52north
exploring horizons

Web Processing: Standardization and Cloud Concepts

Matthes Rieke, Benjamin Proß

Geospatial Sensing | Virtual 2020

Overview and Agenda

1. Web-Processing
2. Standardization / Interoperability
   a. WPS 2.0
   b. OGC API Processes
3. Software Solutions
   a. javaPS
   b. wps-js-ng
   c. Angular Map Client
4. “Hands-on” / Use Case
   a. Docker-based Process Execution

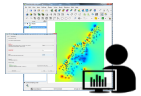
Why Web-based Geoprocessing and WPS?

Motivation and Background

Main focus of 52°North

Sensor Web Enablement

Geoprocessing



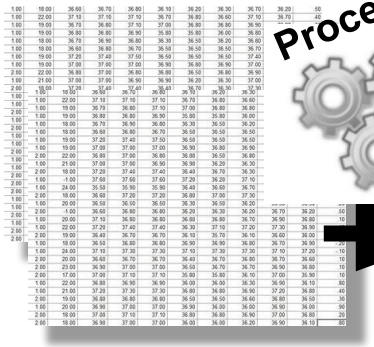
Desktop Apps

Web Apps

SDIs, SOA, Big Data

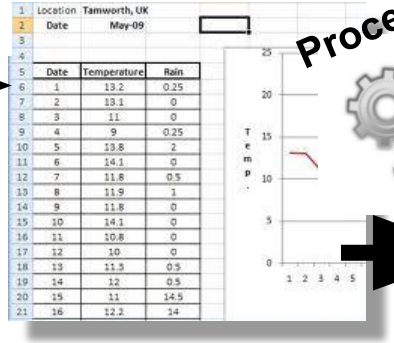
Geoprocessing

Raw data



Processing

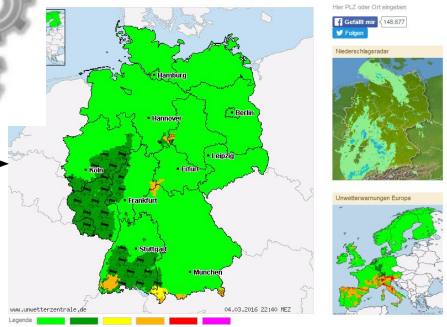
Value-added data products



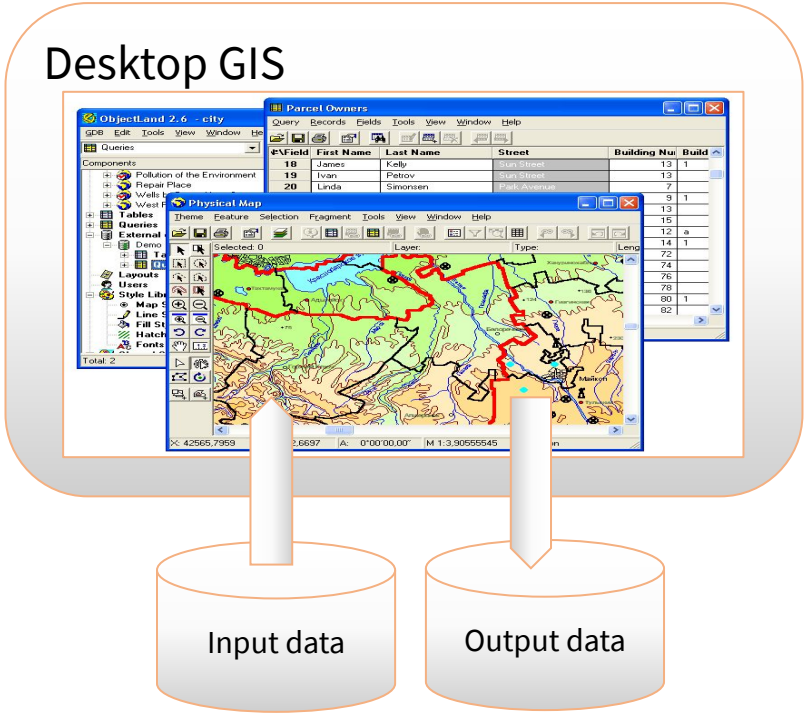
Location	Tamworth, UK	
Date	May-09	
Date	Temperature	Rain
1	13.2	0.25
2	13.1	0
3	11	0
4	9	0.25
5	13.8	2
6	14.1	0
7	11.8	0.5
8	11.9	1
9	11.8	0
10	14.1	0
11	10.8	0
12	10	0
13	11.5	0.5
14	12	0.5
15	11	14.5
16	12.2	14

Processing

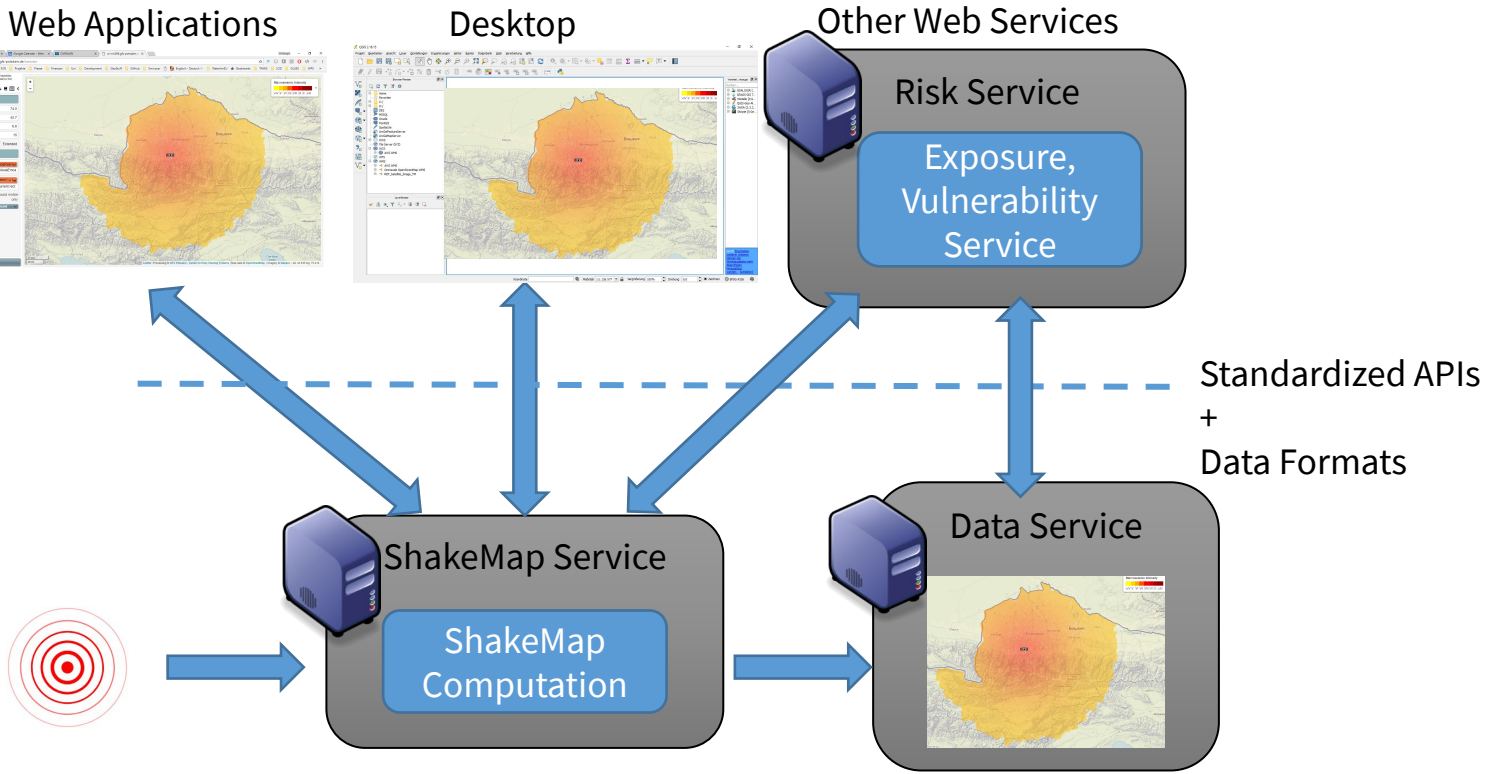
Information products



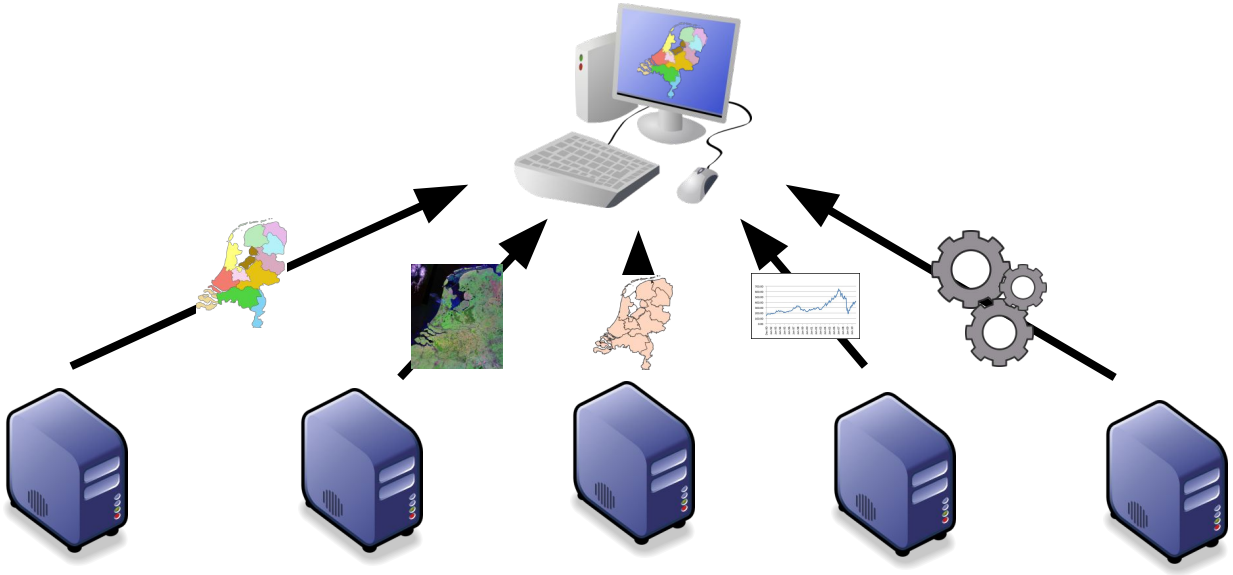
Geoprocessing – Earlier Approach



Web Service - Approach



OGC Web Processing Service (WPS)



