Dissertation

Service-oriented Geoprocessing in Spatial Data Infrastructures

by
Dipl.-Geogr. Matthias Müller

a dissertation submitted for the degree
Doctor of Natural Sciences

Principal Supervisor:	Prof. Dr. rer. nat. Lars Bernard
Associate Supervisors:	Prof. Dr. rer. nat. Andreas Wytzisk
	Prof. Dipl.-Phys. Dr.-Ing. habil. Dirk Burghardt

Faculty of Environmental Sciences
Technische Universität Dresden
Germany

2016

Declaration of Conformity

The conformity of this copy with the original dissertation on the subject
Service-oriented Geoprocessing in Spatial Data Infrastructures
is hereby confirmed.

Dresden, 25 April 2016.

Matthias Müller: *Service-oriented Geoprocessing in Spatial Data Infrastructures*

Supervisors:

Prof. Dr. rer. nat. Lars Bernard
Prof. Dr. rer. nat. Andreas Wytzisk
Prof. Dipl.-Phys. Dr.-Ing. habil. Dirk Burghardt

Date of defense: 11 March 2016

Faculty of Environmental Sciences, Technische Universität Dresden, Germany

Preface

This work is a thesis by publication. The contributing journal articles, short papers, and manuscripts are listed below.

Journal publications (copies included)

Müller, Matthias; Bernard, Lars; Brauner, Johannes: Moving Code in Spatial Data Infrastructures – Web Service Based Deployment of Geoprocessing Algorithms. In: Transactions in GIS, 14(S1), 2010, pp. 101–118.

Müller, Matthias; Bernard, Lars; Kadner, Daniel: Moving code – Sharing geoprocessing logic on the Web. In: ISPRS Journal of Photogrammetry and Remote Sensing, 83, 2013, pp. 193–203.

Müller, Matthias: Hierarchical Profiling of Geoprocessing Services. In: Computers and Geosciences, 82, 2015, pp. 68–77.

Official standard (summary included)

Müller, Matthias (Ed.), Pross, Benjamin (Co-Ed.): OGC WPS 2.0 Interface Standard. Published by the Open Geospatial Consortium, 2015, OGC document number 14-065.

Related short papers (not included)

Kadner, Daniel; **Müller, Matthias**; Brauner, Johannes; Bernard, Lars: Konzeption eines Marktplatzes für den Austausch von Geoprozessierungsimplementierungen. In: gis.SCIENCE, 25(3), 2012, pp. 118–124.

Müller, Matthias; Wiemann, Stefan; Grafe, Bernd: A framework for building multi-representation layers from OpenStreetMap data. Proceedings of the 15th ICA Workshop on Generalisation and Multiple Representation, 2012, Istanbul, Turkey.

Müller, Matthias: Hierarchical process profiles for interoperable geoprocessing functions. Vandenbroucke, D.; Bucher, B.; Crompvoets, J. (Eds.), Proceedings of the 16th AGILE Conference on Geographic Information Science, 2013, Leuven, Belgium.

Henzen, Christin; Brauner, Johannes; **Müller, Matthias**; Henzen, Daniel; Bernard, Lars: Geoprocessing Appstore. In: Bação, F.; Santos, M.Y.; Painho, M. (Eds.), Proceedings of the 18th AGILE Conference on Geographic Information Science,2015, Lisbon, Portugal.

Contents

List of Figures

List of Tables

Listings

Glossary

.NET	A software framework
API	Application programming interface
BMVJ	Bundesministerium der Justiz und für Verbraucherschutz
COM/COM+	Component Object Model / Component Services
CORBA	Common Object Request Broker Architecture
DKRZ	Deutsches Klimarechenzentrum
EC	European Commission
ECMA	European Computer Manufacturers Association; an international standards organisation
EJB	Enterprise JAVA Beans
ESRI	Environmental Systems Research Institute
FME	Feature Manipilation Engine; a software product by Safe Software
GeoJSON	A data format for simple geographical features, based on JSON
GEOSS	Global Earth Observation System of Systems
GIS	Geographic Information System
HTTP	Hypertext Transfer Protocol
IDL	Interface Description Language
IETF	Internet Engineering Task Force
INSPIRE	Infrastructure for Spatial Information in the European Community
ISO	International Organization for Standardization
JAVA	A programming language
JSON	JavaScript Object Notation
MathML	Mathematical Markup Language
Mime type	Multipurpose Internet Mail Extensions type; a widely used internet media type
O&M	Observations and Measurements
OCCI	Open Cloud Computing Interface
OGC	Open Geospatial Consortium
OSGi	OSGi Alliance (formerly the Open Services Gateway initiative)
OMG	Object Management Group
OWS	OGC Web Service Common; a standard
REST	Representational State Transfer; an architecture for distributed computer systems
SDI	Spatial Data Infrastructure
SensorML	Sensor Markup Language
SOA	Service-oriented Architecture

SOAP	Simple Object Access Protocol
SOS	Sensor Observation Service
SWE	Sensor Web Enablement
SWG	Standards Working Group
UML	Unified Modeling Language
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
USGS	United States Geological Survey
W3C	World Wide Web Consortium
WCS	Web Coverage Service
WFS	Web Feature Service
WPS	Web Processing Service
WSDL	Web Service Description Language
WS*	A set of specifications that are used in conjunction with WSDL and SOAP
XML	eXtensible Markup Language

1. Introduction

Geoprocessing functions are vital components in Geographic Information Systems (GIS; Goodchild 2002; Worboys and Duckham 2004). Examples range from simple geometric operations, classifications or schema transformations to more demanding operations such as interpolation, watershed computation, generalisation or network analysis. They are either used in a stand-alone fashion for simple analyses or as building blocks for larger workflows (deSmith et al. 2007; Longley et al. 2011). Traditionally, geoprocessing functions were delivered by Desktop GIS software in so-called *tool boxes* or as an instruction set that could be used for high-level programming (Albrecht 1996). Although a basic set of similar geoprocessing functions is provided by almost any GIS, there are significant differences in the tool boxes' contents and it is hardly surprising that GIS professionals often use multiple products to accomplish a particular task or re-implement foreign tools in their own software system.

With the transition from monolithic GIS to distributed, network-based systems during the last ten to fifteen years, Spatial Data Infrastructures (SDI) have spurred the idea of geoprocessing Web services that would no longer require a particular GIS installation but could rather be invoked over a network (Mineter et al. 2003; Nebert 2004; Friis-Christensen et al. 2007). The reuse of such well-defined and well-tested third-party services provides opportunities to increase productivity as well as credibility and confidence in the correctness of the produced results (Davis and Anderson 2004; Marshall et al. 2010). Provided that interoperability issues can be overcome, geoprocessing Web services may be invoked by arbitrary software systems and clients. Unfortunately, geoprocessing Web services have not yet delivered this vision. In fact, there are technological, organisational and security related obstacles, which will be discussed in this thesis, that impede or prohibit the use of geoprocessing Web services in many applications.

Geoprocessing Web services, however, have revitalised the debate on distributed and service-oriented geoprocessing in general and revealed several interoperability issues with current GIS. By definition, a service is a

> "... mechanism to enable access to one or more capabilities, where the access is provided using a prescribed interface and is exercised consistent with constraints and policies as specified by the service description" (W3C 2006, 12).

