was on the chaining of processes to generate processing workflows.

Web-based Processing

Already before the final release of the OGC WPS specification, scientists developed systems for web-based processing for example for the generalization of vector data [Foerster and Stoter 2006] and the management of water resource [Diaz et al. 2007]. First requirements from the users to chain processes and further improvements like “sufficient communication mechanisms” and “semantically enriched description of processes” [Foerster and Stoter 2006] were published. Even today the semantics play an important role within research of web-based processing systems [Foerster 2010; Janowicz 2010; Kiehle et al. 2007; Owonobi and Baumann 2009; Xu 2009].

After the publication of the WPS specification as version 1.0.0 [OGC 2007b] Friis-Christensen et al. [2007] reported that the performance of a WPS must be improved when transporting data to the processes. Granell et al. [2007] were engaged in the processing of remote sensing data and showed that the main problem is the transport and the processing of large data amounts. They proposed a registration mechanism of available processes in public catalogues and the use of loosely coupled data and algorithms to overcome the problem with data exchange between data servers and processing servers. Both, Friis-Christensen et al. [2007] and Granell et al. [2007], had the idea to bring the algorithms to the data to limit the data transfer between different servers. Michael and Ames [2007] showed the necessity to implement a mechanism that lists all available data on the processing server to act as data repository. With such a data repository a user can select possible input data or just can make use of already available data.

In 2009 Brauner et al. published a “research agenda for geoprocessing services” and defined three critical areas:

1. Service orchestration
2. Semantic descriptions
3. Strategies to improve performance

The aim of the service orchestration is to allow value added chains as well as the creation of new business models for a Spatial Data Infrastructure (SDI) based on external processing tasks. The limiting factor of the fully automatic orchestration is the missing semantic descriptions of each process. Strategies to improve performance are in demand, for example by using GRID infrastructures or cloud computing [Brauner et al. 2009].

Further, prospective research has to be done for example in the areas of WPS profiles to have standardized descriptions of processes with similar functionalities or domain-specific processes [Nash 2008], the distribution of processes within grid-enabled systems [Baramski 2008] and the integration of the Web Service Resource Framework (WSRF) to
avoid redundant data transport through unique data references [Keens 2006]. Even though there is many research around WPS, the described ideas and requirements showed that the WPS specification has to be further developed. An OGC standards working group working for version 2.0 was established to concentrate on the further development of this specification and to consider ideas like request cancelation, application profiles and more [OGC 2012].

Chaining WPS Processes

Alameh [2003], Einspanier et al. [2003] and Peltz [2003] worked in the field of chaining geo-web-services in general and described basics, possible concepts and technical realisations. Consistently the OASIS-standard “Business Process Execution Language” (BPEL) is mentioned to chain services [Alameh 2003; Einspanier et al. 2003; Peltz 2003]. BPEL describes a standard to chain web-services based on the Web Services Description Language (WSDL). This standard is also mentioned within the OGC WPS specification [OGC 2007b] on how to chain processes to provide process workflows. Kiehle et al. [2006] as well as Lemmens et al. [2007] discussed similar approaches and methods to chain processes.

In many projects related to geoprocessing BPEL is used, but problems with SOAP compatibility within OGC-standards were always mentioned [Weiser and Zipf 2007; Schaeffer 2008]. However, they propose to use BPEL because complex workflows can be implemented. Schaeffer [2008] implemented a transactional interface (WPS-T) to insert and execute workflows based on BPEL dynamically. Friis-Christensen et al. [2007] compared three possible methods for service chaining (transparent, semi-transparent and opaque) and conclude, that the semi-transparent chaining in combination with BPEL is the best way. Also Bernard et al. [2003] described the semi-transparent chaining as “the way forward”.

In contrast to BPEL, Stollberg and Zipf [2007; 2008] proposed to use a WPS process itself for chaining processes, because WPS processes do not provide WSDL descriptions. To solve this problem, Brauner and Schaeffer [2008] developed a XML transformation to create WSDL documents providing SOAP compatibility for WPS; Sancho-Jimenez et al. [2008] implemented a proxy to provide WSDL documents for WPS processes.

So there are many approaches to chain processes with BPEL. But for software solutions which implement a WPS process itself for chaining WPS processes, there is current no standardised way to describe these process chains, so that a WPS can execute these workflows. Already Foerster and Stoter [2006] remarked that consistent rules for process chaining are missing.