WMS – Maps as Images (jpg, tiff, …)

WCS – Coverages (geoTiff, netCDF, …)

WFS – Vector Data (GML, shp)

SOS – Observations (O&M, SweCommon,…)

WPS – Geoprocesses, Simulations, …

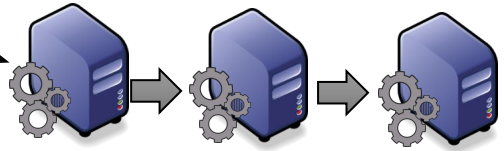
Geoprocessing in the Web, because…

…the set-up of software is complicated

…I lack the computational capacity in my own environment

…I want to reuse processing in different environments

…processing is tightly coupled to data that is available only remotely



…I want to couple different technologies („Processing Mash-Ups“).
…the analysis should be automated and controllable via the web

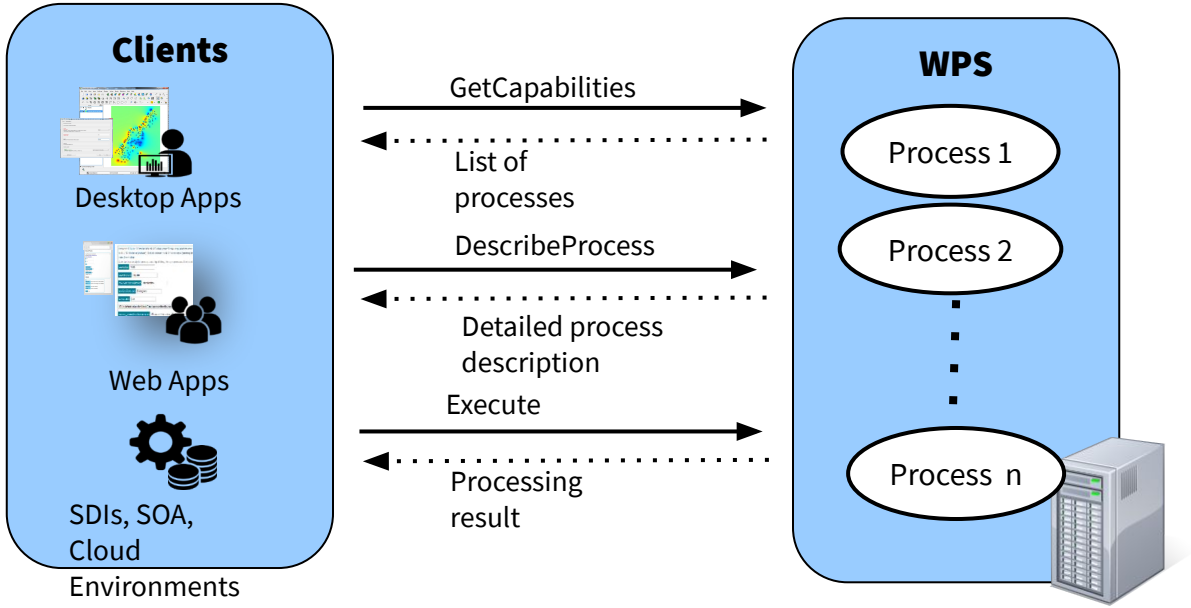
Standardization for Geoprocessing

OGC and related Concepts

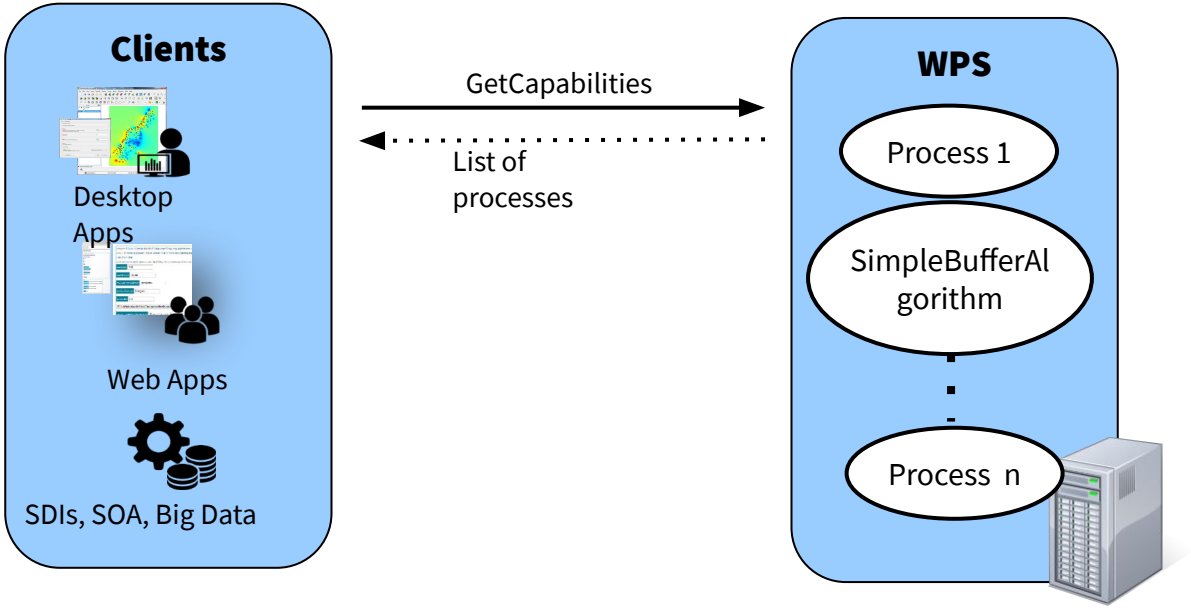
OGC Web Processing Service - Overview

- Official OGC Standard since 2007, Version 2.0.0 since 2015:
  - http://www.opengeospatial.org/standards/wps
- Standardized description of geoprocessing functionality („processes“)
  - Identifier
  - Textual description
  - Input and output parameters
- Predefined service operations for the description and execution of processes (synchronous, asynchronous)
  - DescribeProcess, Execute, GetResult
- Software:
  - 52°North WPS, PyWPS, Zoo WPS, ArcGIS Server, ERDAS Imagine, …

OGC Web Processing Service – Basic operations



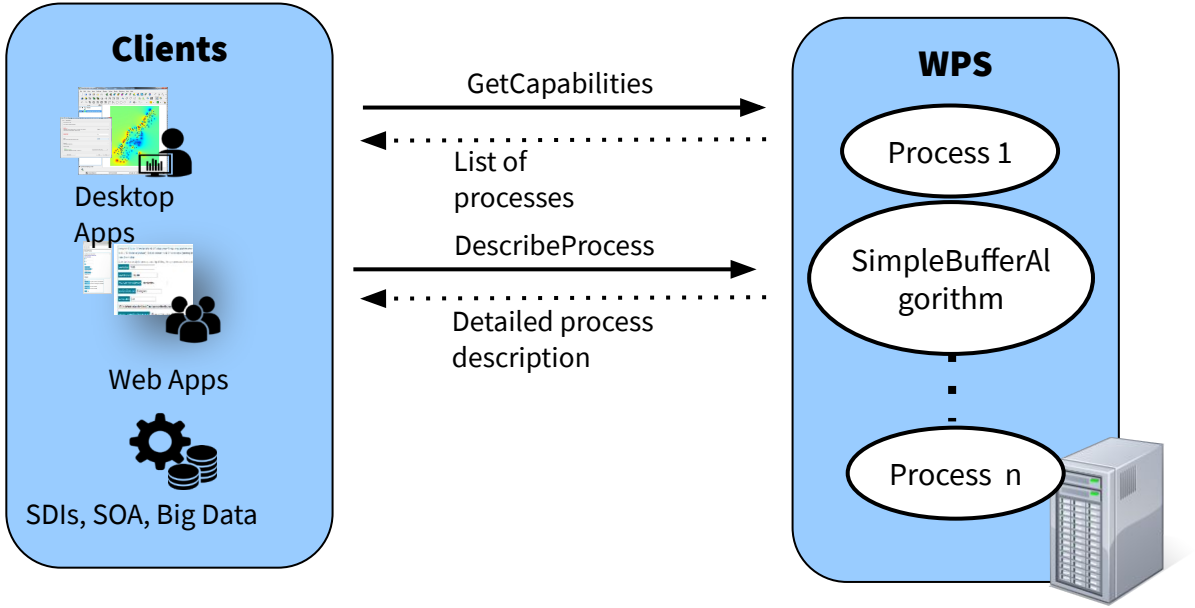
WPS – Which processes are available?



WPS – GetCapabilities Operation

- Request via URL (HTTP GET with Key-Value-Pair encoding) or XML Request (HTTP POST)
- Returns service description of the WPS
- Basic information:
    - Endpoints
    - Technical request mechanisms
    - Information about the service provider/access constraints
- Short information about the offered processes

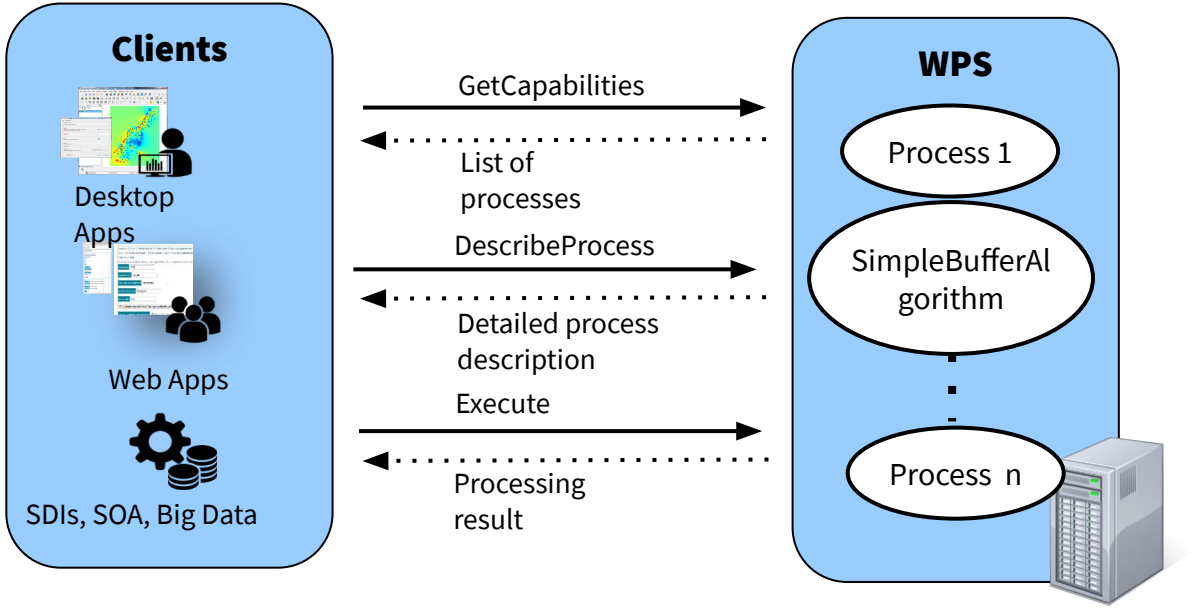
WPS – What does the process do? What are the inputs/outputs?