For GIS and geoprocessing, these "capabilities" may refer to any geoprocessing function that helps to compute new results from existing data, such as geographic

objects, networks, or remote sensing imagery. GIS processionals and analysts make use of such services in their everyday work but are limited by the available functionality of their own software systems. By making an analogy to SDIs, which to date attempt to leverage ubiquitous access to spatial data, this thesis investigates a broad range of provisioning approaches for geoprocessing functions. The following sections review the particular drawbacks of Web services for geoprocessing and consider a broader range of provisioning approaches for geoprocessing services. Subsequently, two major research challenges are identified which are addressed by the publications that contribute to this thesis (chapters 2-5). The findings from theses individual contributions are recapitulated in chapter 6 and reviewed in the context of the stated research challenges.

1.1. Terminology

In literature, the terms *GeoComputation* and *geoprocessing* are often used interchangeably. In SDI-related publications there seems to be a bias towards using *geoprocessing* while environmental modelling papers seem to prefer *GeoComputation*.[1]

The body of source material on *geoprocessing* is diverse and there is no single common definition. Without doubt, the term's connotation has been influenced by the widespread use of ESRI's ArcGIS software which provides a geoprocessing toolbox to process and analyse geographic data (ESRI 2015). Wade and Sommer (2006, p. 89) define *geoprocessing* as the manipulation of spatial data and speak of *geoprocessing operations* that transform input data sets into output datasets.

In ISO (2006) the term *geoprocessing* is used as an umbrella term for the integrated use of geographic data and covers both data access and processing. It is stated that

> "... geodata users can query remote databases and control remote processing resources, and also take advantage of other distributed computing technologies, such as software delivered to the user's local environment from a remote environment for temporary use" (ISO 2006, 4).

Geographical data processing in terms of mathematical calculation or data transformation is performed by so-called *geographic processing services* which are organised into the four categories spatial, thematic, temporal, and metadata processing. *Transformation services* for spatial data are also defined by the directive on Infrastructure for Spatial Information in the European Community (INSPIRE; EC 2007, 2010a). These services are either used to convert geospatial data between different coordinate reference systems (EC 2010b) or perform transformations between data schemas (Howard et al. 2010).

Based on the definition of *geoprocessing* given by Wade and Sommer (2006), Brauner (2015) developed his notion of a *geooperator* which is

[1] Statement based on the results of Google's search engine, so there could be a chance of bias.

> "... a distinct, well defined and usually implemented piece of software serving a particular purpose for geospatial analysis or transformation.
>
> If a geooperator is well-defined in the literature but not implemented in any GIS, it is defined as an abstract geooperator. Geooperators have an arbitrary number of input and output operands, most of them spatial. Additional non-spatial parameters can be defined to control the operator, e.g. a buffer distance" (Brauner 2015, p. 20).

The *geooperator* provides a descriptive framework that formalises empirical knowledge about existing *geoprocessing operations* (Wade and Sommer 2006), i.e. software artefacts. An extensible set of *perspectives* permits the classification of a *geooperator* according to properties such as input and output data types, typical applications, or their availability in legacy GIS software.

The term *GeoComputation*, which is closely related to geoprocessing, appeared in the second half of the 1990s. Its definition is still a subject of discussion and definition struggles have a long history (Couclelis 1998; Longley 1998; Gahegan 1999; Openshaw 2000; Cheng et al. 2012; Gahegan 2015). Its scope covers quantitative geographical analysis and modelling and is thus a bit broader than *geoprocessing*. One of the most pressing challenges in *GeoComputation*, which is also relevant for this thesis, is the provisioning of reliable, well-performing tools and operators for data analysis. In particular, there is

> "... a gap in knowledge between the abstract functioning of these tools (which is usually well understood in the computer science community) and their successful deployment to the complex applications and datasets that are commonplace in geography. It is precisely this gap in knowledge that GeoComputation aims to address" (Gahegan 2015).

This thesis and the related publications use the term *geoprocessing* which means the processing of spatial or geo-referenced data that relate to conceptual data models and encoding formats from geoscience and neighbouring disciplines (cf. chapter 4).

The contributions included with this thesis represent the current state of related research, academic discussion and insight at the time of writing. The evolution of ideas has inevitably led to some naming inconsistencies during the discourse. The following terms and definitions are used in this thesis and in the latest publications.

Geoprocessing function. A function that serves a particular purpose in geographical analysis or transformation of geographical data which derives a set of output data from a given set of input data. The inputs and outputs usually relate to data models and schemas in geospatial applications and neighbouring disciplines.

This definition is largely consistent with the concept of the *geooperator* (Brauner 2015) or *geoprocessing operation* (Wade and Sommer 2006). Furthermore, it treats such operations very much like functions in mathematics, i.e. relations that provide

a unique mapping $p : X \rightarrow Y$ from the set of inputs (domain X) to the set of outputs (co-domain Y).

With regard to modelling and simulation, a *geoprocessing function* is also consistent with the definition of a *function specified system* (FSS; Zeigler et al. 2000, p. 116) that is used to create static, stateless and thus time-independent system models. The previously mentioned bulding block models (cf. deSmith et al. 2007; Longley et al. 2011) usually fall into the category of FSS since they are composed from chained geoprocessing functions.

Software component. The Unified Modelling Language (UML) defines a component as a

> "... modular unit with well-defined interfaces that is replaceable within its environment"

where *well-defined* means that the interface is completely described in syntax and semantics. UML further distinguishes

> "... logical components (e.g. business components, process components) and physical components (e.g. EJB components, CORBA components, COM+ and .NET components, WSDL components, etc.), along with the artifacts that implement them and the nodes on which they are deployed and executed. It is anticipated that profiles based around components will be developed for specific component technologies and associated hardware and software environments" (OMG 2005, p. 139).

Szyperski and Pfister (1997) emphasise on the composition aspect of components and envision the emergence of component markets where producers and consumers of software components come together. With this background they define a component as a

> "... unit of composition with contractually specified interfaces and explicit context dependencies only. A software component can be deployed independently and is subject to composition by third parties" (Szyperski and Pfister 1997, p. 130).

Depending on their purpose, components may not only incorporate computing instructions (or code) but also contain static data required for their operation.

This thesis uses the term *software component* to refer to *physical components* as defined by OMG (2005) which have explicit context dependencies on particular software and hardware environments.

Computing instructions or Code. A series of commands or expressions that constitute an implementation of a processing function. Computing instructions may be codified in machine code, byte code, a scripting language, or in a domain specific or algebraic language (Crnković et al. 2011).

Moving Code paradigm. This paradigm has been discussed as an alternative to classical client–server processing where large amounts of data are transferred between geoprocessing Web services (Bernard et al. 2005; Friis-Christensen et al. 2007; Brauner et al. 2009). It represents a subset of the code mobility paradigms defined by Carzaniga et al. (1997, cf. section 1.2) which involve the migration of computing code between different locations in a computer network.

This thesis defines the moving-code paradigm as an approach where *computing instructions or code*, that represent a particular *geoprocessing function*, are moved between different network locations in a computer network. The *moving code* approach serves two purposes:

- It strives to perform geoprocessing tasks more efficiently by moving the computing instructions closer to the other participating resources (data, computing resources) involved in these tasks.
- It enables and facilitates the sharing of software components that represent geoprocessing services which are intended to be shared and reused within SDI.

The development of new approaches for sharing and exchanging software implementations of geoprocessing functions is a central topic in this thesis. In the older publications, the terms *code package* (Müller et al. 2010) and *moving-code package* (Müller et al. 2013) were used to express a portable and well-defined implementation of a particular geoprocessing function. The more generic term *[portable] software component* is used in later publications.

