Web Processing Service (WPS) Orchestration

A Practical Approach

Dorian Alcacer-Labrador B.Sc.
University Of Applied Sciences Osnabrueck - Faculty of Engineering and Computer Science
Albrechtstr. 30
Osnabrueck, Germany
d.alcacer@hs-osnabrueck.de

Felix Bensmann, M.Sc.
University Of Applied Sciences Osnabrueck - Faculty of Engineering and Computer Science
Albrechtstr. 30
Osnabrueck, Germany
f.bensmann@hs-osnabrueck.de

Prof. Dr.-Ing. Rainer Roosmann
University Of Applied Sciences Osnabrueck - Faculty of Engineering and Computer Science
Albrechtstr. 30
Osnabrueck, Germany
r.roosmann@hs-osnabrueck.de

ABSTRACT

The overall experience of any geospatial processing environment depends on the successful collaboration of different distributed components, such as clients, web portals, multiple processing-nodes and service discovery-measures. Many aspects of web service based geospatial processing have already been tackled and put into action at various operational levels and by different means. This contribution provides a brief review of conducted research and work done in the past. It continues with demonstrating a concept for workflow composition based on local and distributed processes. Therefore a domain-specific language is introduced and the field of application demonstrated by an accompanying use case. The overall concept is evaluated using the framing project RichWPS.

Keywords

Geographic Information System (GIS), Spatial Data Infrastructure (SDI), Web Processing Service (WPS), Domain-Specific Language (DSL), Orchestration.

1. INTRODUCTION

Gathering, linking and processing of data is the key to geospatial analysis and hence the very basis for turning data into re-usable information. Web based provision and processing of geospatial data can be achieved with service-oriented architectures (SOA) by means of web services (e.g. [10]). With the INSPIRE Directive (Infrastructure for Spatial Information in the European Community) [7] put into action and the increasing capabilities of networks and hardware, web service based processing has become even more important for delivering data and to support and publish processing functionality. Firstly web service based processing promises to bring dynamic content to end-users like researchers or decision makers by using Geographic Information Systems (GIS) and WebGIS applications. Secondly in-house workflows can be realized re-using existing web-services. Defined and issued by the Open Geospatial Consortium (OGC), the Web Processing Service (WPS) [13] defines an XML based interface for publishing, describing and executing predefined geographic processes and algorithms in a standardized and interoperable manner. In the past years much research has been conducted to investigate the efficient use of WPS for modeling scientific processes and workflows by different means.
The following contribution aims at providing a focused overview of related work that outlines recent research as well as examined and given solutions in the field of web processing. This will be followed by introducing a concept that tries to apply gained insights to a possible runtime environment for process-chaining and orchestration. In a third step a first prototype implementation will be discussed whilst applying the concept to a given use case. This is followed by a conclusion and an outlook towards further research.

2. RELATED WORK

According to the WPS 1.0 specification [13], its purpose is to serve geospatial related functionality that is encapsulated in processes. Those processes are made usable via a standardized web service and can thus be chained to achieve greater tasks. The introduction of process chaining makes any basic geospatial functionality, scientific methodology or workflow representable through the use of a process. When also using third-party web services, chaining is commonly referred to as *web service orchestration* (WSO) (e.g. [9]).

In the context of composing the combination of WPS with the Web Service Description Language (WSDL) and Simple Object Access Protocol (SOAP) has been reported to increase the re-use of geospatial web services [10]. Further research succeeded in chaining processes by using the Business Process Execution Language (BPEL) in combination with an external Web Service Orchestration Engine (WSOE) [1]. In order to improve the WPS standard itself, others demonstrated the shift of the WSOE into the Web Processing Service server [15], which enables the dynamic definition and deployment of new workflows at runtime.

Despite the fact that the use of the WS standards (WSDL, SOAP, BPEL) is widely accepted in mainstream informatics, that solution, specifically the choice of BPEL, is only one among many (e.g. [8]). Considering WSO and modeling on a bigger scale, like suggested in the context of GEO/-GEOSS Model Web, various technical and conceptual challenges when dealing with web service composition are still immanent and solutions yet to be found [12]. For instance, at the moment of writing, multiple commercial and open-source realizations of the WPS 1.0 specification are available. Considering open source implementations[1], it appears as if the ready-to-use support for the above shown orchestration concepts is not yet covered. Moreover, the platforms notably differ in details of implementation such as the level of WS standard support, especially when it comes to interface description via WSDL.