WPS: DescribeProcess Operation

- Request via URL (HTTP GET with Key-Value-Pair encoding) or XML Request (HTTP POST)
- Returns detailed description of a process based on the process id
- Defines the inputs and outputs
  - IDs
  - Default data formats
  - Further supported data formats

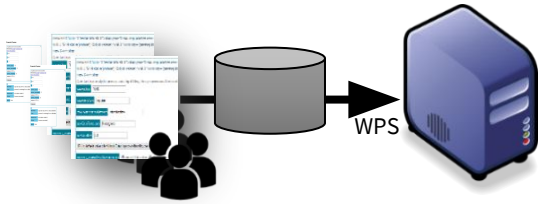
WPS – How do I execute a process?



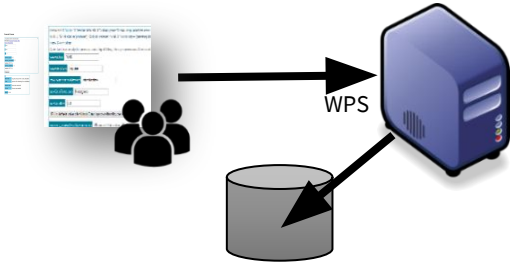
WPS – Execute Operation

- XML Request (HTTP POST)
- Execution of an offered process
- Request:
  - Must contain id and input parameters according to the process description
- Result
  - Can be returned directly or as reference to a web accessible resource
- Can be executed asynchronously for long running processes -> Client doesn't directly get the result, but can request the status of the execution

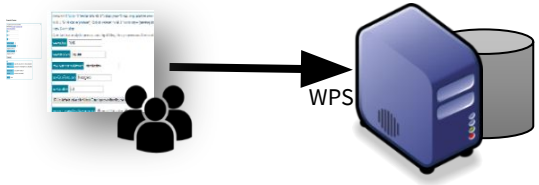
WPS – How to transfer the input data



Option 1: Direct transfer

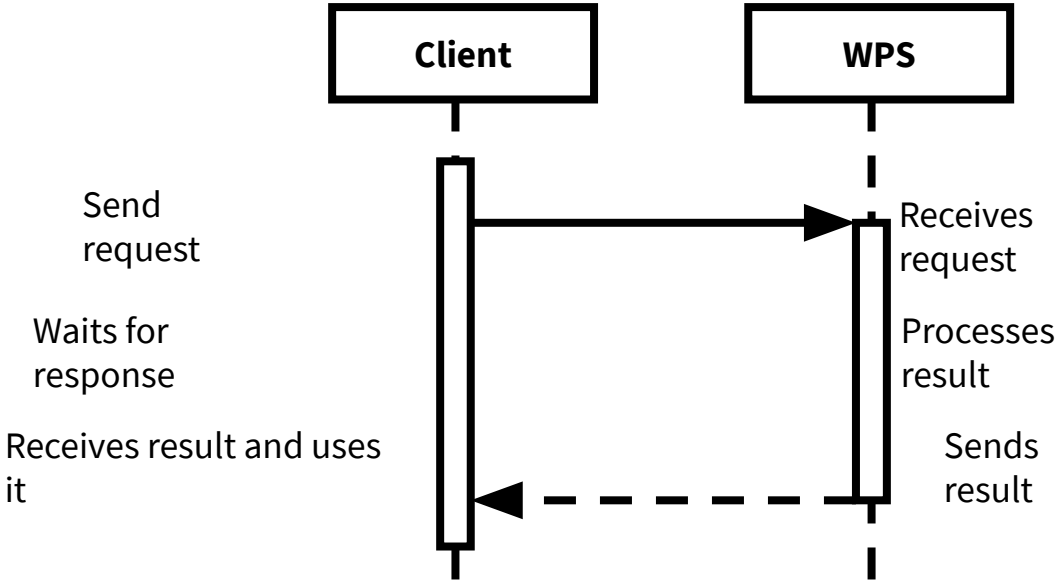


Option 2: Reference to web accessible resource



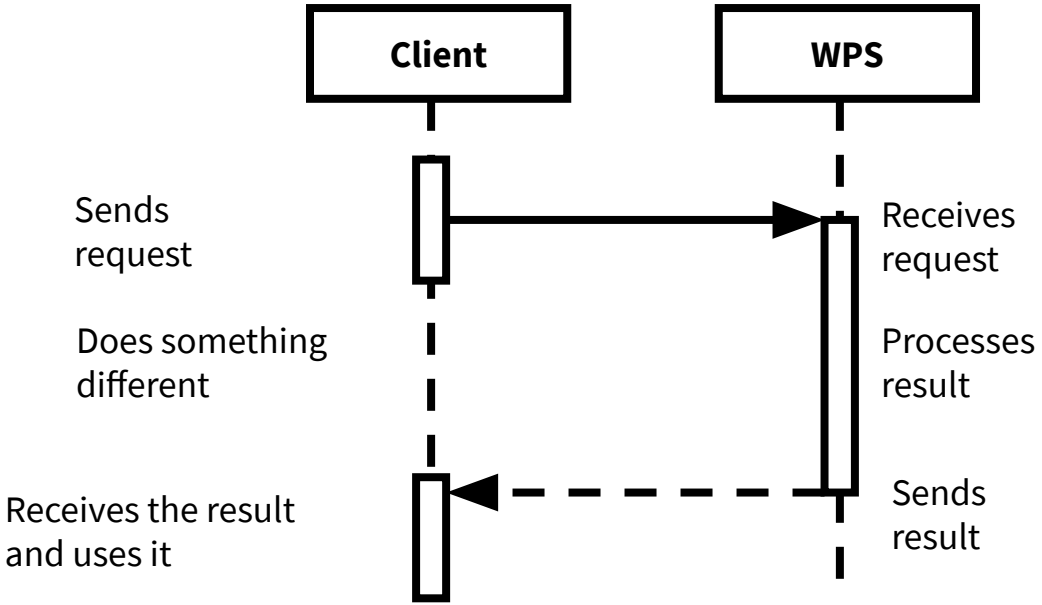
Option 3: Query on encapsulated data

WPS – Synchronous execution

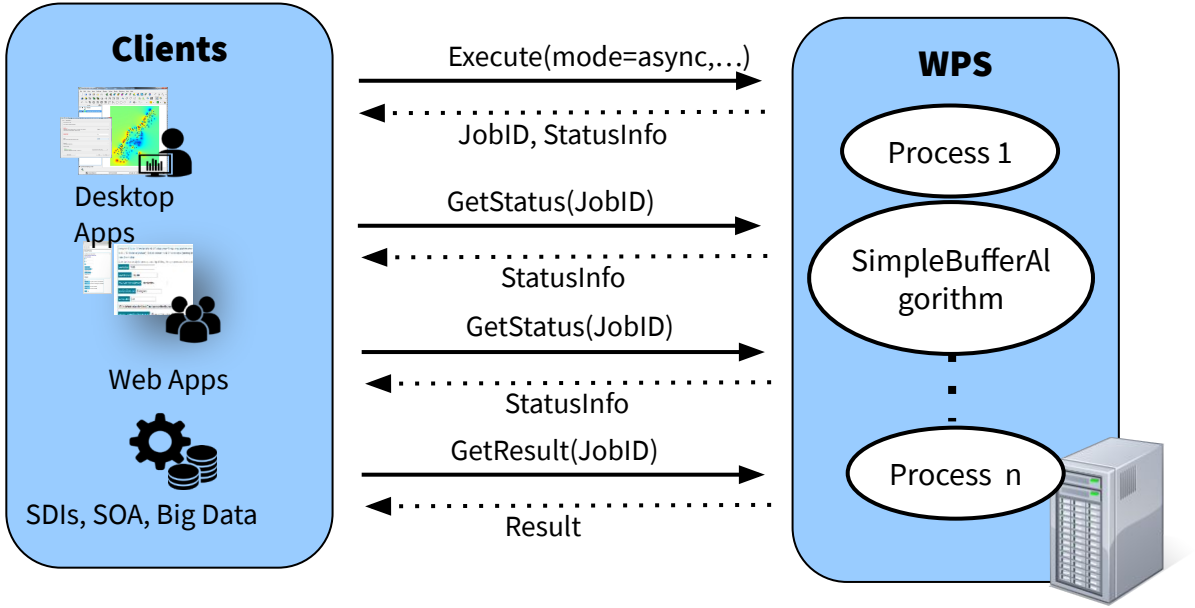


WPS – Asynchronous Execution(Push-Model)