Geoprocessing service. An implementation of a well-specified, interoperable, and reusable geoprocessing function that receives and generates data in a common encoding format and can be readily embedded into a workflow (cf. chapter 4). A *geoprocessing service* is not necessarily a Web service (although it may be typically published as a Web service) but refers to any implementation that complies with the basic principles of service-oriented design (Erl 2007). In particular, a *geoprocessing service* has a well-defined interface which is sufficient for its use. Knowledge about implementation details is not required. Compared to a [physical] *software component* a service definition also abstracts from particular runtime environments or implementation technologies and uses a generalised interface description language for cross-platform use.

Geoprocessing Web service. A technology that provides client–server interaction for performing geoprocessing tasks on the internet (cf. chapter 4). A geoprocessing Web service is defined independently from a geoprocessing service. Current standards rather focus on Web based protocols for client–server interaction rather than the provision and use of well-defined implementations.

Web Processing Service (WPS). An implementation standard for geoprocessing Web services, published by the Open Geospatial Consortium (OGC; OGC 2007c,

2015a). The WPS standard provides an interface definition language for *processes*, whose definition is quite consistent with the notion of geoprocessing functions as provided in this thesis, and a Web service protocol for client–server geoprocessing.

1.2. Problem Statement

In an ideal networked SDI, a user has ubiquitous access to the available data *and* to all available geoprocessing functions. The achievement of both goals still faces various research challenges (Bernard et al. 2013), but technologies and solutions for data access surely have an edge over technologies that enable ubiquitous access to geoprocessing functions. Evidence is found in various operational infrastructures[2], international standards[3], as well as legal regulations which largely promote open access to data. Various bodies attempt to standardise data exchange formats and establish interoperable data access services. For instance, national and European legislation require standards-based access to environmental data (EC 2007; BMVJ 2009).

Some data access services can provide primitive ad-hoc processing capabilities on tightly coupled data. In the geospatial domain, Web Coverage Services (WCS, OGC 2012) with extensions for sub-setting, processing etc. (OGC 2015c) or Web Feature Services (WFS, OGC 2014d) with Filter Encoding support (OGC 2014a) for geospatial queries are typical examples. Advanced analysis functions are sometimes provided by Geoportals and Geospatial Web applications. These are usually canned tools and services that operate on a pre-defined database and provide end users with interactive analysis functions (Hofer 2015).

Apart from these tightly coupled solutions, the capabilities to share, exchange and reuse implementations of geoprocessing functions are rather underdeveloped in current SDI. Web services which offer geoprocessing functions have been around for a while but they are largely used in research or within institutions and have not gained much visibility compared to publicly accessible geospatial data and mapping services (Lopez-Pellicer et al. 2012). There are several arguments that could explain this fact:

1. There are technological boundaries that constrain or prohibit the application of client–server paradigms for large amounts of data. Network bandwidth is often limited and reliable connectivity in larger computing workflows may be hard to ensure. If reliability and fail-safe operation is a decisive criterion, Web services might better be avoided. If performance matters, the overhead of the Web service invocation due to the required data encoding and decoding as well as transfer over the network can be a limiting factor.

[2] E.g. the Global Earth Observation System of Systems, GEOSS, or the Earth observation programme Copernicus of the European Union

[3] E.g. provided by the International Standardisation Organisation (ISO; http://www.isotc211.org/, accessed 2015-06-16) or the Open Geospatial Consortium (OGC; `http://www.opengeospatial.org/`, accessed 2015-06-16)

2. If confidential or sensitive data is to be processed, its transfer over a network might be problematic. Under such circumstances, the use of Web services for data processing is inappropriate and alternative means must be found to deliver the required processing functions to a closed environment.

3. There is a general lack of infrastructure components (catalogues, for instance) that provide means to discover, search, and retrieve geoprocessing functions (cf. Brauner 2015). This decreases the visibility of existing offerings. Tedious Web based research with generic search engines is often the only option for potential users to discover reusable implementations and services.

4. The operation of publicly available processing services is usually much more costly than the provision of data and mapping services. For the latter category, effective caching and pre-processing mechanisms exist which decrease the costs of operation. For data processing Web services, caching and pre-processing is not applicable and caching rather ineffective. The majority of use cases demands single-time processing of new data where caching of inputs and outputs is of little help.[4]

5. Currently, there are hardly any legal obligations for providing geoprocessing Web services. Access to open data is increasingly mandated or at least encouraged by national and super-national regulations but access to data *processing* services is usually not demanded.

Considering the general benefits of shared access to geoprocessing functions, alternative approaches for distributed geoprocessing are required that avoid the disadvantages of geoprocessing Web services but yet facilitate access to geoprocessing functions in a distributed infrastructure. The identification of other suitable provisioning approaches and their assessment is a goal of this thesis.

As opposed to traditional workstation environments where hardware resources, data, and processing functions were available in the same place, these different resources are usually spread across many nodes in a distributed infrastructure. In an extreme case, a particular analysis workflow might require (cf. Figure 1.1):

- Public and private data from different physical locations,
- Computing code that is only available from a foreign repository, and
- Execution on processing hardware of a public hosting company.

Code mobility is a concept that overcomes the limitations of isolated software systems and has been studied in detail by Carzaniga et al. (1997) and Carzaniga et al. (2007). Both publications define code mobility as

[4] This statement applies to loosely coupled, stand-alone data processing services which are predominantly discussed in this thesis. Some data access services or Web portals provide simple processing capabilities that operate on the provided data. In this case, optimised data structures will, of course, reduce the ad-hoc workload.

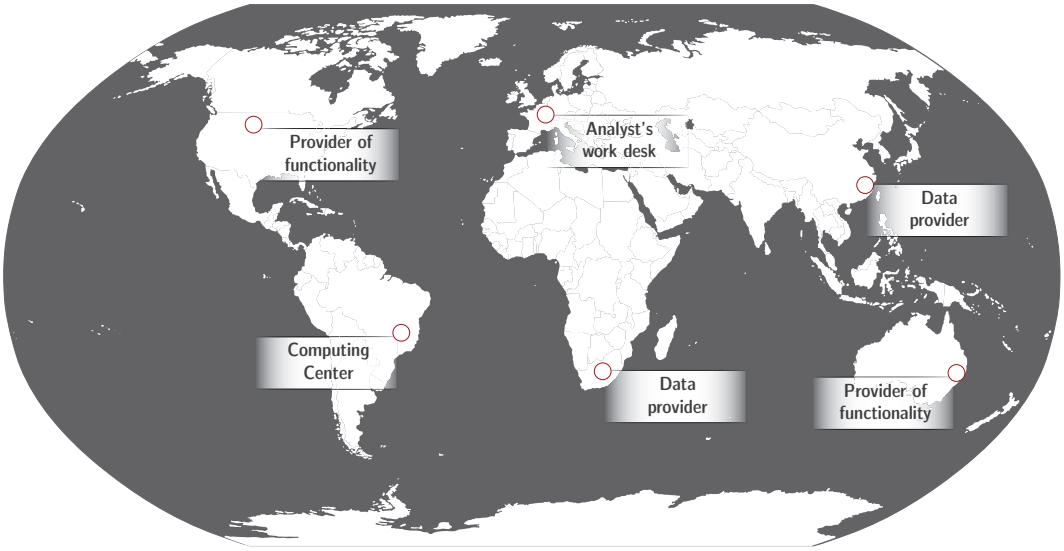


Figure 1.1.: An example for data processing in a globally distributed resource network

> "... the capability to reconfigure dynamically, at runtime, the binding between the software components of the application and their physical location within a computer network" (Carzaniga et al. 1997, p. 22).

The authors define four mobile code paradigms which are summarised in Table 1.1.