3. Design

A system architecture and the components for each service were defined to build-up a web-based processing system. For the interaction between these service rules for data exchange and data management were specified. In the following section the designed system architecture, service specifications for web-based geoprocessing and possibilities to orchestrate processes are described.

3.1 System Architecture

The system architecture utilized contains three parts: the data layer, the service layer and the client layer (Figure 1). The data layer describes the different data sources: the external remote sensing data archive, the local data for processing and the geodatabase with products created for publishing. A catalogue service interface [OGC, 2007c] establishes a connection to the metadata catalogue of the archive. Processing and data access services are communicating with the data layer. The central component is a WPS, which communicates with the data archive and the local WPS data directory to process this data and publishes the finally derived products in the geodatabase. In the “Local data” directories files created by the executed process are stored. A service for executing processing chains communicates with the WPS and the geodata access service communicates with the geodatabase. These services provide interfaces like OGC WMS, WFS, WCS and WPS, so that a client can work with these data using OGC compliant services. These services can then be implemented in a Web-GIS or GIS.

Figure 1. Technical schema of the implemented system.
Desktop software like e.g. uDig [uDig 2012], ESRI ArcMap [ESRI 2012], Quantum GIS [QGIS 2012], gvSIG [gvSIG Association 2011].

A closer look to the data layer shows that the data management between several systems must also be defined (Figure 2). Data from the remote sensing data archive of the German Aerospace Center is provided within the “Data Information and Management System” (DIMS) which provides an interface to communicate with the archive (Processing System Management (PSM)). Requesting archive data, the PSM pulls the data to a registered server using the File Transfer Protocol (FTP). A WPS process transfers this data to the local processing directory, another process calls external executables or GRASS modules to process the data. The processing server is divided into several output directories for different processing steps. Finally the outputs are stored as linked files in the public web server directory. Final outputs from automatic process chains are stored in addition to the specified geodatabase.

Any process can specify a processing directory where its local data is stored. This local directory is linked from the web server for direct referencing the output data of the process. This mechanism limits the transfer of data within process chains from one process to the next.

3.2 Web-Services

Nowadays web-based information systems are based on service-oriented architectures. Services can be implemented by SOAP (“Simple Object Access Protocol”) or by using HTTP REST (“Representational State Transfer”) specification. Services from OGC are commonly using a REST-like specification with plain XML and HTTP-GET using KVP encoding or HTTP-POST using XML encoding to transfer data and information. Additionally, some specifications from OGC cannot be used with SOAP. For these reasons the usage of HTTP-GET and HTTP-POST was preferred in the following developments.

OGC Services

For web-based processing OGC specifies the Web Processing Service [OGC 2007b]. This specification describes an interface for publishing processes on the web. WPS provides three methods: “GetCapabilities”, “DescribeProcess” and “Execute” [OGC 2007b]. “GetCapabilities” lists all published processes, “DescribeProcess” lists further details (e.g. inputs and outputs) of a given process and “Execute” begins the execution of a process with given input data. The user can decide whether the process will be executed synchronously or asynchronously. An asynchronous execution means that the user gets a response after execution immediately with the information, that the process was started and an HTTP URL for retrieving status updates. The process runs then in the background and the user can obtain the status of the executed process by the given URL. A process contains several inputs and outputs with a defined quantity and format. It can be distinguished between literal, complex and bounding box data. Literal data are strings or numbers; complex data are for example binary or text files like XML which can be referenced by URL or included in the response [OGC 2007b].

Process Orchestration

Complex processing can be realised by chaining simple processes. In process chains these processes can then be executed parallel or one after another; processes can depend on prior processes. As the processes are based on services, we describe service chaining first. The International Organization of Standardisation (ISO) defines a service chain within ISO 19119—Geographic information—Services [2005, p. 6] as “sequence of services, where, for each adjacent pair of services, occurrence of the first action is necessary of the second action”. Three different types of chaining are mentioned there:

1. Transparent or user defined chaining
2. Semi-transparent or workflow-managed chaining
3. Opaque or aggregate service chaining

In transparent chaining the user calls each process one after another. Using the semi-transparent chaining the processes were linked to one workflow, but the user knows which processes are inside the

Figure 2. Technical schema of data management within the WPS.
workflow. With the opaque chaining the user does not know which processes are called. The workflow runs in the background.