Sequence Diagram

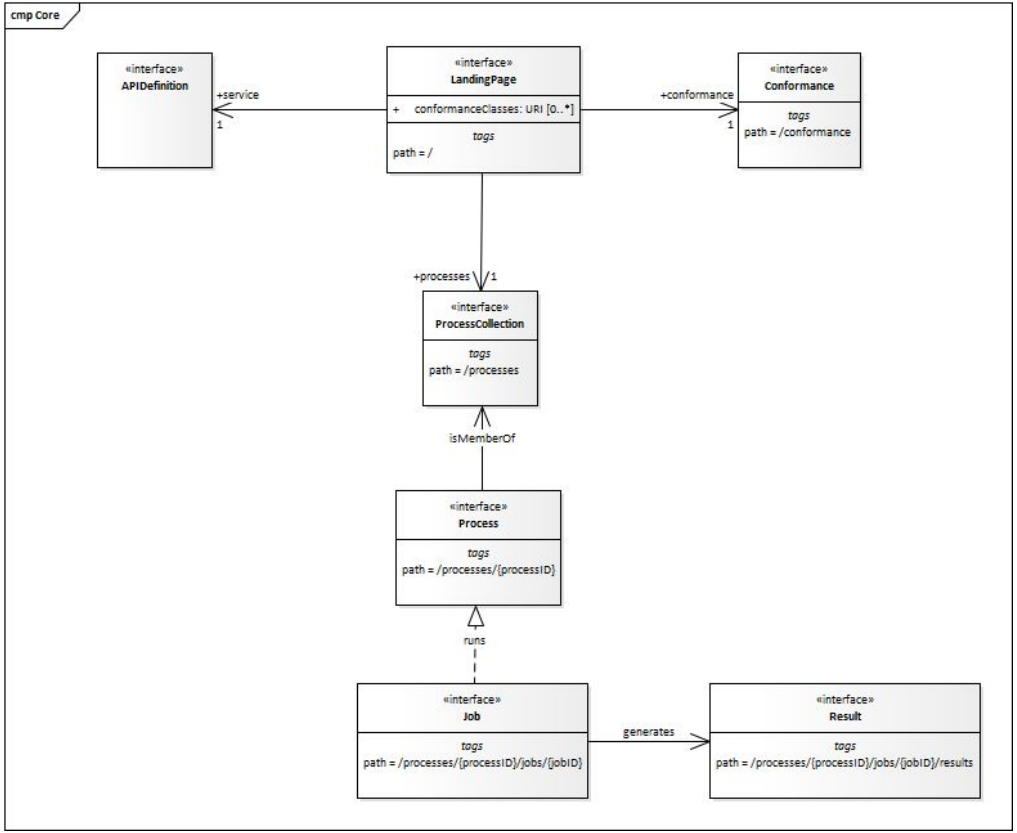


WPS Spec 2.0 – Asynchronous execution



OGC API - Processes

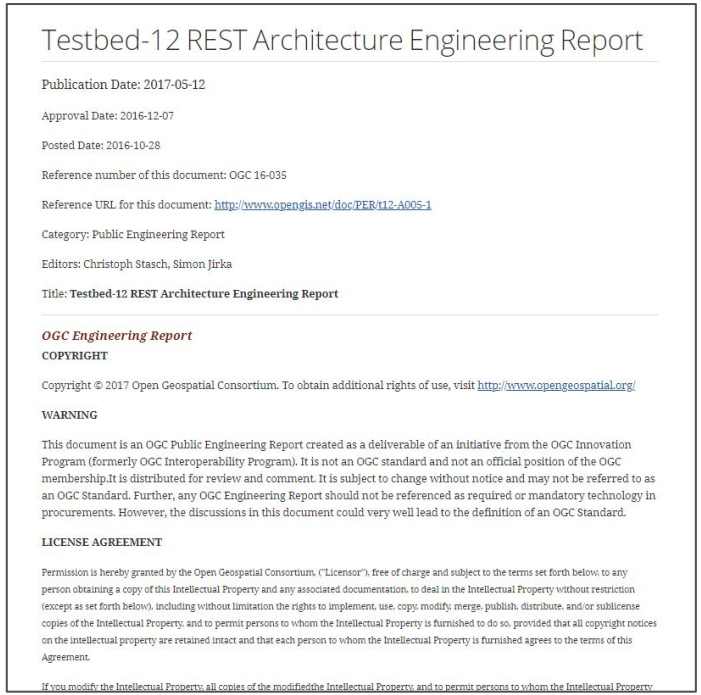
- Currently in the process of standardization
  - Next step: Request for public comments
- Several implementations are available for testing
- Core and extensions



https://github.com/opengeospatial/wps-rest-binding

Motivation For OGC APIs

- WPS 2.0 and other OGC specifications normally define a SOAP/XML Binding as necessary interface
  - Exceptions : OGC Web Map Tiling Server and Sensor Things API
- REST APIs with JSON Encodings leaner for implementations of Web clients and easier to use
  - Focus on ressources (not on operations)
  - Usa of standard HTTP operations for CRUD of resources
- In OGC Testbed 12 REST APIs for different OGC services were tested and described in the Testbed 12 REST Architecture Engineering Report



Testbed-12 REST Architecture Engineering Report

Publication Date: 2017-05-12

Approval Date: 2016-12-07

Posted Date: 2016-10-28

Reference number of this document: OGC 16-035

Reference URL for this document: http://www.opengis.net/doc/PER/t12-A005-1

Category: Public Engineering Report

Editors: Christoph Stasch, Simon Jirka

Title: **Testbed-12 REST Architecture Engineering Report**

OGC Engineering Report

COPYRIGHT

Copyright © 2017 Open Geospatial Consortium. To obtain additional rights of use, visit http://www.opengeospatial.org/

WARNING

This document is an OGC Public Engineering Report created as a deliverable of an initiative from the OGC Innovation Program (formerly OGC Interoperability Program). It is not an OGC standard and not an official position of the OGC membership.It is distributed for review and comment. It is subject to change without notice and may not be referred to as an OGC Standard. Further, any OGC Engineering Report should not be referenced as required or mandatory technology in procurements. However, the discussions in this document could very well lead to the definition of an OGC Standard.

LICENSE AGREEMENT

Permission is hereby granted by the Open Geospatial Consortium, ("Licensor"), free of charge and subject to the terms set forth below, to any person obtaining a copy of this Intellectual Property and any associated documentation, to deal in the Intellectual Property without restriction (except as set forth below), including without limitation the rights to implement, use, copy, modify, merge, publish, distribute, and/or sublicense copies of the Intellectual Property, and to permit persons to whom the Intellectual Property is furnished to do so, provided that all copyright notices on the intellectual property are retained intact and that each person to whom the Intellectual Property is furnished agrees to the terms of this Agreement.

If you modify the Intellectual Property, all copies of the modifiedthe Intellectual Property, and to permit persons to whom the Intellectual Property

http://docs.opengeospatial.org/per/16-035.html

OGC API - Processes: Resource Model

- Based upon the general process model of WPS 2.0

- Hypermedia approach
  - `Landing Page` contains link to `ProcessList`
  - `ProcessList` contains links to single `Process` –ressources
  - …

Endpoints for resource access

Ressource	Description	HTTP Operation	Endpoint	Message Body
Landing Page	Request of the service description	HTTP GET	{baseURL}	-
Process list	Request of the list of processes	HTTP GET	{baseURL}/processes	-
Process	Request of a single process description	HTTP GET	{baseURL}/processes/{processID}	-
Job list	Request of the list of jobs (executions) of a process	HTTP GET	{baseURL}/processes/{processID}/jobs	-
Job info	Execution of a process/creation of a new job	HTTP POST	{baseURL}/processes/{processID}/jobs	Execute Request in JSON
Job info	Request of the status of a job	HTTP GET	{baseURL}/processes/{processID}/jobs/{jobID}	-
Results	Request of the results of a job	HTTP GET	{baseURL}/processes/{processID}/jobs/{jobID}/results	-

52N JAVAPS-REST

- Implements OGC API - Processes

- GitHub Repo:
  - https://github.com/52North/javaPS

- Example instance for testing:
  - http://geoprocessing.demo.52north.org:8080/javaps/rest/

Request Landing Page

HTTP GET <baseURL>

- No predefined pattern for endpoint-URL
- Returns Capabilities document containing the list of available processes


```
{
    "title": "52°North OGC API - Processes",
    "description": "52°North OGC API - Processes, powered by javaPS",
    "links": [
        {
            "href": "http://geoprocessing.demo.52north.org:8080/javaps/rest",
            "rel": "self",
            "type": "application/json",
            "title": "this document"
        },
        {
            "href": "http://geoprocessing.demo.52north.org:8080/javaps/rest/api/",
            "rel": "service",
            "type": "application/openapi+json;version=3.0",
            "title": "the API definition"
        },
        {
            "href": "http://geoprocessing.demo.52north.org:8080/javaps/rest/conformance/",
            "rel": "conformance",
            "type": "application/json",
            "title": "Conformance classes implemented by this server"
        },
        {
            "href": "http://geoprocessing.demo.52north.org:8080/javaps/rest/processes/",
            "rel": "processes",
            "type": "application/json",
            "title": "The processes offered by this server"
        }
    ]
}
```


General service information

Link to process list

http://geoprocessing.demo.52north.org:8080/javaps/rest

Request Process list

HTTP GET <baseURL>/processes

- Returns a list of short process summaries containing links to detailed process descriptions


```
[
    {
        "id": "org.n52.javaps.test.EchoProcess",
        "title": "org.n52.javaps.test.EchoProcess",
        "keywords": [],
        "metadata": [],
        "version": "1.0.0",
        "jobControlOptions": [
            "async-execute",
            "sync-execute"
        ],
        "outputTransmission": [
            "value",
            "reference"
        ],
        "links": [
            {
                "href": "http://geoprocessing.demo.52north.org:8080/javaps/rest/processes/org.n52.javaps.test.EchoProcess",
                "rel": "process-desc",
                "type": "application/json",
                "title": "Process description"
            }
        ]
    },
    {
        "id": "org.n52.javaps.test.MultiReferenceInputAlgorithm",
```


Link to detailed process description

http://geoprocessing.demo.52north.org:8080/javaps/rest/process

Reqest process description

HTTP GET `<baseURL>/processes/<process-id>`

- Returns the detailed description of a process including input and output parameters

Request process description


```
{
    "id": "org.n52.javaps.test.EchoProcess",
    "title": "org.n52.javaps.test.EchoProcess",
    "version": "1.0.0",
    "jobControlOptions": [
        "async-execute",
        "sync-execute"
    ],
    "outputTransmission": [
        "value",
        "reference"
    ],
    "links": [
        {
            "href": "http://geoprocessing.demo.52north.org:8080/javaps/rest/processes/org.n52.javaps.test.EchoProcess/jobs",
            "rel": "execute",
            "title": "Execute process",
        }
    ],
    "inputs": [
        {
            "id": "complexInput",
            "title": "complexInput",
            "input": {
                "formats": [
                    {
                        "default": true,
                        "mimeType": "application/xml"
                    },
                    {
                        "default": false,
                        "mimeType": "text/xml"
                    }
                ]
            },
            "minOccurs": 0,
            "maxOccurs": 2
        }
    ],
    "outputs": [
        {
            "id": "complexOutput",
            "title": "complexOutput",
            "output": {
                "formats": [
                    {
                        "default": true,
                        "mimeType": "application/xml"
                    },
                    {
                        "default": false,
                        "mimeType": "text/xml"
                    }
                ]
            }
        }
    ]
}
```


Execute-URL

Input parameter

Output parameter

Execution of a process (1)

HTTP POST
`<baseURL>/processes/<process-id>/jobs`

- Parameter:
  - Execute Request in JSON (see to the right)
- A new job resource (process execution) is created