Code mobility strives to perform computing tasks more efficiently by moving the executable content (or code) closer to the other resources involved in that task. Resources involved in the efficiency consideration are the locations of input data, the target locations for the computing results, and the location of computing power (Docan et al. 2011). For any of these paradigms there are examples in the context of SDI which shall be briefly discussed.

Client–Server. Less data-intensive computing may be accomplished by invoking common Web services that provide access to data and functionality. ESRI's ArcGIS online platform (and ArcGIS Server instances) provides a so-called geometry service.[5] This service contains a growing set of geometry operations that are frequently used in Web based mash-ups for proximity analyses or the creation of heat maps. The Web application sends its data to the desired geoprocessing service by using standard network protocols and receives the result in a well-known encoding, ready for display. The public use of these services is currently free up to a certain amount of requests. It is primarily intended for occasional computations on negligible data volumes.

[5] `http://www.arcgis.com/home/item.html?id=2e18b487043641538f02028cc2495c0e`, accessed 2015-06-16

Table 1.1.: Mobile code paradigms according to Carzaniga et al. (1997)

Paradigm	Description
Client–Server	The client has neither the resources nor the know-how to perform a particular computing task (thin client). It invokes a remote service that provides the input data, functionality and hardware resources. This service performs the computation on the client's behalf and delivers the result.
Remote Evaluation	The client owns the functionality/know-how to perform a particular computing task but lacks the required resources (input data and hardware) for execution. It transfers the required functionality to a remote service that provides access to the missing resources. The remote service executes the functionality on the client's behalf and delivers the result.
Code on Demand	The client owns the resources (input data and hardware) to perform a particular computing task but lacks the proper know-how. It requests a copy of the missing functionality from a remote service. The client applies the functionality in its private environment and stores the result.
Mobile Agent	The client owns functionality and input data but lacks the hardware resources to perform a particular computing task. It replicates the required functionality and input data in a remote location that provides the missing hardware resources. After the functionality is applied to the data in the remote infrastructure, the results may be migrated back to the client's private environment.

Remote Evaluation. The preparation of code and software for remote evaluation of computing-intensive tasks has been practised for a long time in the geospatial sciences. The task was usually prepared in a local environment with limited resources, where researchers have gathered the required data and code. Once completed, this work was manually transferred to a more capable workstation computer or a cluster which would actually run the task. Portable code that can be run on a foreign machine is an essential prerequisite here.

Another case for remote evaluation can be made in the context of data protection. Private or personal data stored in research data centres needs to be processed in a tightly secured or even isolated network with restricted or no connection to public external networks. Researchers who need to access the data must use dedicated workstations provided by the research data centres for filtering and processing the protected data. There is no alternative to moving the processing instructions to a particular workstation in that secured infrastructure and perform all computations

intra muros. After a final check by privacy experts, the (anonymised) result data may be cleared for external use and publication.

Processing big volumes or massive amounts of data is another class of use cases where remote evaluation is beneficial. The Copernicus platform and infrastructure is expected to host and generate remote sensing data from European space missions at an unprecedented scale (Soille and Marchetti 2014). The new generation of instruments provide an extended swath, increased spectral and radiometric, and offer a shorter revisit time (Vanhellemont and Ruddick 2014). Thus the data volume of individual scenes easily exceeds the gigabyte threshold, amounting, for instance, to 2.5 terabytes per day from a single Sentinel 1 satellite (Soille and Marchetti 2014). Delivering the data to an analyst's workstation or invoking the data from a remote Web service is still an option but it has three disadvantages:

1. The data download is time consuming and uses a lot of bandwidth at both client and server side,
2. The client needs to provide significant storage to store the input data temporarily, and
3. More capable hardware is required to process the huge satellite scenes which may overstrain the client's resources.

Therefore, the European Space Agency evaluated the potential of hosted computing. This feature of the Copernicus infrastructure provides storage and computing resources to the users of the data and they may deploy their own algorithms and software code in so-called sandboxes (Almeida et al. 2014). Instead of moving around data, code is moved to the data centre's infrastructure and the computation is done close to the data.

A more generic case for this paradigm is made by NIST (2014, p. 6) which specifies a reference architecture for Big Data interoperability. The current draft foresees generic processing services which are operated by application providers. For a particular task, these services receive analytic code from users which is then deployed and executed in a scalable computing environment. In this setting the use of network bandwidth is reduced to the transmission of analytic code and result data. A bandwidth-intensive transport of input data for the purpose of a single computation is avoided as far as possible.

Code on Demand. Despite the centralisation of computing power in data centres or computing clusters, GIS workstations still play an important role and are used by many specialists in their daily work. In many scenarios it provides the best balance between flexibility and computing speed. The toolboxes of desktop GIS contain implementations of many pre-defined geoprocessing functions. In the past, toolboxes used to provide a fixed set of tools which was only updated during major software releases. Meanwhile, some software vendors have started to decouple the toolboxes' contents from product cycles and provide on-demand toolbox updates.

Safe Software's Feature Manipulation Engine (FME) operates the FME store, a publishing platform for new or very special processing tools (called *transformers*). Users of the FME workbench may use this feature to query new functionality on demand or periodically update their local installation with new tools from a Web based distribution platform.

The ever evolving set of computing functions developed for the geospatial sciences can hardly be put into a conclusive set of operations. This functionality is usually developed across multiple sub-disciplines in environmental and Geoinformation sciences or neighbouring disciplines. Natives from these disciplines are natural experts in functionality and continue to develop and publish new algorithms and compliant software implementations. Code on demand helps to provide interchangeable self-contained software components which are produced, maintained, and updated by domain experts and can be delivered to third-party users (Granell et al. 2010, 2014). An early example of a *code on demand* platform was ESRI's ArcGIS code gallery for the model builder where users could publish new useful ArcGIS tools. Once published, these contributions could be discovered in the portal and downloaded by end users for deployment in their local ArcGIS environment.

Code on demand approaches have also been discussed in conjunction with reproducible research. For instance, Hill et al. (2001) report on the design of a code sharing standard that allows for achieving computational models in digital libraries. The provision of archived and reviewed computing code may not just increase reuse but also enhance reproducibility of scientific activity in the context of eScience (Gray 2009).

Mobile Agent. One of the core settings for the mobile agent paradigm is cloud computing which is increasingly applied in the geospatial sciences. Special consideration to cloud-enabled processing was given by so-called geospatial cyberinfrastructures (Pierce et al. 2010; Yang et al. 2010). The users of such computing environments have access to a potentially unbounded pool of hardware resources. Data and code are shared among computing nodes to solve CPU and memory-intensive tasks. Code mobility is an enabler in this scenario since it allows the dynamic deployment of the required software components on any node in the cloud infrastructure.

Transactional Web Processing Services (WPS-T) which have been discussed by Schaeffer (2008) are another approach to an implementation of the *mobile agent* paradigm. In contrast to the regular *client–server* interaction with a WPS, where the client consumes the static offerings of the remote service, WPS-T allows pushing own code to a WPS server prior to execution. Since the the overall scenario includes code provision, deployment and execution it qualifies qualifies as *mobile agent*, while individual steps are a combination of *remote evaluation* (the code is sent for interpretation to a remote server) and *client–server* paradigms (the client invokes the submitted functionality by calling the updated remote server).

1.3. Service-oriented Geoprocessing

Web services for geoprocessing have been proposed to counter the poor scalability of monolithic GIS software (Friis-Christensen et al. 2007; Kiehle et al. 2007). The use of geoprocessing Web services follows the *publish-find-bind* paradigm (cf. Figure 1.2).