In context of WPS, the chaining of processes with custom XML and subsequent delivery to a WPS server could present a more robust, time efficient solution with less infrastructure, effort and general overhead [16]. Moreover, turning away from the full use of WS standards can be a viable approach, which is shared by different parties for server- and client side workflow processing (e.g. [16], [4]). For example the considered use of a subset of WS standards (WSDL, SOAP) for client side workflow description and execution in combination with SCUFL has been demonstrated [4]. Those approaches are considered to be steps towards a smaller technology-stack and eventually less complex concepts, which could lead to a better usability and spread of technologies at hand.

Another aspect when using web services for geospatial processing can be resource efficiency. Especially the recurring exchange of mass data in terms of redundant transfers of geospatial data through network interfaces could become costly and consequently negatively affect the overall processing time. There exist various concepts, such as grid-computation and the moving code paradigm (*MC-paradigm*), that deal with those shortcomings [3]. The moving code-paradigm results from the concept of moving the processing-algorithm near to the data source. This idea is accompanied by the need for expressing and interpreting geospatial processes e.g. through a common *process algebra* and the intention of providing corresponding processing back-ends. Another viable concept might be the comprehensive contraction of software- and hardware-preconditions needed for executing additionally installed components [11].

Different solutions for orchestrating have been presented, as well as their benefits and shortcomings. The approach of a hard coded chaining, as proposed in the WPS 1.0 standard, is rigid in that contained instructions cannot be understood or changes not be made, without an expert and reasonable effort, considering a full fledged general purpose language is applied. However, an advantage is the use of control structures and mechanisms provided by the parenting language. Another approach, the on-off cascading chaining is suited at a higher level, but suffers from the lack of reuse options through other users, because there is no central deployment. Furthermore even in this case the composition is rather static. A third approach, using an orchestration engine that executes scripts, seems to be the most practical approach in that it does not share disadvantages mentioned afore. Available technologies, particularly BPEL, might be to complex for domain experts. The WPS 1.0 specification does not require WPS implementations and individual processes to be described using WSDL. The OGC follows the strategy of propagating self describing services, so any further description would be redundant. Also, BPEL does not offer a graphical notation. The commonly known Business Process Modeling Language (BPML) cannot be mapped in a one-to-one fashion to the BPEL notation.

[1]Geoserver V 2.3.4, 52° North WPS Server V. 3.2.0 , Deegree V. 3.3.0 and PyWPS V 3.2.1

3. RICHWPS ORCHESTRATION

As the chapter related work indicates, various technical approaches for workflow composition were tested by the community in the past. Still, developing more complex, distributed WPS based applications requires experts e.g. for programming languages and software libraries used. Finally putting processes, workflows and services into action, e.g. by means of WSO, requires further expertise and efforts in terms of invested time and necessary administration.

Nevertheless, orchestration is a traditional approach to deal with complex scenarios. Complex procedures hidden in black boxes and incompatible services are wrapped with adapters and re-offered as new services. Composed processes like these are already powerful, yet they can still be further nested within one another to allow workflows with almost any complexity. However, orchestration in the area of SDIs on a level known from SOA does not yet exist. WPS offers the ideal interface for workflow publishing: with its broad design, typical SDI functionality, in particular download and processing services, can be approximated. This circumstance given, the present work suggests to further investigate the orchestration within the WPS server, and to put emphasis on the division between the two domains of process and workflow definition.

All in all, the framing project RichWPS aims at offering an easy to use environment for geospatial web service orchestration to domain experts without in-depth knowledge in informatics. The research and development is accompanied by two major use cases that are provided by the Federal Waterways Engineering and Research Institute (BAW) and Schleswig-Holstein's Government-Owned Company for Coastal Protection, National Parks and Ocean Protection (LKN) (see chapter use case).