Input data

Desired outputs

```
1  {
2      "inputs": [
3          {
4              "id": "complexInput",
5              "input": {
6                  "format": {
7                      "mimeType": "application/xml"
8                  },
9                  "value": {
10                     "inlineValue": "<test/>"
11                 }
12             }
13         }
14     ],
15     "outputs": [
16         {
17             "id": "complexOutput",
18             "format": {
19                 "mimeType": "application/xml"
20             },
21             "transmissionMode": "value"
22         }
23     ],
24     "response": "document",
25     "mode": "async"
26 }
```

http://geoprocessing.demo.52north.org:8080/javaps/rest/processes/org.n52.javaps.test.EchoProcess/jobs

Execution of a process (ii)

- *Asynchronous* execution:
  - HTTP 201 with link to job ressource
  - After execution is finished link to results


```
{
    "status": "successful",
    "jobID": "2ceb5c7d-69cb-4064-a941-1c812506bfd8",
    "links": [
        {
            "href": "http://geoprocessing.demo.52north.org:8080/javaps/rest/processes/org.n52.javaps.test.EchoProcess/jobs/2ceb5c7d-69cb-4064-a941-1c812506bfd8",
            "rel": "self",
            "type": "application/json",
            "title": "This document"
        },
        {
            "href": "http://geoprocessing.demo.52north.org:8080/javaps/rest/processes/org.n52.javaps.test.EchoProcess/jobs/2ceb5c7d-69cb-4064-a941-1c812506bfd8/results",
            "rel": "results",
            "type": "application/json",
            "title": "Job results"
        }
    ]
}
```


Finished job with link to results

Execution of a process (iii)

- Synchronous execution:
  - JSON result document (success|failure)

Request processing results

HTTP GET:

`<baseURL>/processes/<process-id>/jobs/<job-id>/outputs`

- Returns JSON result document

```
1  {
2      "outputs": [
3          {
4              "id": "complexOutput",
5              "value": {
6                  "inlineValue": "<test/>"
7              }
8          }
9      ]
10 }
11
```

JAVAPS REST TESTCLIENTS

HTML view of execute endpoints, e.g.:
http://geoprocessing.demo.52north.org:8080/javaps/rest/processes/org.n52.javaps.test.EchoProcess/jobs

Jobs

Show this page as JSON document

- 9cccb20e-c47d-4e57-a886-97b124b7c50a
- 2ceb5c7d-69cb-4064-a941-1c812506bfd8

Submit new job

Load example request

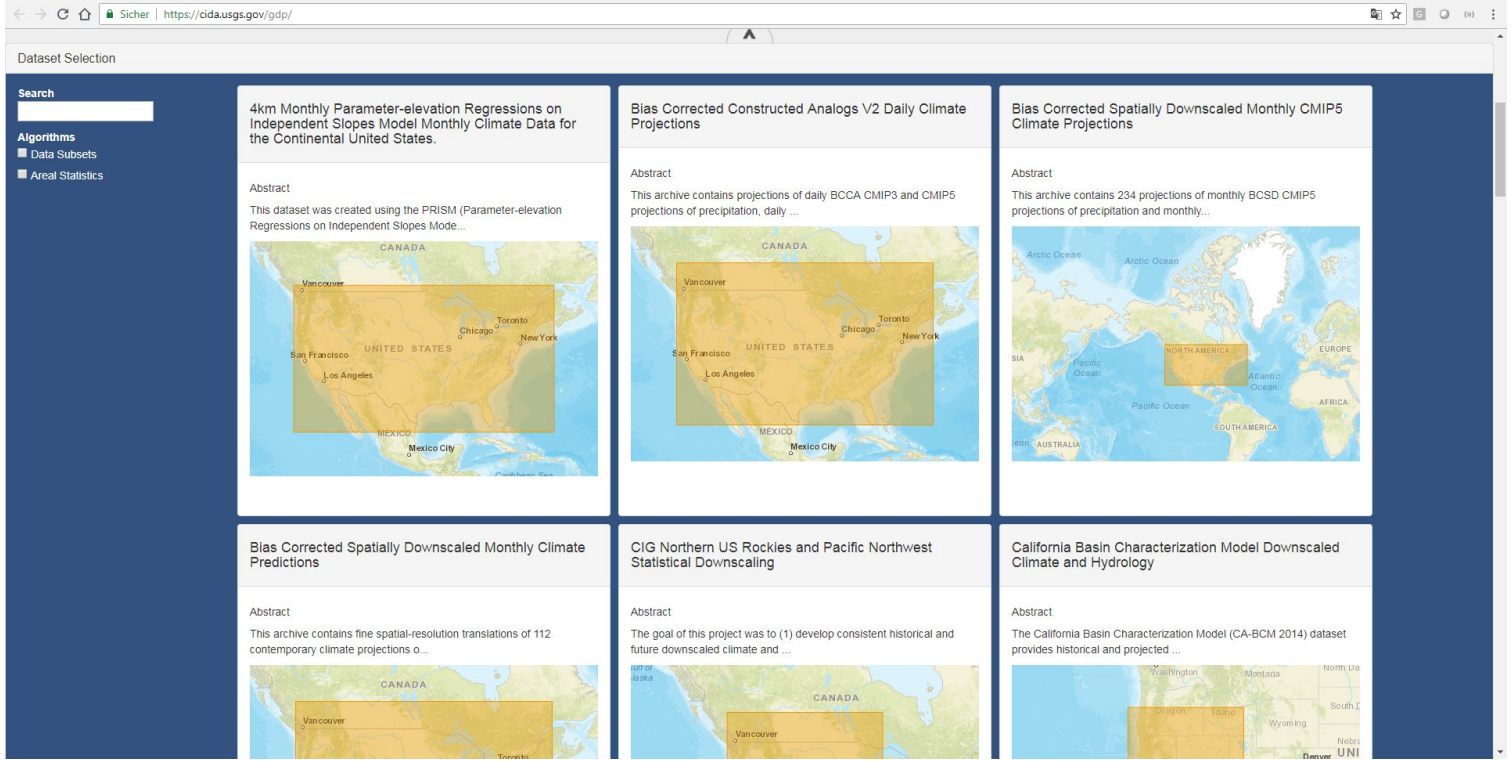

```
{
    "inputs": [
        {
            "id": "complexInput",
            "input": {
                "format": {
                    "mimeType": "application/xml"
                },
                "value": {
                    "inlineValue": "<test/>"
                }
            }
        },
        {
            "id": "literalInput",
            "input": {
                "dataType": "double",
                "value": "0.05"
            }
        },
        {
            "id": "boundingboxInput",
            "input": {
                "bbox": [
                    51.9,
                    7,
                    52,
                    7.1
                ],
                "crs": "EPSG:4326"
            }
        }
```