Beside ISO 19119 the OGC WPS specification defines three solutions to chain processes [OGC 2007b]:

1. Usage of the BPEL engine
2. WPS process as orchestration engine
3. Simple service-chaining within execute query

The third possibility is only for simple chaining of OGC-services. For complex chaining, BPEL [OASIS 2007], the WPS process or other software solutions has to be used. BPEL is an IT standard to chain complex processes based on SOAP with an extension for HTTP-GET/POST services. The second possibility can be solved with any software integrating the WPS standard or software only for chaining processes as a process following the OGC WPS specification.

4. Implementation

4.1 Web Processing Service

To provide OGC WPS, the Python-based software PyWPS [Cepicky and Becchi 2007; Cepicky 2009] was used and was implemented using Apache Webserver software [The Apache Software Foundation 2012]. After installing PyWPS system-wide, it is possible to create several instances for example to provide a service for public availability and one for restricted administration and dissemination tasks.

In PyWPS a process is defined in one python file, which is available after linking it to an instance of the WPS. The need of compilation does not exist in the process keeping development simple. The processing of input data is dependent on the computation implemented in the process and is therefore flexible. Extensions and own developments can be integrated by changes within the source code or the usage of Python packages. The process is developed by writing a process class with an initial and an executing method (Figure 3). The initial method describes the main properties of this process and defines the inputs and the outputs. The executing method accesses the information from input data, runs the process algorithm and adds values of the output data. Python modules like the integration of the Geospatial Data Abstraction Library [Python Software Foundation 2011] are available to work with geodata. Furthermore modules for connecting to databases, executing programs from command-line or accessing and editing special file formats can be used.

For this implementation some customization were made to improve performance when working with large data: To reference output data directly to the processing directory, the source code was changed, so that output data can be referenced directly by a defined URL. Changing the value of an output data to this URL, the data will not be copied to the web servers output directory. Another function was integrated allowing a process to define which output has to be a reference data value. Normally the

```python
from pywps.Process import WPSProcess
class Process(WPSProcess):
    def __init__(self):
        # process definition
        WPSProcess.__init__(self,
            identifier = "modis_mod14", title="process modis mod14 fire algorithm",
            version = "0.1", storeSupported = "true", statusSupported = "true",
            abstract="Executing MOD14 algorithm on MODIS L1B Data (extracting fire points")
        )

        # inputs
        self.in_l1bLkm = self.addComplexInput(identifier="data_l1bLkm", minOccurs=0,
                                               title="MODIS Level 1B 1KM", formats=["application/x-hdf"], maxOccurs=1)

        # outputs
        self.out_mod14gml = self.addComplexOutput(identifier="data_mod14gml", title="MODIS MOD14 GML",
                                                    formats=["application/xml", "schema": http://schemas.opengis.net/gml/3.2.1/gml.xsd"])

        def execute(self):
            # getting inputs
            l1bLkm = self.in_l1bLkm.getValue()

            # process inputs to outputs
            # execution of command-line MOD14 algorithm

            # setting outputs
            self.out_mod14gml.setValue("MOD14.gml")
```

Figure 3. Structure of a process implemented in PyWPS.
user defines whether output data comes back as reference URLs or encoded within the response. This extension leads to the possibility that the process can decide which outputs, normally large files, have to be referenced and should not be integrated as binary data in the XML-based output.

4.2 Orchestration Engines

Several steps are necessary to derive specific products based on remote sensing MODIS data like fire hotspot data or land surface temperature. For these specific products the first steps are always the same when processing from unprocessed raw MODIS data. For this reason every single step was developed in one process. To chain these processes, software is needed to execute this workflow. An additional demand for this software is the feature, that users should create workflows on the web by linking process outputs and inputs dynamically and publish this workflow as a normal WPS process. Three software solutions were tested: 52° North WPS Orchestration, Apache ODE with BPEL orchestration and the internal software DIMS-WPS from the German Aerospace Center.