The intended environment issues the graphical modeling of workflows and the remote testing, profiling, provision and execution of composed workflows. It consists of three major software components. The RichWPS ModelBuilder enables the graphics-aided composition of workflows based on existing local and distributed processes, and geospatial services. It obtains available processes and services from a directory service, the RichWPS SemanticProxy. Once tested, a composition can be deployed for production use on the RichWPS Server. The ModelBuilder is used to manage the workflows' lifecycle and is able to visualize results and debugging-information. The RichWPS Server, developed by the company Disy Informationssysteme GmbH, provides three interfaces for the ModelBuilder. One for testing compositions, one for profiling and estimating their performance and a third interface to publish compositions as common processes. In order to achieve this, the upcoming WPS standard and its ability to transactionally deploy and undeploy processes via WPS-T has been adopted.

This contribution focuses on the realization of the composition, the description and the deployment of executable workflows.

3.1 Orchestration Concept

The OE's architecture is determined by the requirement to integrate into given SDI structures, to basically comply with present OGC standards, and to enable the re-use of invested effort in terms of already implemented WPS processes. Especially in order to facilitate the re-use of defined workflows, the current solution rests on the given WPS specification as standardized interface. Compositions, as well as processes already available, are encapsulated using WPS, as intended by the standard. That way, the environment uses the advantage of an orchestration engine (OE) within an WPS server in combination with a corresponding workflow definition language, but avoids the afore-mentioned drawbacks that an integration of a traditional WSOE in an SDI would show.

The chosen approach, the combination of OE and WPS in a single software component, leads to the circumstance that, beneath remote processes and services, locally stored processes can be taken into account as well. For example the OE is capable to issue calls across the central memory instead of using network interfaces - a concept, which addresses the problematic field of mass data transfer. Distinguishing between local and remote processes also becomes an interesting aspect, when considering performing the dynamic reconfiguration of processes within the composition at runtime. Using a custom workflow definition language can support those and further SDI specific issues.

As shown, realizing workflow descriptions and managing their execution can be achieved with full or part wise use of the shown WS standards. Nevertheless, recurring and descriptive tasks can often be handled in a streamlined fashion by using internal or external domain-specific languages (DSLs). Considering available alternatives, a custom language, the RichWPS Orchestration Language (ROLA), is designed with the intend to avoid the use of a) pure programming languages (e.g. Java or Python) or b) XML based definitions like BPEL or custom XML. Doing so, it is assumed that a custom, designed language can be optimized for the respective task. This can be done with regard to a smaller syntax, which can arguably reduce complexity, while enhancing maintainability and readability - a decision that might also lower many given entrance barriers, and reduce the needed technology-stack.

3.1.1 ROLA

There exist two different ways of realizing domain-specific languages; namely internal DSLs or external DSLs. The major difference is that internal DSLs share language elements with the host programming language, whilst external DSLs are based on custom grammar [6]. The choice whether to use an internal or external DSL directly affects diverse aspects like the portability of the suggested solution, or the time needed for design and implementation. Relying on a given language often makes it possible to re-use already defined language elements , such as control structures, which are essential for workflow definition. Though, internal DSLs require a predefined host programming language and thus a specific runtime environment. Compared to that, external DSLs can be designed and implemented in any given programming language but the upfront design and realization seems to be more complex. Taking these aspects into consideration, an external DSL is used for creating a more specific description of workflows, that would be readable by domain experts without programming capabilities.

The language itself is designed to be process and data centric. To maintain readability, the textual representation is supposed to be a mere sequential call of available processes. Though, considering the later interpretation it contains the necessary information for e.g. parallel processing: the flow of data from the contracted inputs and outputs and the used processes. An according graphical notation enables the visual modeling. The designed notation contains a higher degree of information, which is split into several artifacts whilst deployment or testing. Among these artifacts is an interface contraction (process description) and a lean script file.

The implementation of the language takes place in an iterative procedure. The first iteration focuses on the basic language elements, necessary for achieving simple process-chains. In combination with the given use cases, it serves as testbed for identifying benefits and shortcomings, as presented in this work. The second iteration addresses the parameterization and more extensive use of the WPS protocol in order to implement more complex process-chains, and take advantage of the given specification. However, in order to approach a more scientific modeling, certain aspects such as adaptiveness, conditions, loops, basic arithmetic and exception handling need to be considered as well [2]. Also, when thinking of data-sources, the fine-tuned selection of specific datasets, e.g. selecting the n-th element of a collection, have to be considered. Therefore, the third iteration focuses on more advanced concepts of workflow modeling.