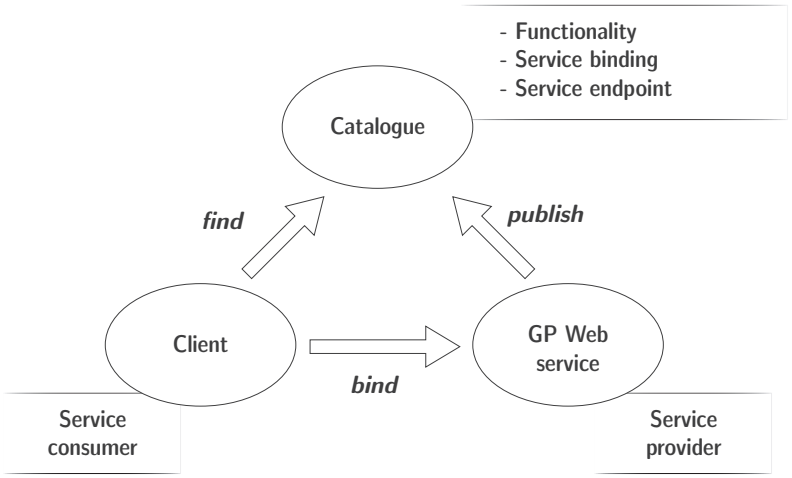


Figure 1.2.: Publish-find-bind with geoprocessing Web services

This paradigm defines the basic roles and interactions between organisations that participate in a Service-oriented architecture (SOA, e.g. Alonso et al. 2004). Service providers supply functionality and service consumers use it. Catalogue operators run service catalogues or registries where service providers *publish* their service offerings and which are queried by service consumers to *find* a service that supplies the desired functionality. This paradigm usually assumes that the same (or similar) functionality is offered by multiple service providers so that service consumer may pick one of those services at random or select one by non-functional criteria such as pricing, availability, or terms of use. Once a service consumer has found a suitable provider he *binds* himself to that service, i.e. he calls that service to perform a particular task.

For data acces and mapping services, the service consumer usually wants access to a particular data set (or subset) or a map within a certain area of interest. For geoprocessing services, the consumer is interested in computing derived data from one or more input data sets according to some well-defined geoprocessing function. WPS or ESRI's geoprocessing service are two technologies that provide geoprocessing Web services. If a service consumer has bound himself to such a service, he sends the input data and awaits the results. Depending on the available bandwidth, the data volume, the provided computing power, and the computing complexity, the whole process may take an arbitrary amount of time. If the data volume is small and the computation simple, client and server communicate synchronously by keeping an open connection between client and server until the result is delivered. For long

running tasks, asynchronous communication is preferred. Here the client submits a single request to the geoprocessing Web service and performs occasional status polls. Once the server has indicated a finished computation, the client may query the result (OGC 2015a).

Studies on geoprocessing Web services have started to appear at the beginning of the 21st century. One of the major issues evolves around the *publish* and *find* activities. To make the publish-find-bind work, catalogues need to provide the following information about a service (Alonso et al. 2004; W3C 2007):

1. The provided functionality, e.g. in terms of a service interface or service contract,
2. The service binding, describing the supported communication protocol, and
3. The service endpoint, which indicates where the service can be accessed.

Service binding and the communication of service endpoints are easy to specify and well covered by existing specifications. A persisting issue is the description of the provided functionality which has become apparent with the emergence of geoprocessing Web services but in fact dates back to early attempts to specify and harmonise geoprocessing functionality in general (Tomlin 1990; Egenhofer and Franzosa 1991; Albrecht 1996). The lack of a common framework to express geoprocessing functionality might be one reason that impedes the proliferation of geoprocessing Web services on the internet. If service consumers are generally unable to specify the required geoprocessing functionality and create appropriate search queries, an infrastructure with isolated geoprocessing Web service is of limited use.

Another set of issues arises from the strong coupling between geoprocessing functionality and computing resources. Operators of public WPS servers must provide significant computing resources to their clients. Data services hardly face this issue since the downloaded data can be reused by the client for a different activity. With geoprocessing Web services, the reuse of functionality requires another invocation of the remote Web service. Bandwidth issues that may occur during the processing of large data sets have already been discussed. In extreme cases, network outages can lead to broken workflows and degrade the stability of depending workflows.

Portable software components that facilitate remote execution have been proposed to avoid these issues. In the SDI research this approach has been largely discussed in relation to data transfer overhead and efficient use of bandwidth. *Moving code* concepts have been proposed to avoid the relocation of large amounts of data in Web based geoprocessing workflows (Friis-Christensen et al. 2007; Brauner et al. 2009). Delivering software components from a remote location to local environments for temporary use is also foreseen by ISO (2006, p. 4). In data-intensive computations it was found that moving data between network nodes was cumbersome and had a significant impact on the whole processing time. In contrast, the transportation effort of computing logic in terms of coded software or processing instructions is

manageable to negligible, and a lot of overall processing time can be saved by moving code instead of data (Müller et al. 2010).

This thesis claims that, in principle, the publish-find-bind paradigm can also be applied to service-oriented code sharing architectures (Kadner et al. 2012; Müller et al. 2013; Henzen et al. 2015). Instead of providing access to computing facilities for geoprocessing, in this setting the service providers supply portable machine code representations of geoprocessing functions which can be run in diverse computing environments (cf. Figure 1.3). Similarly to the provisioning architecture for geo-

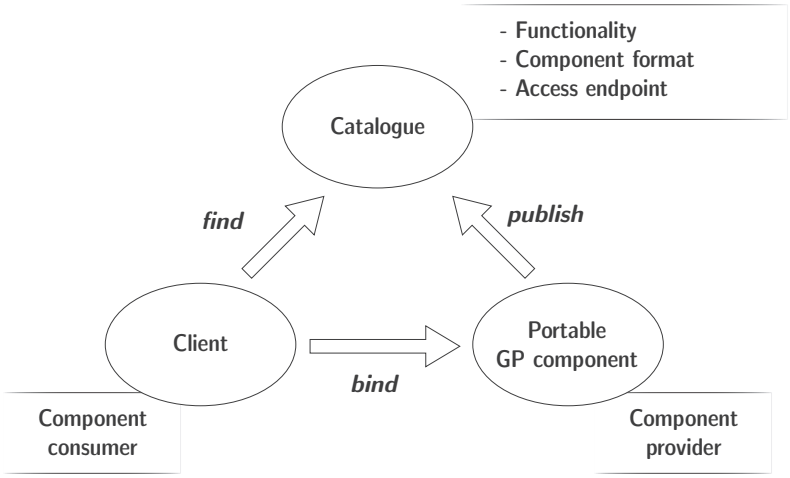


Figure 1.3.: Code on Demand – Publish-find-bind with portable software components

processing Web services, the catalogue operator maintains an inventory of service providers that deliver geoprocessing software components. Consumers may query this catalogue to find a software component that matches their task and search criteria. Similarly to geoprocessing Web services, the provided functionality is a decisive search criterion. Additional aspects of the search query might be the supported runtime environments or software licenses that apply to the software component. Next to functionality, the service binding and the service endpoint are important aspects for Web service invocation. For shared software components these properties are replaced by the component format to ensure interpretability the consumer and an access endpoint that delivers a copy of the software component. Once a consumer has discovered a suitable component, he can download and deploy it in his own environment. The subsequent invocation and execution is not part of the binding process. It happens at a lower level and is covered by the specification of the component format. Since the consumer is in full control of the execution process, status monitoring, error handling, and result retrieval is less complex compared to the Web service invocation. Additional requirements for the consumer arise from the general deployment overhead and the necessity to supply computing resources for execution.