52° North WPS Orchestration [52° North Initiative 2010] is based on the 52° North WPS. A special process is available for handling workflows calling other WPS processes; the process chain is assigned by an input parameter. Executing a workflow, each single process is started with the property that outputs are integrated in the returned XML code. Therefore, all linked data between processes are integrated in the requests and responses, data could not be transferred from one process to the next using references to output files. For this reason large raster data lead to memory problems parsing large XML code. Workflows for this orchestration engine can be developed using the available Java API or the provided web-application based on Google Web Toolkit. But both solutions could not easily be integrated in our web-based information system developed without Java.

Another workflow solution is the orchestration with Apache ODE [The Apache Software Foundation 2010] which follows the BPEL standard. BPEL is a XML based language where any kind of process can be executed e.g. within sequences, loops or if-conditions. The workflow is defined in XML files in which the calls to the processes can be integrated and the responses are parsed individually. In context of providing workflows for WPS there is no special linking beside a HTTP POST interface to call these services. Every WPS has to be defined as a HTTP binding and the workflow process itself as SOAP binding. All these steps make the creation of a workflow very complex. Providing these BPEL workflows as WPS processes, additional software around Apache ODE is needed.

Our third possibility to chain processes is the usage of internal software from the German Aerospace Center (DLR), called DIMS-WPS [Werum AG 2010]. This internal closed-source software was developed for the DLR based on technologies integrated previously for the existing remote sensing data archiving system. It provides a WPS interface for executing workflows and defines its own XML based language for describing these. Figure 4 shows this XML with the workflow

```xml
<?xml version="1.0" encoding="UTF-8"?>
<wpsw:wps-workflows>
  <wpsw:workflow identifier="modis_chain_mod14">  
    <wpsw:sequence>
      <wpsw:step identifier="modis_l1b" successor="http://pyros/pywps.cgi">
        <wpsw:input identifier="l0_url" globalInput="true"/>
      </wpsw:step>
      <wpsw:step identifier="modis_mod14" successor="http://pyros/pywps.cgi">
        <wpsw:input identifier="data_l1b1km" sourceStep="modis_l1b" sourceIdentifier="out_l1b1km"/>
        <wpsw:input identifier="data_l1bgeo" sourceStep="modis_l1b" sourceIdentifier="out_l1bgeo"/>
      </wpsw:step>
    </wpsw:sequence>
    <wpsw:outputs>
      <wpsw:output identifier="data_l1bgeo" sourceIdentifier="out_l1bgeo" sourceStep="modis_l1b" asReference="true"/>
      <wpsw:output identifier="data_l1b1km" sourceIdentifier="out_l1b1km" sourceStep="modis_l1b" asReference="true"/>
      <wpsw:output identifier="data_mod14gml" sourceIdentifier="out_mod14gml" sourceStep="modis_mod14" asReference="true"/>
      <wpsw:output identifier="data_noFires" sourceIdentifier="out_noFires" sourceStep="modis_mod14"/>
    </wpsw:outputs>
  </wpsw:workflow>
</wpsw:wps-workflows>
```

Figure 4. XML based definition of workflow for DIMS-WPS.
process identifier, two WPS calls (wpsw:step inside of wpsw:sequence) and the definition of the outputs (wpsw:outputs). Every workflow is stored as one XML file in a defined directory of this orchestration engine and latter appears as a process within the WPS.

For providing an orchestration service the DIMS-WPS was chosen because the 52° North Orchestration service could not work with large raster data leading to memory problems and our external non Java-based client was not able to define workflows easily. Apache ODE with BPEL was too complex to provide a real WPS as an orchestration service. With DIMS-WPS the shown XML code (Figure 4) could be generated easily to chain individual processes. Further, a proxy for DIMS-WPS was implemented in Drupal CMS to provide a security-interface for executing integrated workflows. These workflows were created by the developed web-application within Drupal CMS generating the workflow XML file.

Other workflow software like VisTrails [e.g. Callahan et al. 2006] with an extension for Earth Observation eo4vistrails [Van Zyl et al. 2011] or Kepler workflow system [e.g. Altintas et al. 2004; Ludäscher et al. 2006] do exist, but there were no interfaces to execute implemented workflows as OGC-compliant WPS.

4.3 Geodata Access Service

To visualise the created products a WMS is needed. The software GeoServer [OpenGeo 2011] has an API based on HTTP REST, which was used to add new layers on GeoServer. It allows the client to transfer the generated output data (vector or raster data) to GeoServer and display it on the fly in a map. Using this interface spatial output data can be displayed directly after processing in the web-based processing client.