Create job

Example Applications

Practical Use Cases

Example application: USGS

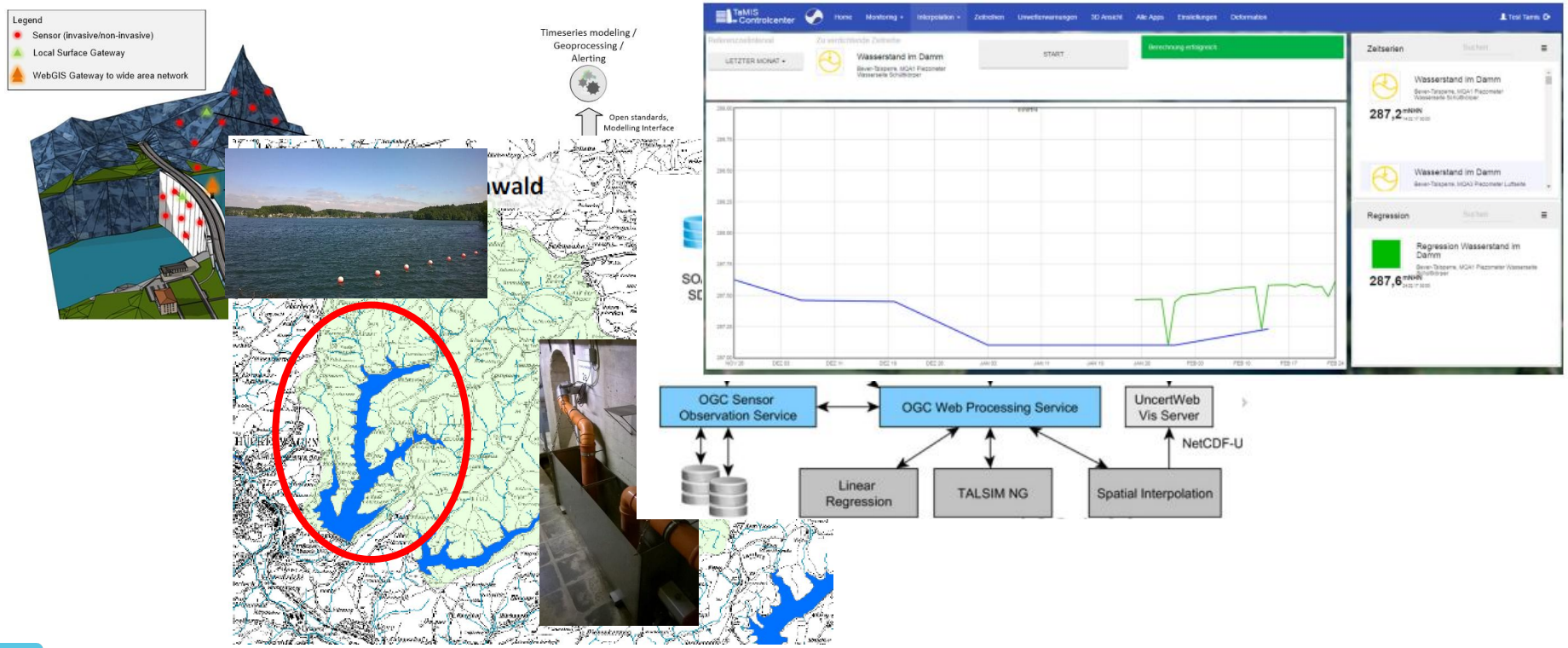


https://cida.usgs.gov/gdp/

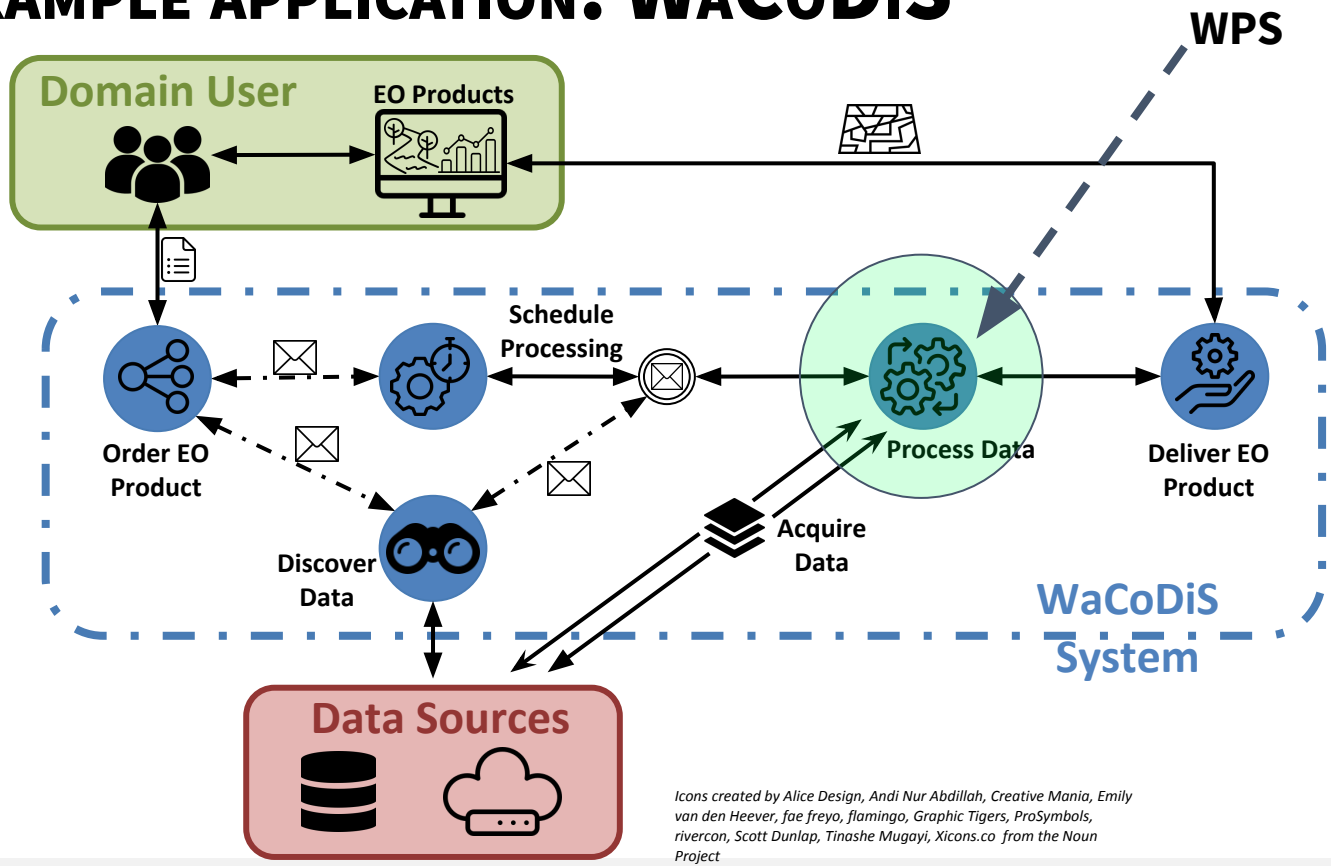
Example application: TAMIS

GEOTECHNOLOGIEN

Bundesministerium für Bildung und Forschung



Example application: WaCoDiS



Icons created by Alice Design, Andi Nur Abdillah, Creative Mania, Emily van den Heever, fae freyo, flamingo, Graphic Tigers, ProSymbols, rivercon, Scott Dunlap, Tinashe Mugayi, Xicons.co from the Noun Project

Software Solutions - Server & Client

52N Software, other Open Source solutions

Server Implementations

- OGC lists 69 implementations for WPS 1.0/2.0
- Comercial:
  - ESRI, FME, Intergraph, Envitia, ERDAS, …
- Open Source:
  - 52°North, pyWPS, ZOO, Geoserver, …

52°North WPS

- Beta-Release of version 4.0.0 available
- Supports (all) features and operations of the WPS specification Version 1.0.0 and 2.0
- Support of:
  - GRASS 7, Sextante, R, Java/Python
- Java Process/R Script upload via Web UI
- Parser/Generators for common data formats:
  - SHP-Files, GML, GeoJSON, GeoTIFF, NetCDF, …

52°North JAVAPS

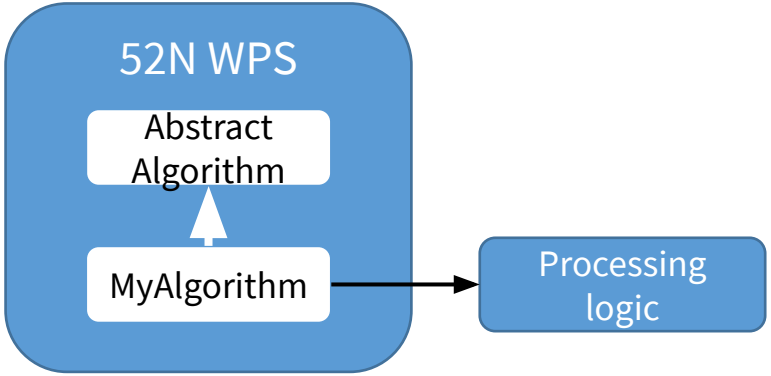
- Version 1.5.0
- Complete new implementation using state-of-the-art technology
- Supports WPS 2.0 and OGC API - Processes
- Supports deployment of new processes (transactional extension)
- Parser/Generators for common data formats:
  - SHP-Files, GML, GeoJSON, GeoTIFF, NetCDF, …

52°North WPS Deployment Patterns

- Extending an existing WPS
- Custom Binaries with configuration files
- Rich Data Interfaces
- Deployment of annotated scripts (currently R)

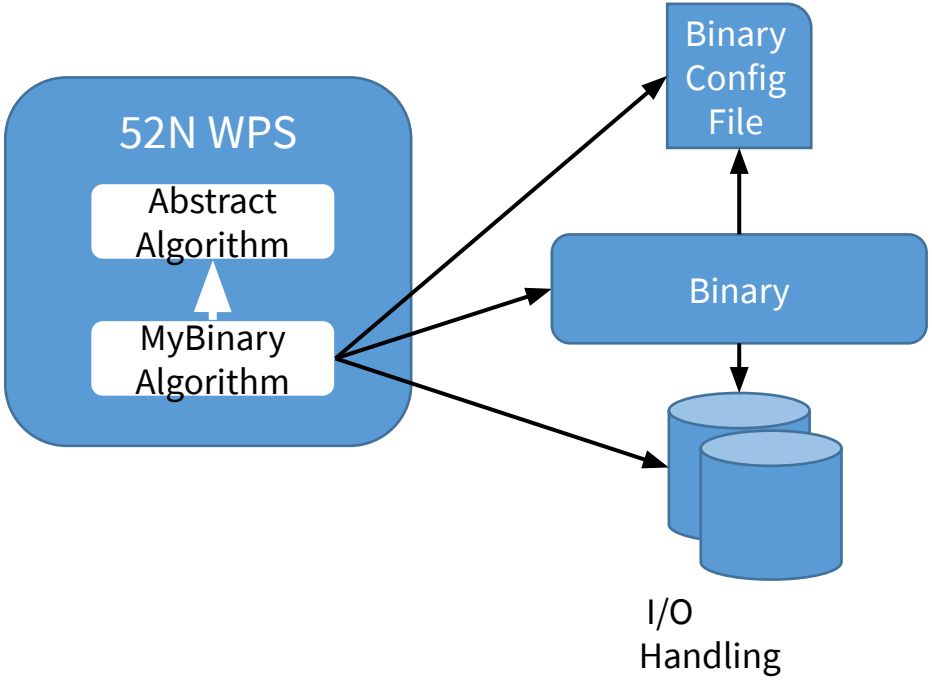
WPS EXTENSION

- Direct extension of a existing WPS Server implementation
  - Reuse of I/O handlers
- Example: Implementation of an algorithm in Java as direct extension of the 52N javaPS



Execution of Custom Binaries

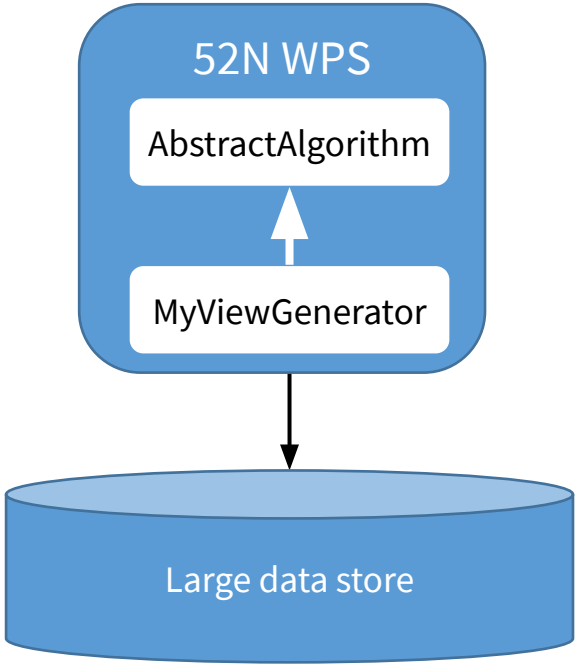
- Execution of a program in binary code
- Configuration via config files
  - Parameters for controlling the process
  - Inputs/Outputs are stored locally and are referenced using config parameters



WPS as Rich Data Interface

- WPS as interface for a large data store
- Dynamic views can be generated using input parameters

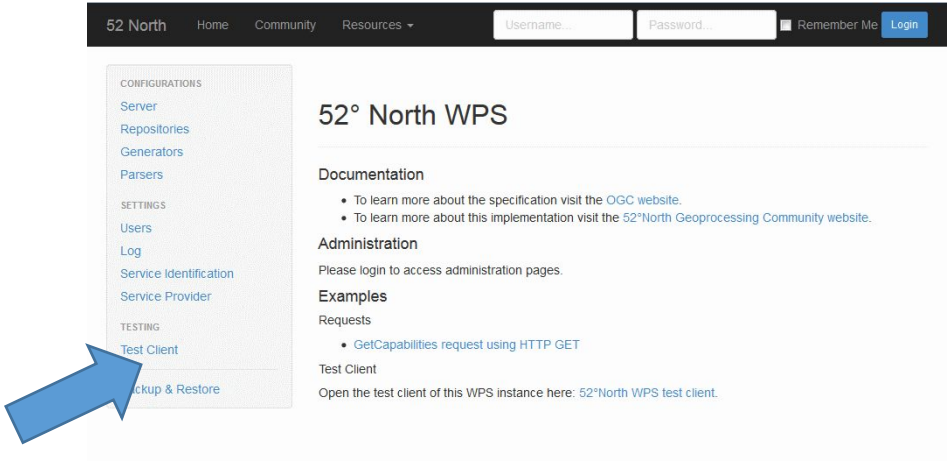
  - Inputs don’t contain data for processing