The language design rests on three key concepts: 1.) Locally available and remotely bound processes need to be explicitly distinguished in order to be able to enhance execution. 2.) Bindings for remote and local processes can potentially be exchanged by equivalent processes. 3.) Interface contraction and the workflow description are considered to be two individual domains. Even though, they share common elements and interdepend each other. Separating those concerns aims at maintaining readability of the script, and is used to hide protocol complexity wherever possible.

For now, all those measures lead to lean textual manifestation, that deals with process-chaining by providing four major language elements:

1. *References* are used to refer to runtime elements, among these are in- and output parameters defined by the interface description, and free to choose variables that

store intermediate results and values.

```
in.wps-input-identifier
out.wps-output-identifier
var.unique-identifier
```

2. *Assignments* are used to set output parameters or variables. The use of assignments is subject to restrictions. E.g values of input parameters can not be altered at runtime. In combination with references and most basic datatypes, assignments are used to steer the data flow right up to output parameters.

```
out.wps-output-identifier = var.unique-
    identifier
var.unique-identifier = in.wps-input-
    identifier
var.unique-identifier = "value"
```

3. *Bindings* are used to declare processes, and to define their respective endpoint. They are identified and referred to by unique shorthands. Bindings are distinguished in local and remote bindings. The listing below shows a local binding. The runtime environment responsible for execution is instructed not to use exterior interfaces for process calling.

```
bind process wps-compliant-identifier
    to org/shorthand
```

The listing below shows a remote binding. Its endpoint is identified through the classical Uniform Resource Identifier (URI): protocol, host, port and path. Once declared, the processing runtime environment has sufficient information for process discovery and retrieval of process descriptions.

```
bind process http/https, port, path,
    wps-compliant-identifier to org/
    shorthand
```

4. *Execute Statements* are used to bring together references and bindings, and to finally execute WPS processes. As the listing below shows, bindings are referred to by their shorthand. The execute statement itself is divided into two sections. The *with section* for input parameterization and the *store section* for output parameterization. Unlike assignments, these sections are used to connect given runtime references with process specific parameter.

```
execute org/shorthand
  with
    in.example as INPUT1
    var.example as INPUT2
  store
    RESULT1 as out.result1
    RESULT2 as var.variable
```

Regarding further iterations, it is likely that the aspect of readability becomes more important when dealing with more verbose syntax elements. In order to maintain simplicity, one strategy at hand is to provide a minimum syntax and default values. This means that the use of further language elements and attributes is considered to be optional. Finally, this concept implies certain default behaviors within the runtime environment; e.g. the initial handling of errors or timeouts. However, further language elements such as assertions for providing more graceful exits are conceivable.

At the present time, the language does not explicitly support parallel execution. Although, based on the variables used, parallel operations can be identified and utilized by the runtime environment. Language elements that set up the runtime environment's behavior, like switching features on and off, are still to be considered.

3.2 Implementation

With the publication of this article, an early prototype of the RichWPS Server and the needed processes for evaluating the concept has been realized. Its implementation is based on Java as programming language, the Xtext framework[2] for generating parsers for ROLA-scripts. A further component, a custom runtime environment or orchestration engine, is subject to current development by the company Disy Informationssysteme GmbH. It is embedded within the open-source 52° North WPS Server. Workflow definitions and interface contracts are deployed using an adaptation of the WPS-T draft.

The prototype clearly puts emphasis on handling process-chains, which are derived from the given use cases. Regarding the language and the runtime environment, the key concept of local and remote processing has been realized. The support of different datatypes is based on the generator- and parser- implementation of the 52° North WPS Server. Still, addressing the implementation of the orchestration engine, a major issue will be datatype conversion and the alignment between the introduced components. Interpreting a ROLA workflow spans a context in which resolved references and variables need to be stored at least temporarily. The above shown language is based on dynamic datatype detection. This aspect serves the purpose of readability, but confronts the realizing orchestration engine with challenges. In context of the Web Processing Service the amount of major datatypes tend to be limited at a first glance, though dealing with specific subtypes could become complicated. A more static typecasting was considered but overruled. The reliable handling of datatypes and subtypes is considered to be done by included processes. However, depending on the runtime environment's implementation, a check whilst execution is made possible by performing discovery and adhering to the WPS interface and process contracts.