4.4 Drupal Content Management System (CMS)

Drupal [2011] is a web content management system and a web application framework with a modular design which can be extended by a large number of modules. This framework provides the option to organize structured content in several ways with a sophisticated user authorisation mechanism. Acting as a backend tool, different content-types for handling WPS servers, WPS execute requests and processing chains were integrated. Any content is assigned to the user who created it, so that independent users can create workflows only for their own and finished processes appears only in the users own list of processes.

4.4.1 Process Management and Monitoring

At the start of a process execution from the developed information system, the WPS execution request is registered and stored as content with the specific content-type for process execution. Figure 5 shows the complete sequence from executing a process until the user information for a finished process. To register the running process as content in Drupal CMS, a proxy for executing WPS processes was implemented. The proxy provides an endpoint (e.g. http://pyros/modisportal/wps/insert) and parses the execute request, which is redirected from the developed web portal, inserts a new content object within Drupal CMS and forwards the request without changes to the passed WPS endpoint. To check the status of each running process regularly, a script within Drupal CMS passes through all registered and running processes and fetches the status URL every minute. The retrieving status document contains further information about the running process (e.g. which task is actually processed). The new status information will then be updated in the Drupal CMS content object. To protect the writing mechanism (insert and update) the script logs in to Drupal CMS via HTTP authorisation, which was configured only for these services.

If execution is finished or an error occurred, Drupal CMS starts an internal function. The handling of this function can be defined individually within the Drupal CMS, for example sending a mail to a defined mail address. With this mechanism it is possible, that users are informed when processing

![Figure 5. Sequence diagram between users executing a process, Drupal CMS as process management system and OGC WPS server where the process is running.](image-url)
is finished. Then the user can login back in the system to visualize or download the outputs. Within Drupal CMS any executed process will last until the user manually deletes it. Deleting an executed process within Drupal CMS, another process is executed directly within our WPS removing the output files from the processing server to clean up the output directories.

4.4.2 Processing Chains

Based on DIMS-WPS Orchestration the process chains are described in XML documents. To create such chains in the web-based information system an interface between information system and DIMS-WPS was developed. To link processes the JavaScript library jsPlumb [jsPlumb 2011] was integrated. In an editor window (Figure 6) it is possible to add several processes and bind output data to input data. Furthermore, some input data can be selected for global inputs so that the user can declare this input by starting this new opaque process. After saving a new workflow it acts as a normal process. Every new workflow is stored in Drupal CMS and transferred to DIMS-WPS over an administration process on the same server. With the individual rights management within Drupal CMS it is possible to assign every process chain only to registered users with respective rights, therefore Drupal CMS acts as a proxy for DIMS-WPS, which is then only accessible through the proxy.

4.5 Web-Client

The web-based information system acts as a client for the developed processing system. The defined requirements indicate a map to visualise satellite data and products, a search component for archived satellite data, a tool to monitor running processes as well as the creation of workflows for product generation. For this information system open-source software was used. The frontend was developed with JavaScript libraries like ExtJS [Sencha 2011], GeoExt [GeoExt 2011] and Openlayers [Openlayers 2011]. ExtJS was used to ensure interactive web frontend components, OpenLayers as mapping framework and GeoExt to bring OpenLayers and ExtJS components together. The backend was built on the framework Drupal CMS.

The Web-GIS is structured in three parts (Figure 7): the header containing the main navigation tools, the left part with the metadata catalogue search and the processing system and the middle part, where the interactive map or other content is displayed. The map is based on OpenLayers and GeoExt functions. The components in the left part are based on ExtJS and GeoExt. To display content from Drupal CMS within the interactive frontend, a module called “ext_json” was used to request content items and special views on specific content types. With simple Ajax calls the ExtJS components could be filled with their content based on the open standard “JavaScript Object Notation” (JSON) that is designed for interchanging simple data structures.