For completeness *ubiquitous availability* shall be mentioned as a third "provisioning approach". In contrast to the other two provisioning options, it is assumed that

a sufficiently large set of harmonised and well-defined geoprocessing functions are provided by any relevant system participating in a distributed SDI. In this scenario, providers and consumers of geoprocessing functions are no longer distinct entities. Computations can be performed in any network location and the decision about the place of computation becomes merely a matter of security constraints, quality of service, and hardware resources allocation. Since many GIS products provide a basic set of processing tools, a functional overlap between the different toolboxes is often silently assumed (Albrecht 1996), but occasional studies revealed minor or major differences in the tools' names, interfaces, and behaviours (Lutz et al. 2003; Fisher 2006). A brief survey of *Buffer* functions in current GIS and spatial databases shows that harmonised functionality and interfaces are the exception rather than the rule (cf. annex A.1). Brauner (2015) has recently examined a broad range of GIS tools and came to a similar conclusion. Due to the diversity of implementations, a user who wants to apply a particular function to his own data or reproduce a colleague's workflow is required to install and maintain a corresponding software configuration on his computer. Obviously, this practice hardly scales with an increasing number of GIS products, spatial databases, and programming libraries for spatial data processing. Furthermore, it does not take into account "new" software components that were developed on top of existing libraries to perform specialised analyses or represent computational models.

The OGC has made an attempt to standardise some basic analysis functions for spatial geometry (OGC 1999; ISO 2004), but this effort was not extended beyond a basic set of *Overlay* (Egenhofer and Frank 1992), *Buffer* and *Convex Hull* operations.[6] For the majority of geoprocessing functions ubiquitous availability paired with coherent behaviour is an unrealistic expectation. With current standardisation efforts it is only met for tiny subset of geoprocessing functions.

1.4. Research Challenges and Contributions

The discussion of provisioning approaches for service-oriented geoprocessing has revealed two major research challenges which are now broken down into a set of detailed issues.

(1) Portable software components for geoprocessing

Web service interfaces and communication protocols, such as those described in the WPS standard, have been established for geoprocessing Web services. Portable software components and interoperable *code on demand* architectures for geoprocessing have yet to be developed. The publish-find-bind model shown in Figure 1.3 suggests that service-orientated architectures might serve as a starting point for sharing

[6] The specifications list seven distinct operations: *Distance*, *Intersection*, *Difference*, *Union*, *Sym-Difference*, *Buffer*, and *ConvexHull*

computing components. This hypothesis is broken down into the following research questions:

- How is code mobility achievable in SDI?
- What would be a suitable component model for exchanging the software representations of geoprocessing functions in an interoperable manner?
- How would a possible code sharing architecture look like and which actors and services participate in a code-sharing environment?

Contributions. The *moving code* paradigm, which discusses the ability to move implementations of geoprocessing functions between computing nodes in an SDI, has been investigated in Müller et al. (2010, reprinted in chapter 2) and Müller et al. (2013, reprinted in chapter 3). The first publication evaluates general patterns for service-oriented geoprocessing and provides initial work towards portable software components for geoprocessing. The second publication continues and extends this work. It contains a comprehensive discussion of the requirements for portable components and develops a code sharing architecture that can be used to offer, consume, and exchange implementations of geoprocessing functions over the internet.

In parallel the idea of an *Appstore* for geoprocessing software was discussed in related publications (Kadner et al. 2012; Henzen et al. 2015). It is scoped as a community platform for sharing and disseminating portable software components for geoprocessing and supplies a processing sandbox for testing and demonstration.

(2) Interoperable interface descriptions for geoprocessing services

A common approach for describing and cataloguing geoprocessing functionality and services is required in any provisioning approach. It is also one of the fundamental challenges posed in the debate on *GeoComputation* (cf. section 1.1). Despite the various research efforts that have addressed this issue (e.g. Lutz 2007; Yue et al. 2007; Zaharia et al. 2008; Fitzner et al. 2011; Maué et al. 2012; Farnaghi and Mansourian 2013), a conclusive solution has yet to be found.

In order to support basic search and retrieval tasks, a feasible approach is required to describe, document, and catalogue geoprocessing services. This approach should be built on top of existing standards for geoprocessing services, enable human users to fully understand the functioning of a geoprocessing service, and permit machines to unambiguously identify a particular geoprocessing function or agree that two services provide the same geoprocessing function. From these requirements the following research questions can be derived:

- How can functional descriptions be implemented in conjunction with existing standards for geoprocessing services?
- What are meaningful granularities at which geoprocessing services can be described and compared?

- How can SDI catalogues be enhanced to support the search and retrieval of geoprocessing functions and services?

Contributions. The possibility to use WPS process descriptions as a common interface description language for arbitrary software artefacts implementing a geoprocessing function has been suggested in Müller et al. (2010, reprinted in chapter 2) and subsequent publications on the *moving code* concept. Finally, in the WPS 2.0 standard (OGC 2015a, summarised in chapter 5), the process description model was separated from the Web service model for exactly this purpose.

An approach for the description of geoprocessing services at multiple granularities has been proposed in Müller (2013) and Müller (2015, reprinted in chapter 4). These papers identify recurring levels of detail at which meaningful descriptions can be obtained, covering both syntax and behaviour of geoprocessing services. The framework is applicable to geoprocessing Web services, portable components, or can be used to harmonise documentations of existing GIS toolboxes at an abstract level. This work had significant influence on the WPS 2.0 specification (OGC 2015a, summarised in chapter 5) which benefits from this approach and provides the respective encodings and conformance classes.

2. Moving Code in Spatial Data Infrastructures – Web Service Based Deployment of Geoprocessing Algorithms

Müller, Matthias; Bernard, Lars; Brauner, Johannes: Moving Code in Spatial Data Infrastructures – Web Service Based Deployment of Geoprocessing Algorithms. In: Transactions in GIS, 14(S1), 2010. pp. 101–118.

Abstract. This article proposes a concept for offering complex geoprocessing functionality in service-based Spatial Data Infrastructures (SDI). Today, geoprocessing in SDI is typically realized in a data driven manner. Applying the suggested *moving code* approach in a case study in the field of Spatial Decision Support proves its applicability. The proposed solution is analyzed and assessed in terms of gained efficiency, performance behavior and support for distributed development of geoprocessing functionality. In data and computation intensive SDI applications the deployment of moving code proves to be beneficial.

Transactions in GIS, 2010, 14(S1): 101–118

Research Article

Moving Code in Spatial Data Infrastructures – Web Service Based Deployment of Geoprocessing Algorithms

Matthias Müller
Geoinformation Systems
Department of Geosciences
Technische Universität Dresden

Lars Bernard
Geoinformation Systems
Department of Geosciences
Technische Universität Dresden

Johannes Brauner
Geoinformation Systems
Department of Geosciences
Technische Universität Dresden

Abstract
This article proposes a concept for offering complex geoprocessing functionality in service-based Spatial Data Infrastructures (SDI). Today, geoprocessing in SDI is typically realized in a data driven manner. Applying the suggested "moving code" approach in a case study in the field of Spatial Decision Support proves its applicability. The proposed solution is analyzed and assessed in terms of gained efficiency, performance behavior and support for distributed development of geoprocessing functionality. In data and computation intensive SDI applications the deployment of moving code proves to be beneficial.