Pending on the implementation, once loaded information and transmitted data can be kept within the executing system. That way the transcoding of data can be kept to a minimum, in some cases.

The 52° North WPS Server enables developers to provide fundamental processes by using different programming languages and techniques[3]. By using the framework's internal interfaces, the current implementation is agnostic to that circumstance. This empowers the service provider to uti-

[2]`http://xtext.org`
[3]e.g. Java, R, Python

lize given in-house expertise in order to provide fundamental processes.

4. MACROPHYTE ASSESSMENT - A USE CASE

The concept of RichWPS orchestration environment was designed bearing in mind the idea of application in public administration. The respected authorities are the Federal Waterways Engineering and Research Institute (BAW), and Schleswig-Holstein's Government-Owned Company for Coastal Protection, National Parks and Ocean Protection (LKN). In the context of the recent setup of the MDI-DE SDI for public information and reporting [14, S. 171ff.], one of RichWPSs' aims is to explore how WPS can add value to current SDIs by means of WPS orchestration. The principles and use of orchestration have been proofed times before [8]. The following use case is intended for the demonstration and verification of the technical implementation and the suitability for related scenarios. Besides, required/lacking features for accurate application in practice are identified in this realization.

4.1 Technical implementation

In a first step, a use case scenario provided by LKN is realized. In this scenario a makrophyte assessment is setup in which the distribution of algae and sears is analyzed aiming at an assessment of the state of the Wadden Sea [5]. The scenario has been realized earlier as a tightly integrated WPS process which is used for reference. For validation purposes, it is disassembled into various smaller processes and recomposed using RichWPS orchestration.

Finding the breakpoints is a major issue in this task and requires knowledge about the greater environment of application. However, this task is not considered to be done by orchestration users. The result may serve as an impression of which kind of processes may be requested in the future by LKN. The original process is subdivided into the five processes: 1) selectReportingArea, 2) selectMSRLD5, 3) selectTopography, 4) intersect 5) characteristics.

The resulting processes are deployed on the OE WPS server and can be used independently. In a next step the workflow process is created by composing the processes with the ModelBuilder. Then, the process description is deployed on the OE. Figure 1 demonstrates the model and listing 1 shows the corresponding ROLA script.

Figure 1 shows the graphical model of the composition. The model is to be read from the top to the bottom. Inputs and outputs of the workflow are displayed as black boxes with a letter indicating the associated data type, namely "L" for literal data, "B" for bounding box data and "C" for complex data. Inputs are at the top of the model and outputs are at the bottom. The remaining white boxes in the middle represent the involved processes with their individual inputs and outputs. The connecting lines indicate the data flow between the processes and inputs and outputs.

Listing 1: ROLA script for LKN use case scenario

```
bind process net.disy.wps.lkn.processes.
    SelectReportingArea to lkn/selreporting
bind process net.disy.wps.lkn.mpa.processes
    .MSRLD5selection to mpa/selmsrld5
bind process net.disy.wps.lkn.mpa.processes
    .SelectTopography to mpa/seltopo
bind process net.disy.wps.lkn.mpa.processes
    .Intersect to mpa/intersect
bind process net.disy.wps.lkn.mpa.processes
    .Characteristics to mpa/characteristic

var.area = "NF"
execute lkn/selreporting
        with
  var.area as area
  in.reportingareas as reportingareas
        store
  reportingarea as var.reportingAreasNF

var.area = "DI"
execute lkn/selreporting
        with
  var.area as area
  in.reportingareas as reportingareas
        store
  reportingarea as var.reportingAreasDI

(code ommited)

execute mpa/characteristic
        with
  var.relevantYears as relevantYears
  var.existingTopographyYears as
      existingTopographyYears
  var.intersectionTidelandsReportingAreasNF
       as
      intersectionTidelandsReportingAreasNF
  var.intersectionTidelandsReportingAreasDI
       as
      intersectionTidelandsReportingAreasDI
  var.relevantSeagras as relevantSeagras
  var.relevantAlgea as relevantAlgea
  var.reportingAreasNF as reportingAreasNF
  var.reportingAreasDI as reportingAreasDI
        store
  mpbResultGml as out.mpbResultGml
```

Listing 1 displays an excerpt of the corresponding ROLA script. It lists the sequentialized execution steps with the port mapping. The bindings at the beginning of the script enable referencing and repeated use of the bound processes. As figure 1 already indicates, the processes *lkn:selectReportingArea* and *mpa:intersect* in the scenario at hand could be re-used. Also, the ROLA script demonstrates the assignment of static literal data to variables and their use in complexity hiding within the orchestration model.