4.5.1 Data Search

The search for satellite data which can be processed is based on a standard-compliant OGC CSW interface with SOAP binding. This interface is provided by the German Remote Sensing Data Center within the German Aerospace Center. A client was developed to submit the search form to the CSW interface and transfers the response to the user interface. A “GetRecords” request integrated in a SOAP envelope returns a “GetRecordsResponse” and lists available remote sensing data along with specific information fields. An own Drupal module was developed to use the search parameters and to return a GeoJSON object for further processing within the frontend. This GeoJSON object contains
the bounding box of the data and metadata information. With the GeoExt “FeatureStore” component it is possible to visualize the bounding box in the GeoExt map automatically after loading the GeoJSON object (Figure 8). Based on the dataset id given within the metadata it is possible for a WPS process to request the data itself for further processing. Figure 8 shows the search form and the response items in the left part and the bounding boxes in the map. Paging was integrated for the list of items to foster clarity.

4.5.2 Processing Services

A client with ExtJS components was developed to start WPS processes registered in Drupal CMS and to request their status. A list of processes is generated by retrieving and parsing the “getCapabilities” response of each WPS. After selecting a process the “DescribeProcess” request is executed. From the given information the process is graphically displayed and input-fields are generated (Figure 9). This development is based on a WPS extension [PyWPS 2011] for OpenLayers published by PyWPS.

The running and completed processes are listed in an ExtJS-based Gridpanel. For each process it is possible to request the actual status information with their input and output data (Figure 10). A direct connection to the open-source software GeoServer allows an instant visualisation of the vector- and raster-based output data.

5. Case Study

To evaluate the developed web-based processing and information system, data from MODIS sensor was selected due to its defined product algorithms and high availability. The German Aerospace Center (DLR) owns a satellite data archive with data from MODIS sensor over Europe. Other MODIS data from all over the world is freely available from NASA servers. Also, NASA provides software components to process these data from unprocessed level to final information (Table 1).

5.1 Data and Products

At the DLR, the data are available as unprocessed files, so called “Level 0” files, from the satellites Aqua and Terra. With the software SeaDAS [NASA 2011c] from NASA it is possible to create pre-processed files (“Level 1B”). From this processing level, specific remote sensing products can be created, for example surface reflectance data with corrected reflectances (software “CREFL_SPA”), fire hotspot points (software “MOD14_SPA”), or land surface temperature data (software “MODLST_SPA”). These software components base upon published algorithms [NASA 2011a] for NASA MODIS data and can be executed from command-line. The output data of NASA's software components are data in form of the Hierarchical Data Format (HDF). This is used from NASA as output data format of their Earth Observation System. The software component “HDF2GeoTIFF” [NASA 2010] was used to transfer these HDF files to GeoTIFF files for further processing and visualisation. A WPS process covers each of this software to have the possibility to chain these processes to individual workflows.
There are several products that can be derived from MODIS data in the fields of atmosphere, cryosphere, land and ocean products [Salomonson et al. 2006]. The main goal of this study was to publish fire hotspot points derived from near real-time MODIS data. To show the flexibility of the developed system, the derivation of land surface temperature and corrected surface reflectance were also implemented.

### 5.2 Processing Services

The developed processes were parted into several groups (Figure 11): Initialization, Data acquisition, Pre-processing, Product derivation, Post processing, Dissemination and Administration. Processes of groups one to five are accessible from the public whereas the processes within groups six and seven run on a separate WPS instance, which is protected for the usage from defined clients only. For executing external software the Python module “subprocess” was used. With “subprocess” it is possible to execute commands on the command-line from the operating system. The process can wait until the command is finished and then continue working with the files generated by the external software.

In any process there is an optional feature declaring a processing directory as an additional input. In that directory all processed files are stored. It replaces the temporary processing directory of PyWPS and is created by a separate process (“init_get_processdir”). This processing directory offers the possibility to access the processed data from another process at a later time and to directly reference the output data without copying it to the servers, thus relieving the server from time-consuming copying of large amounts of data. Declaring one processing directory for the whole workflow further provides the possibility to create a log file with information of each process called in the workflow. After finishing a process chain the operations of the workflow can be followed by viewing this log file.

To download the unprocessed MODIS data either from the DLR archive system or the NASA servers by an HTTP URL, the process “modis_download” was developed. If a metadata ID is given as input, the process requests this specific data from the DLR archive, or, if a HTTP URL is given, the data from the URL is downloaded. As output, an unprocessed MODIS file downloaded either from the archive system or the NASA servers is available for further processing.