Deployment of R Scripts

- Annotation in R scripts define inputs/outputs
- Upload of annotated R scripts allows deployment as WPS processes

```
# wps.des: id = Random, title = Random number generator,
# abstract = Generates random numbers for uniform distribution;
# wps.in: min, double, Minimum, All outcomes are larger than min, value = 0;
# wps.in: max, double, Maximum, All outcomes are smaller than max, value = 1;
# wps.in: n, integer, ammount of random numbers, value = 100;
# random number:
x = runif(n, min=min, max=max)
output = "outputfilename"
write.table(x, output)
# wps.out: output, text, Random number list, Textfile containing n random numbers in one column;
```

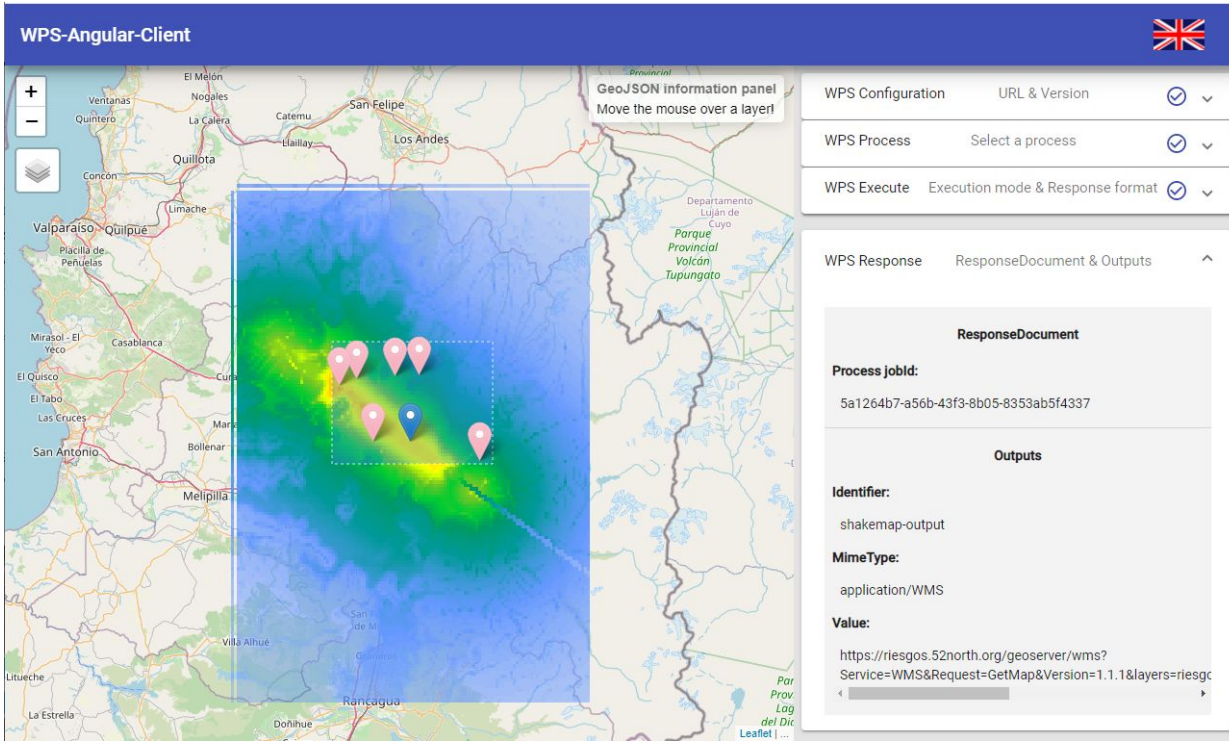


Client Implementations

- 52°North ArcGIS WPS Client
  - ArcGIS Extension in collaboration with ESRI Inc.
  - Available as Open Source
- 52°North WPS-JS
  - JavaScript library for the creation of Web clients
- Quantum-GIS WPS – Plugin
- Custom clients for specific applications, e.g.USGS

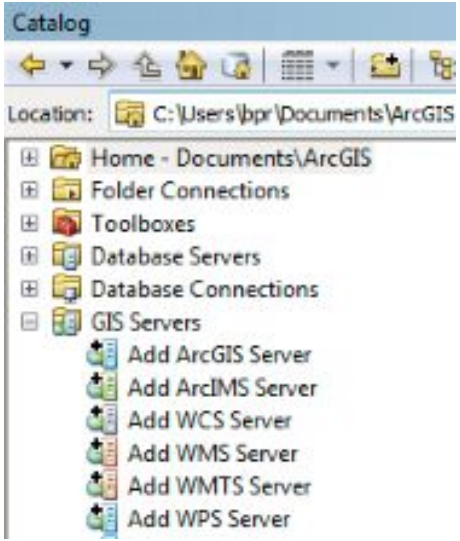
52°North WPS-NG-CLIENT

- wps-js:
  - JavaScript client library
- Wps-ng-client:
  - JavaScript client
  - Based on the Angular Framework

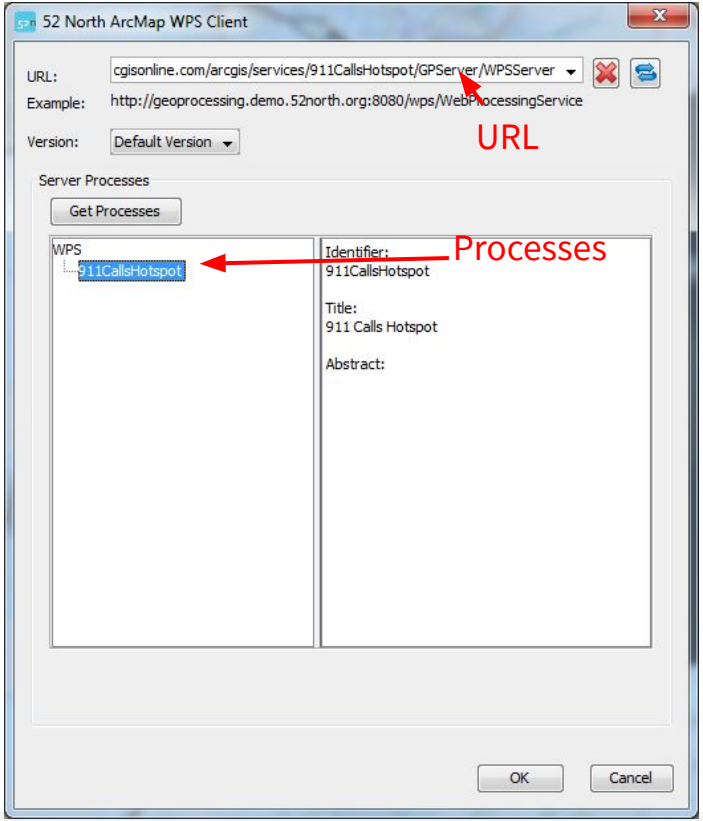


https://github.com/52North/wps-ng-client

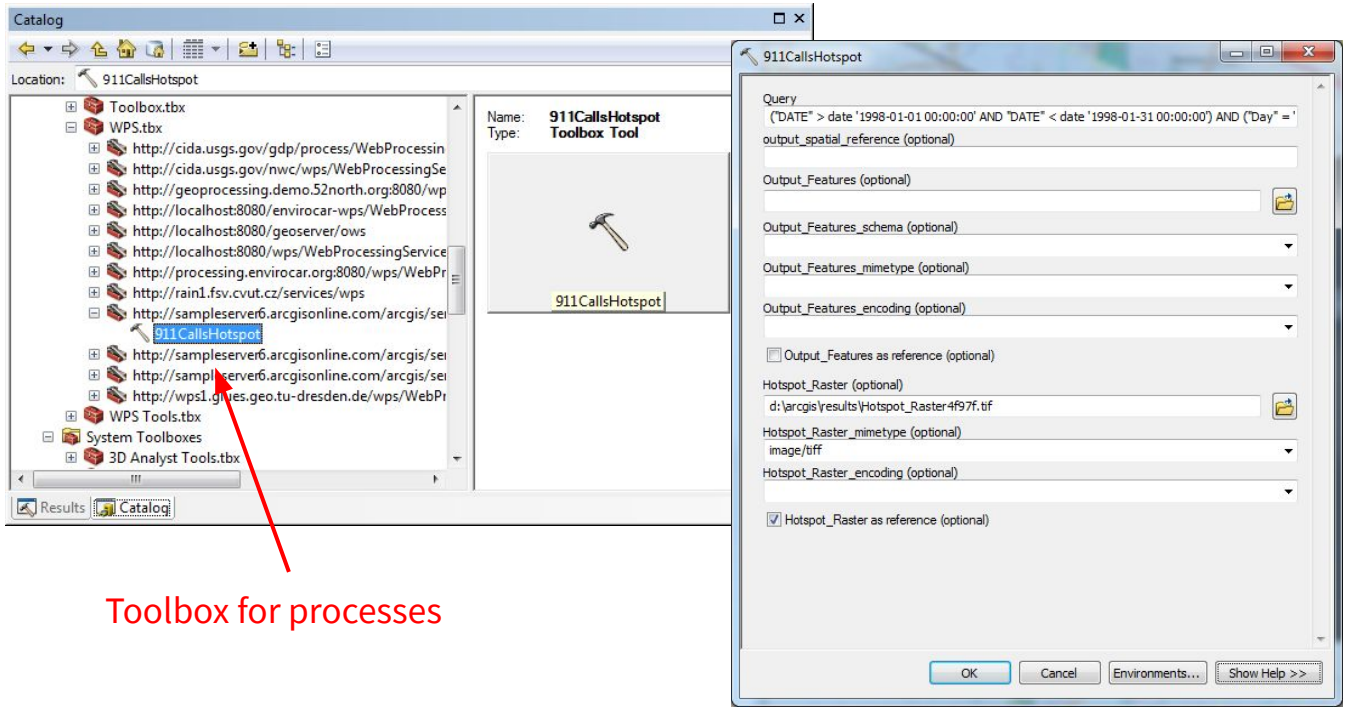
52°North ArcGIS WPS Client (I)