The ROLA workflow description is offered by the OE as a regular WPS process ready to be executed by any WPS client already working with the server. Offering the workflow as a process also permits its use in further composition leading to nested workflows. A process for the generation of a report in PDF is already at hand and can be added in a next level surrounding workflow.

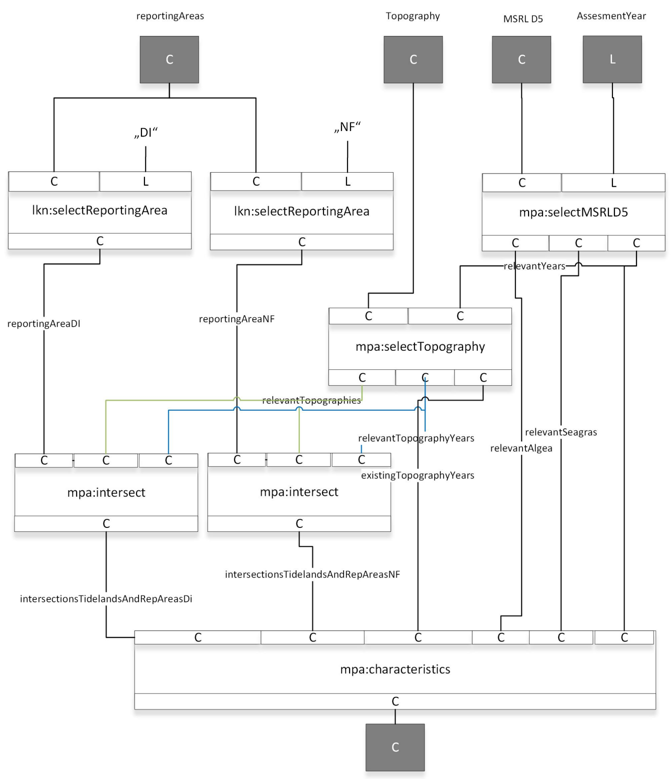

Figure 1: Composition model for macrophyte assessment use case scenario

Executing the process results in the delivery of the expected return values.

4.2 Trends and Shortcomings

Though resulting in the correct results, the amount of time needed for execution is increasing at about 80% in relation to the tightly integrated process. The deviation is mostly caused by additional time for data transport in combination with parsing and generation of exchanged messages; a circumstance which can part-wisely be considered as implementation detail and can be improved by different means. Considering bigger amounts of data, additional factors, which include hardware and software alike, can affect the overall processing time. This indicates the need for further research to address possible enhancements individually. Potential fields are the granularity of the processes or a more efficient internal handover of data.

In hindsight, challenges are found in determining granularity and re-use ratio of resulting processes. However, the process in question sets preconditions to breakpoints, so unchaining of the original process is rather pragmatic. For a proper and comprehensive orchestration the proceedings should go the other way around. Therefore, further attention is to be paid to a thoroughly designed model of processes that are yet to identify through analysis of further scenarios. At the moment there are two levels of re-use characterizing the processes at hand. Some processes like lkn:selectReportingArea can be reused within LKN other processes can most likely be reused in just this scenario like the whole rest. A further level providing basic geospatial operations is desirable. New implementations should be realized from the beginning with the application of WPS in mind.

At this point special attention has to be paid to the newly resulted interfaces of the various processes causing additional effort for implementing new parsers and generators. In order to keep it at an acceptable level, in the future static resources could be added along the ROLA script as it is happening at the moment with static literals. Also, a company policy defining a set of complex data types for common use helps to reduce the development of further and further parser-generator pairs.

For closer integration into existing SDIs, RichWPS should be extended with support for interaction with further OWSs and original web services. This will be reevaluated in the second use case.