Starting from unprocessed MODIS data the first step is to call the “modis_l1b” process. This process executes the SeaDAS software, which generates a georectified “Level 1B” product file. To extract further information, such as fire hotspot points (process “modis_mod14”) or land surface temperature (process “modis_lst”), individual algorithms are available as executable software (Table 1) that are executed in the processes. For further processing the product data can be converted into GeoTIFF format (raster based) or if vector data, from GML to KML (“convert_gml_to_kml”) or Shapefile format (“convert_gml_to_shp”) to directly provide downloadable files. The open-source software GRASS GIS was implemented for the intersection of fire points with landcover and country border information by using the command-line interface of GRASS GIS (process “info_points_grass”). A temporary GRASS location is created and GRASS commands can be executed in this location within the process. Before using GRASS some properties to the installation

<table>
<thead>
<tr>
<th>Command-line programs</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>MODISL1DB_SPA (includes SeaDAS)</td>
<td>Convert unprocessed sensor data (as Production Data Set (PDS) format, “Level 0”) to “Level 1B” product data.</td>
</tr>
<tr>
<td>CREFL_SPA</td>
<td>Program to correct reflectances to create surface reflectance data to provide different band composites (HDF files).</td>
</tr>
<tr>
<td>MOD14_SPA</td>
<td>Extract fire hotspot points from processed “Level 1B” MODIS data HDF file.</td>
</tr>
<tr>
<td>MODLST_SPA</td>
<td>Extract land surface temperature data from processed “Level 1B” MODIS data as HDF file.</td>
</tr>
<tr>
<td>HDF2GeoTIFF (H2G_SPA)</td>
<td>Convert MODIS products (e.g. CREFL, MOD14, LST) from their output file format HDF to GeoTIFF.</td>
</tr>
</tbody>
</table>

**Figure 11.** Developed processes in different categories.
have to be declared in the configuration file of the PyWPS instance. The programming language Interactive Data Language (IDL); [Exelis 2011], often used to analyse remote sensing data, was integrated for the generation of “quicklooks” (“modis_l1b_crefl_quicklooks”) using the execution by command-line interface.

The processes for “dissemination” are only used in the automatically running workflow on near real-time data, not on manually executed workflows in the Web-GIS client. The process “publishData” helps to publish data to a website and the processes “storeModis” and “storeFires” store data to a geospatial database. The processes for “administration” are called from the web-based information system to administrate the orchestration engine (“dimsWPSAdmin”) and the WPS, for example to remove output data from specific processes (“cleanupData”).

5.3 Processing Chains

For creating remote sensing products, several steps have to be performed to achieve the final products. Most of these steps—each one implemented as a separate process service—are used in different process chains; thus only a few product-specific processes have to be changed to create another workflow to derive a new product. Figure 12 shows an example workflow to derive fire hotspot points. The input and output data of this workflow is specified on execution of the workflow. Every workflow can work with internal data from the data archive system or with external data referenced by a HTTP URL. The final output of this workflow are: a GeoTIFF file visualising the satellite scene, a GML and a Shapefile representing the fire points as well as a GeoTIFF file showing the classification mask generated for detecting fire points. Similar workflows were generated for the derivation of land surface temperature, for the visualisation of the given MODIS satellite scene and for the fire-service, automatically running on near real-time data. For this service, the derived fire hotspot points are deployed into a PostgreSQL/PostGIS database and the output data is published to an external web server for website integration.

The workflow editor developed and implemented into the Drupal CMS provides the opportunity to edit or create new workflows. These workflows can execute processes from different WPS. If workflows are published, they are then automatically transferred and integrated in the DIMS-WPS software, which controls the execution of the individual processes within the workflow.

6. Conclusion and Discussion

In this study requirements for web-based processing of remote sensing data were identified and evaluated based on the implementation of a processing system. This web-based information system was developed at the German Aerospace Center to derive products based on MODIS satellite data, integrate own processing workflows, monitor started processes and visualise the output data. Based on the literature work the potential to provide web-based processing services with workflows is very visible. But also issues in the processing of large data are present and research have to be enlarged in this areas. Furthermore the chaining of processes is not standardised and descriptions of processes are not yet clearly structured and defined.