Adding of a WPS



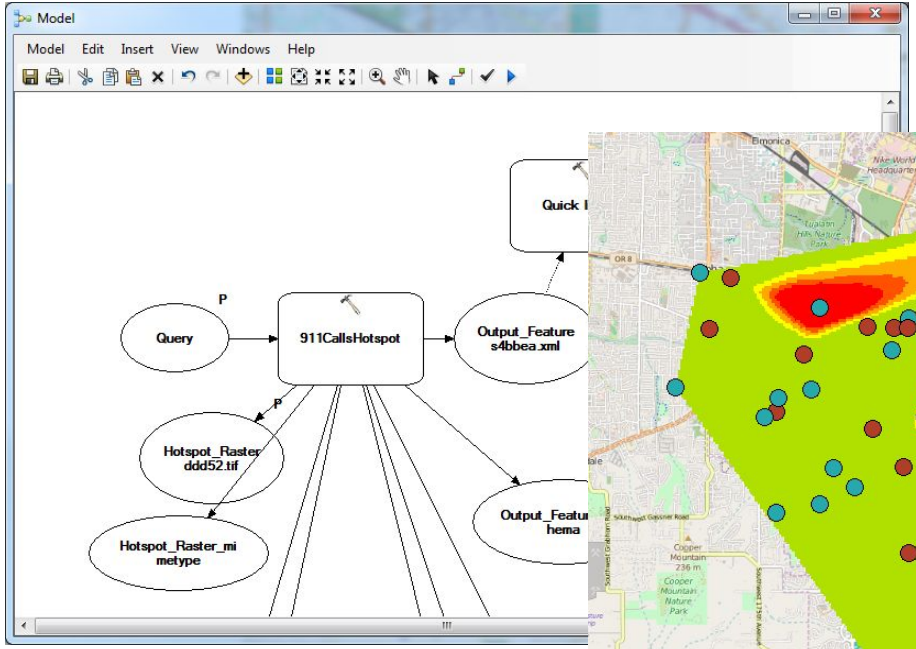
52°North ArcGIS WPS Client (II)



Toolbox for processes

Input form for the execution

52°North ArcGIS WPS Client (iii)



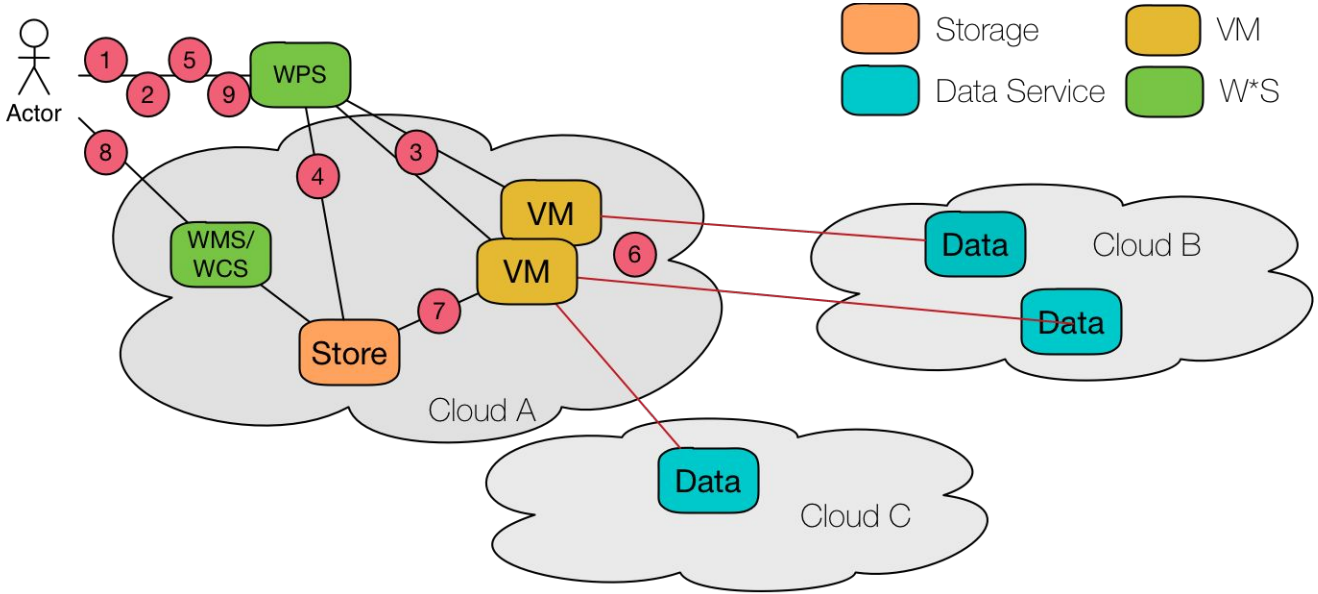
Integration in the Model Builder

Result map

Cloud-based Processing - Hands-on

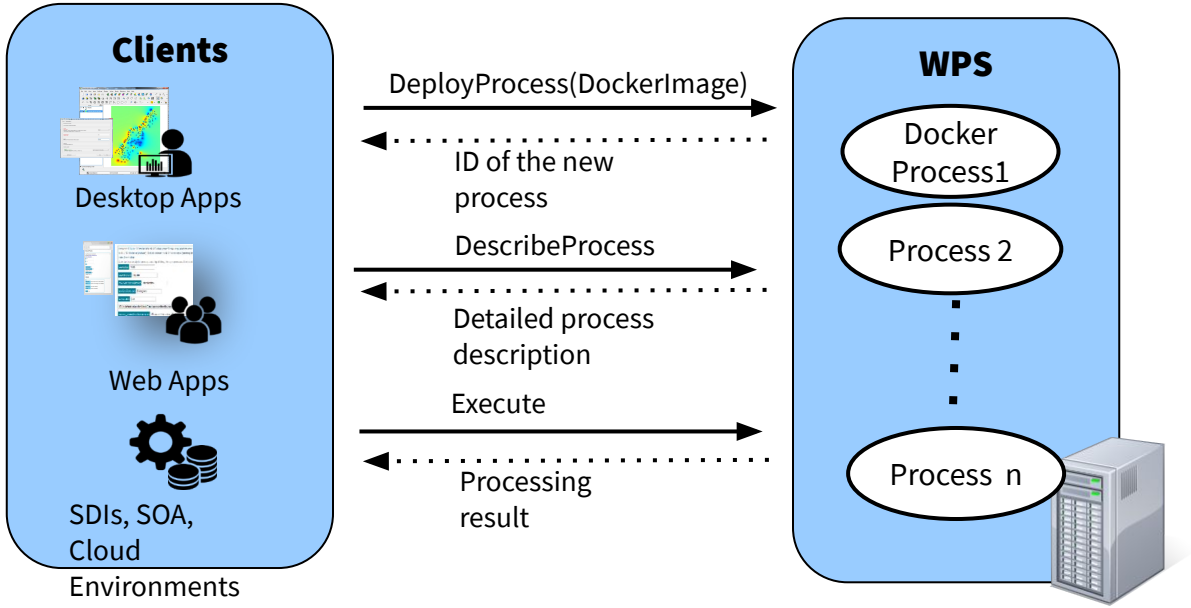
Developing a Process with Python / Jupyter Notebook and making it available on the Web

Interface for Cloud-based Geoprocessing



Quelle:
http://www.opengeospatial.org/node/2526#Cloud

Hosted Processing



Concept & Approach

- What is the current trend of developing “processing” functionality in the research community and the data science/analysis domain?

  → **Jupyter Notebooks or R** [Notebooks, Markdown]

- Idea:
  a. develop a (simple) process
  b. make it executable in an environment-independent way
  c. host it in the Cloud [, close to the data]

Concept & Approach

- Approach:
  a. develop a (simple) process

     → Jupyter Notebooks
  b. make it executable in an environment-independent way

     → Docker with Python Kernel
  c. host it in the Cloud [, close to the data]

     → OGC API Processes, with Docker Execution Backend

Jupyter Notebook Process

https://nbviewer.jupyter.org/github/matthesrieke/gsv2020/blob/master/geojson_buffer.ipynb

- How can we make the (interactive) Notebook executable without manual interaction?

→ **Papermill** allows parameterization

Docker Image Preparation

1. Reasonable Base Image: **continuumio/miniconda3**

   → https://github.com/matthesrieke/gsv2020/blob/master/Dockerfile

2. Setup environment: import required libraries (geopandas)

   → https://github.com/matthesrieke/gsv2020/blob/master/environment.yml

3. Define command that executes the Notebook (Papermill)

Deploy the Process on the Web

ADES: Application Deployment and Execution Service

- Extension to OGC API Processes
- introduces transactional API (registration, management of processes)
- makes use of “Execution Units” (e.g. Docker images, other programming language environments)

→ https://github.com/matthesrieke/gsv2020/blob/master/application-package.json

Demo with Web Client

https://52north.github.io/wps-ng-client/

OGC API Processes (draft) instance:
https://testbed.dev.52north.org/ades/rest/

OGC WPS 2.0 instance:
https://testbed.dev.52north.org/ades/service

ADES - Things to Keep in Mind

- No defined interface between “Application Package” und “Execution Unit”:
  - The way how Inputs and Outputs are communicated between these two interfaces depends (at the moment!) on the implementation
  - Example: https://github.com/matthesrieke/gsv2020/blob/master/Dockerfile#L11
- Docker Execution is most suitable for heavyweight computations as the bootstrapping of Docker containers introduces an overhead → less relevant for longer running processes

ADES - Road Ahead

- Feasibility Study in established Cloud environments
  - Kubernetes / Google Cloud Platform
  - AWS
- Generic Approach for Notebooks → improvements in javaPS <-> container interaction
- Extension to additional frameworks (R notebooks, ...)

Summary

Topics Covered

- Geoprocessing motivation
- WPS 2.0 as the established standard
- OGC “API Processes” as the emerging next generation API
- Client & Server Software
- Cloud-based Processing
  - Jupyter Notebook
  - Docker
  - ADES - OGC API Processes

Thanks for the Attention!

Questions?

Matthes Rieke (m.rieke@52north.org)
Benjamin Proß (b.pross@52north.org)

Resources

- OGC Resources:
  - OGC API Processes: https://github.com/opengeospatial/wps-rest-binding
  - Application Package: http://docs.opengeospatial.org/per/17-023.html
- Papermill: https://github.com/nteract/papermill
- 52N javaPS: https://github.com/52North/javaps
- WPS 2.0 Web Client: https://github.com/52North/wps-ng-client