5. CONCLUSION AND FURTHER RESEARCH

The work at hand shows a way of describing geospatial workflows by means of an individual domain-specific language. It presents an appropriate orchestration engine and concept. The orchestration environment targets orchestrating WPS with its processes. A central concept, the description of compositions, with a script optimized for its field of application has been shown. Although a custom language is used, interoperability and re-usability is maintained by deploying workflows on an orchestration engine that also works as a WPS server. Publishing a composed workflow is achieved by means of conventional WPS processes, which are ready for execution through a well defined interface.

By dynamically deploying composed workflows on a server, administrative overhead that occurs when confronted with new tasks can be minimized. Using a DSL provides an opportunity to specifically address further needs in the area of application. For the purpose of this project, a custom language outweighs the use of certain WS-standards. The current implementation serves as testbed for evolving the idea, the custom languages and runtime mechanism alike.

For a check up with reality and proof of concept, a real life use case is implemented by means of an early version of the environment. It is reasonable that more sophisticated use cases demand a higher degree of workflow-control such as loops and explicit exception handling, and more complex algorithms, including for instance options to connect to different standard services of the OGC standards program. Because of the all-in-all usability depending on the collaboration of all involved software components, more complex use cases will further evaluate the concept. At the present time, it can be determined that a thoroughly defined policy of common processes and data types is a requirement for successful application.

6. ACKNOWLEDGEMENT

This work was made possible by the German Ministry of Education and Research (BMBF) (RichWPS) and the project partners, the Federal Waterways Engineering and Research Institute (BAW), and Schleswig-Holstein's Government-Owned Company for Coastal Protection, National Parks and Ocean Protection (LKN) as well as Disy Informationssysteme GmbH.

References

[1] W. A and A. Zipf. Web service orchestration of OGC web services for disaster management. *Geomatics Solutions for Disaster Management*, 2007.

[2] A. Akram, D. Meredith, and R. Allan. Evaluation of BPEL to Scientific Workflows. 1:269–274, 2006.

[3] J. Brauner and T. Foerster. Towards a research agenda for geoprocessing services. *12th AGILE International Conference on Geographic Information Science 2009*, 1(OGC 2007):1–12, 2009.

[4] J. de Jesus, P. Walker, M. Grant, and S. Groom. WPS orchestration using the Taverna workbench: The eScience approach. *Computers & Geosciences*, 47:75–86, Oct. 2012.

[5] T. Dolch, C. Buschbaum, and K. Reise. Seegras-Monitoring im Schleswig-Holsteinischen Wattenmeer 2008. 2009.

[6] M. Fowler. *Domain Specific Languages.* Addison-Wesley Professional, 1st edition, 2010.

[7] INSPIRE. Directive 2007/2/ec of the european parliament and of the council of 14 march 2007 establishing an infrastructure for spatial information in the european community (inspire). 2007.

[8] E. Ivanova. Orchestrating Web Services–Standards and Solutions. *" Mathematics, Informatics and Computer Sciences"-St.*, 2006.

[9] N. Josuttis. *Soa in Practice: The Art of Distributed System Design.* O'Reilly Media, Inc., 2007.

[10] C. Kiehle. Business logic for geoprocessing of distributed geodata. *Computers & Geosciences*, 32(10):1746–1757, Dec. 2006.

[11] M. Müller, L. Bernard, and D. Kadner. Moving code – Sharing geoprocessing logic on the Web. *ISPRS Journal of Photogrammetry and Remote Sensing*, Mar. 2013.

[12] S. Nativi, P. Mazzetti, and G. N. Geller. Environmental model access and interoperability: The GEO Model Web initiative. *Environmental Modelling & Software*, 39:214–228, Jan. 2013.

[13] OGC. Ogc web processing service (wps) interface standard. 2007.

[14] A. Rieger, J. Kohlus, and K.-P. Traub. Automatisiertes webbasiertes Verfahren zur ökologischen Bewertung von Makrophyten im Schleswig-Holsteinischen Wattenmeer. *Geoinformationen für die Küstenzone, Band 4*, 2013.

[15] B. Schaeffer. Towards a transactional web processing service. *Proceedings of the GI-Days, Münster*, 2008.

[16] B. Stollberg and A. Zipf. Development of a WPS Process Chaining Tool and Application in a Disaster Management Use Case for Urban Areas. *"Proceedings of the Urban Data Management"*, 2009.