In Earth observation typically large data quantities are involved. In such cases, Granell et al. [2007] suggest to “bring the algorithms to the data”. Although a very reasonable idea for centralized web-based processing services, this concepts stands in contrast to the basic idea of SDI, where data is distributed over several systems. For the developed system, the processed data were already available on the same network, so that the algorithms to process these data were implemented on the right place, near the data. This makes sense if the data provider not only wants to provide data, but also to provide derived products, and if data and products are provided for near real-time applications. Processing external data leads to a first download of the input data to the processing server, which increases data transfer amount and processing time. But if the data provider is not providing the product in a format that an application or a user needs, this additional step is essential and data...
have to be transferred to the processing server. The idea from Michael and Ames [2007] to implement a mechanism to list all available data from the processing server leads to an efficient data management. This is important for processing workflows with separate processes because data must be exchanged a lot between processes. Other data repeatedly used as input data for processes, like a digital elevation model, can be placed at the server and reused as input data. Working with large data the management of data used for processing has to be clearly defined. This was a main aspect of this work and therefore centralized processing directories as well as the corresponding program logic were established in the processing system. These mechanisms, for example that the process can define whether outputs have to be returned as URL references rather than binary in the XML code, should be integrated in a special WPS profile for processing remote sensing data. Already Nash [2008] proposed the usage of WPS profiles for specific domains.

For the development of a web-based processing system PyWPS as a possible OGC-compliant WPS was used and Orchestration Engines from 52° North, Apache ODE and an internal software were tested concerning the processing of remote sensing data. Already in the year 2006 Foerster and Stoter remarked that there is no standardised way to chain processes. For complex workflows BPEL can be used, but for easier workflows other standardised solutions have to be developed. The possibility to define process chains with a XML based scheme is obvious. A WPS only has to parse this XML file and execute the specified processes. Such an example XML code was used in the internal software DIMS-WPS to provide the orchestration engine. This XML file can be generated easily with own software, like the developed workflow editor within Drupal CMS.

Another part of this work was the implementation of a web-based geographic information system. Using Drupal CMS as a backend and frontend framework showed the possibilities of a user-dependent information system. Users can store and list their started and finished processes and create their own workflows. WPS and workflows can be provided user-dependent and additional security authorisations can be stored in the CMS. Another possibility is the use of Drupal CMS as WPS proxy so that security limits can be controlled using the rights management of the CMS and user-dependent context like context information of maps or executed processes and their final outputs can be stored there.

But all the possibilities of web-based processing within a web-based geographic information system cannot be used when no data or algorithms for processing are available. Beside manual start-up of processes there should be also algorithms that run automatically to derive information products from the given input data. Having near real-time data additional advantages for example for crisis information are available by automatically processing products and providing official services like fire hotspot points or flood areas. For data from MODIS sensors a couple of algorithms from NASA and near real-time data for Europe from the German Aerospace Center are available. Better spatial resolutions for Europe will come with the Sentinel Family from the European Space Agency, whose data are planned to be free of charge [Berger and Aschenbacher 2012]. Those data can then again be integrated in own applications and services without further restrictions on data side. If data are available for free, no boundaries are set for the development of applications, services and business cases providing further information products based on remote sensing data.

**Acknowledgements**

This paper is based on a master thesis written at the German Remote Sensing Data Center within the German Aerospace Center (DLR) Oberpfaffenhofen in cooperation with Department of Earth Observation within Friedrich-Schiller-University (FSU) Jena. I thank my supervisors, Prof. Dr. Christiane Schmullius (FSU) and Dr. Christian Strobl (DLR), and their institutes for supporting this work.

**References**


Authors

Jonas Eberle is working as research assistant and PhD student at the University of Jena, Germany, at the department of Earth Observation. Within his research project, the “Siberian Earth System Science Cluster”, he focuses on automatic web-based provision and processing of remote sensing data and derived products and the development of web-based geographic information systems with open source software. Jonas holds a Master of Science in Geoinformatics with focus on Earth Observation from the University of Jena and a Bachelor of Science in Applied Informatics from the University of Applied Sciences Ostwestfalen, Hoexter, Germany.

Christian Strobl is working as GIS Scientist at the German Aerospace Center in Oberpfaffenhofen, Germany. The last years his major task was the development of the geospatial infrastructure for the German Indonesian Tsunami Early Warning System (GITEWS). He focuses on processing geoscientific and remote sensing data with open source tools. Christian holds a Ph.D. in Geology at the Munich Ludwig-Maximilians University where he also teaches GIS. In addition he holds lectures about Geostatistics at the Technical University Munich. ❑