1 Introduction

Geospatial algorithms are a fundamental means to extract information from spatial data. Geographic Information Systems (GIS) provide the user with at least a basic set of atomic operators to analyze spatial and spatio-temporal phenomena. Many GIS also offer modeling environments or APIs (Application Programming Interface) to chain and combine these atomic operators to realize complex algorithms. Besides, algebraic languages, defining atomic functions and supporting the specification of complex algorithms

Address for correspondence: Matthias Müller, Technische Universität Dresden, Department of Geosciences, Geoinformation Systems, 01062 Dresden, Germany. E-mail: matthias_mueller@tu-dresden.de

doi: 10.1111/j.1467-9671.2010.01205.x

on an abstract level, have been developed. Prominent examples are the works of Tomlin (1990) and Egenhofer (1994).

The development of Web-based Spatial Data Infrastructures (SDI) has gained enormous momentum during the last decade. Service-oriented architectures (SOA) as the conceptual basis for SDI provide the means to exchange data and functionality across systems borders, allowing a better integration of geoinformation technology into different domains (Friis-Christensen et al. 2007). However, existing SDIs are rather data centric, thus offering schemas and services that mostly aim to facilitate geodata access and sharing. Processing-centric infrastructures would have to support interaction patterns for sharing and accessing geoprocessing functionality. Necessary components, such as geoprocessing services, are still subject to research (Craglia et al. 2008).

As most complex geoprocessing algorithms are tightly coupled to a concrete geoprocessing platform and runtime environment, a number of challenges arise: Offering geoprocessing functionality in a Web Service environment requires costly adaptations or even reimplementations. It is also difficult to move geoprocessing algorithms among different processing service instances. Currently, whenever spatial data is processed in an SDI, the data is rather moved to the processing instance than vice versa. Considering scenarios where large amounts of data need to be processed, it might be advantageous not to transfer the data to the processing instance but to move the algorithmic code to a processing instance that resides closer to the data, thus saving a substantial amount of bandwidth and data transportation time.

This "moving code" paradigm is expected to outrank existing solutions focusing on the orchestration of Web Services in terms of execution performance (Brauner et al. 2009, Friis-Christensen et al. 2007) and lifecycle management, but has not yet been explored in depth. This article presents a service architecture and proposes related interaction patterns that enable the application and assessment of the "moving code" paradigm in existing SDIs.

The reminder of the article discusses state-of-the-art to realize interoperable geoprocessing (next section) followed by the presentation of an architecture to enable moving code for geoprocessing algorithms in a Web-based SDI. The presentation of a prototypical implementation using state-of-the-art geospatial Web Services and a final discussion complete the article.

2 Sharing Geoprocessing Algorithms in Spatial Data Infrastructures – State-of-the-Art

Considering conceptual approaches to deal with geoprocessing in a SOA based SDI, two basic strategies can be discriminated:

1. In data-driven approaches, geodata is sequentially shipped between a number of service instances, each offering a certain set of geoprocessing operators taking geodata as an input and creating an intermediate result. Consequently, the sequence of geoprocessing operators has to be organized in such a way that the last invoked geoprocessing service provides the desired analysis results. Service chaining techniques, such as Web Service orchestration or choreography, are applied to compose static service based operators and thus realize complex geoprocessing algorithms.

		Data Coupling	
		tightly coupled	loosely coupled
Execution Scheme	instant execution	2a)	2b)
	permanent deployment	2c)	2d)

Figure 1 Discrimination of "moving code" approaches

2. In code-driven approaches, a coded algorithm is shipped from a client to a specific service instance ("moving code"). This instance is supposed to interpret and run the coded algorithm and to offer its execution results through a Web Service interface. For "moving code" it is assumed that a specific algebra or higher level geoprocessing language is available. This language provides a well-defined set of atomic geoprocessing operators allowing the coding of the desired algorithm.

"Moving code" solutions can be further divided by their relation to existing data sets and the execution scheme (Figure 1):

2a. The sent algorithm is tightly coupled to some data sets and instantly executed by an invoked service. The required input data has to be shipped with the code or must be known to the invoked service.
2b. The sent algorithm is executed instantly and independently from the location of data. The required input data is retrieved through standardized service interfaces.
2c. The sent algorithm is tightly coupled to some data sets and deployed on a service prior to execution. The required input data has to be shipped with the code or must be known to that service.
2d. The sent algorithm is deployed on a service prior to execution, independently from the location of data. The algorithm is repeatedly executable via a service interface that allows the provision of input data.

Each of these strategies requires different workflows to produce a certain result (Figure 2). Execution performance and flexibility are determined by the data transportation time (t_T) and the time required by the individual geoprocessing operators (t_{OP}) or Web Service invocations (t_S). If the result is to be immediately consumed by a client, the final delivery time ($t_{T\ deliv}$) adds to the execution's duration.

Nowadays SDI implementations mostly focus on a data-driven approach by using OpenGIS Web Services (OWS) to enable spatial data discovery, access and portrayal. Geoprocessing capabilities can be offered using the very generic Web Processing Service Interface (WPS, OGC 2007b). An in-depth analysis of its potential in distributed processing is offered by Kiehle et al. (2007) and Friis-Christensen et al. (2007). A WPS instance offers a set of processes which are controlled by assigning valid input and output parameters. While the syntax for specifying data formats and IO parameters is machine readable, the semantics – what the process actually does with these data – has to be provided in a human-readable way or by extending the interface with semantic annotations (Zaharia et al. 2008). Existing implementations of this standard either offer atomic operators, originating from specific IS libraries like Sextante (52°North 2010) and

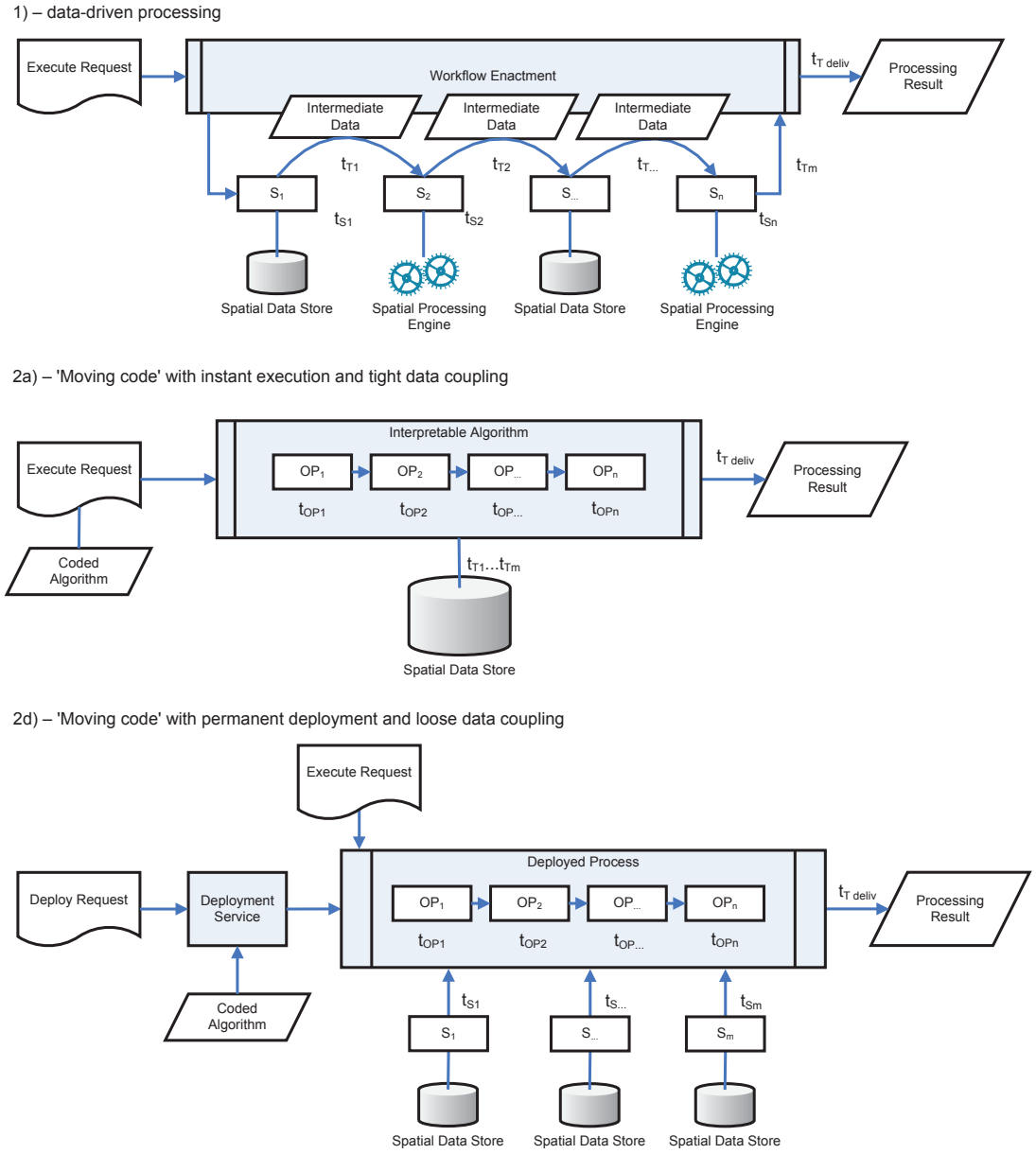


Figure 2 Example workflows of service based geoprocessing

GRASS (PyWPS 2010), or make use of a WPS framework to provide a dedicated and purpose specific set of complex algorithms (e.g. SANY 2009).

2.1 Workflow Enactment for Geoprocessing Services

Friis-Christensen et al. (2007) show an SDI approach that facilitates a rapid development of spatial risk-management applications. They also examine existing OGC implementation specifications and compare transparent, opaque and translucent chaining approaches according to ISO (2005). The presented architecture contains one single statistics service that implements the OGC WPS specification and offers a single process

to calculate areal statistics in a pan-European forest fire use case. The authors discuss different design approaches to facilitate the development process for new or changing user requirements to increase the flexibility of their approach. Complex processes are split into elementary (atomic) operations. Service chaining techniques are applied to flexibly reassemble the atomic operations. Transparent chaining, although being the most flexible chaining scheme, is considered to have a poor overall performance due to frequent client server interactions and repeated exchange of data. Consequently, it is suggested to apply translucent chaining, allowing the user to define a chain of operations on a processing service instance and pass this chain within a single execute request to the service instance. A conceptual and syntactical adoption of the WPS specification is suggested to allow for translucent chaining.

In a conceptually similar way, Kiehle et al. (2007) present a Web Service based architecture to implement complex geoprocessing models and workflows using Web Service orchestration (WSO). The required geoprocessing functionality is offered by a single processing service instance in terms of elementary map algebra operators. Complex functionality is offered by workflow enactment services, invoking the processing service multiple times. The proposed architecture enhances the OWS stack with a Web Service orchestration framework; composite processes are specified by languages for business process management like BPEL (Business Process Execution Language). A typical geoprocessing workflow invokes geodata services, geoprocessing services and geodata portrayal services to generate and present a desired piece of information. Once created, a geoprocessing workflow is persisted in a process repository and executed through an orchestration engine. As the presented approach is similar to a transparent chaining approach, it scales poorly with the number of processing service invocations. To reduce the cumulated data transportation time, it is suggested to cache geodata locally at the processing service instance (Scholten et al. 2006).

Schaeffer (2008) conceptualizes a transactional Web Processing Service (WPS-T) as a workflow enactment service. Adapting the WPS specification, operations for a workflow deployment are introduced. The approach is generic and enables the deployment of executable code on the service. A WPS *ProcessDescription* document serves as a rigorous interface description for the deployed code. The WPS-T falls into Category 2c) or 2d) of the aforementioned classification (see Figure 1). A proof of concept implementation is based on BPEL and shows similar benefits and shortcomings as the approach presented by Kiehle et al. (2007).

The Web Coverage Processing Service (WCPS) Language Interface standard (OGC 2009) defines a set of algebraic operations for raster data processing to be offered by a Web Service. As this language is standardized, it provides the means to exchange geoprocessing algorithms across platforms and thus falls into the category of algorithm-driven approaches. The standard is adopted by the Web Coverage Service (WCS, OGC 2008) specification and allows the processing of locally stored spatial data sets on a WCS instance. Due to the tight coupling of processing engine and data, no additional data transfer over the network is required which is beneficial for execution performance. Similar to the thematic and spatial filtering (Filter Encoding Specification, OGC 2005) that is applicable to select a desired vector data set offered via a Web Feature Service (WFS, OGC 2006), WCPS realizes a "moving code" approach with instant execution and tight data coupling on a WCS (Category 2a) in the aforementioned schema (see Figure 1).

2.2 Grid Approaches to Increase Performance

Grid computing has been proposed by several authors to handle computation intensive geoprocessing tasks or large amounts of data in distributed service-based SDIs (Baranski 2008, Hobona et al. 2007, A. Padberg and C. Kiehle, unpublished). According to Foster et al. (2001) a Grid environment provides a means to securely share storage and computation resources among the participating nodes. Grid infrastructures and SDIs are different and partly incompatible concepts (Hobona et al. 2007). For Grid-enabled geoprocessing services, the common architectural approach is to set up a geoprocessing service instance as a facade to the Grid infrastructure and pass the processing task either directly to a Grid middleware or a third-party service providing access to that middleware. Execution and computation are conducted by the Grid middleware that distributes the task over the Grid's processing nodes. When the execution is finished, the result is returned back to the user through the facade. By sharing algorithmic code and data internally among the individual processing nodes, Grid computation qualifies as a "moving code" approach. Here, code and data distribution are limited to the extent of the Grid infrastructure.

A general partitioning scheme for geospatial data and the development of parallelized algorithms to gain a maximum advantage from the Grid infrastructure is still an issue (Baranski 2008). Another drawback of current implementations is the necessity to ship large data sets to each processing node at runtime. Finally, considering the invocation of existing geoprocessing systems, a "Gridification" is difficult to achieve due to possible incompatibilities between the Grid middleware and the specific geoprocessing system that needs to run on each node. Hence, this article focuses on encoding and deploying the algorithm in an interoperable manner and less on running the deployment services in a Grid or other parallelized computing infrastructure.

3 An Architecture for Ad-Hoc Algorithm Deployment

The proposed architecture adopts different functional domains from the ORCHESTRA Service Network (OGC 2007a), following a best practice example for an open, service-oriented software architecture for multi-risk management. The ORCHESTRA functional domains are defined as follows (OGC 2007a):

- The user domain comprises the client-side applications and user interfaces.
- The mediation and processing domain provides services and functionality to mediate the service calls from the user domain to appropriate services from the integration domain (e.g. workflow enactment services).
- The integration domain represents single Web Service instances wrapping the functionality provided by source system domain (e.g. a WPS encapsulating a GIS backend).
- The source system domain encompasses legacy backend functionality (e.g. GRASS or ArcGIS) and data sources (e.g. a geo-database).

To preserve flexibility, a general scenario for "moving code" with permanent deployment (DMC) will be considered, including the possible use of tightly and loosely coupled data as well as hybrid cases (Figure 3).

The three possible approaches to integrate geoprocessing functionality in an ORCHESTRA service network are captured in Figure 4. The source system domain

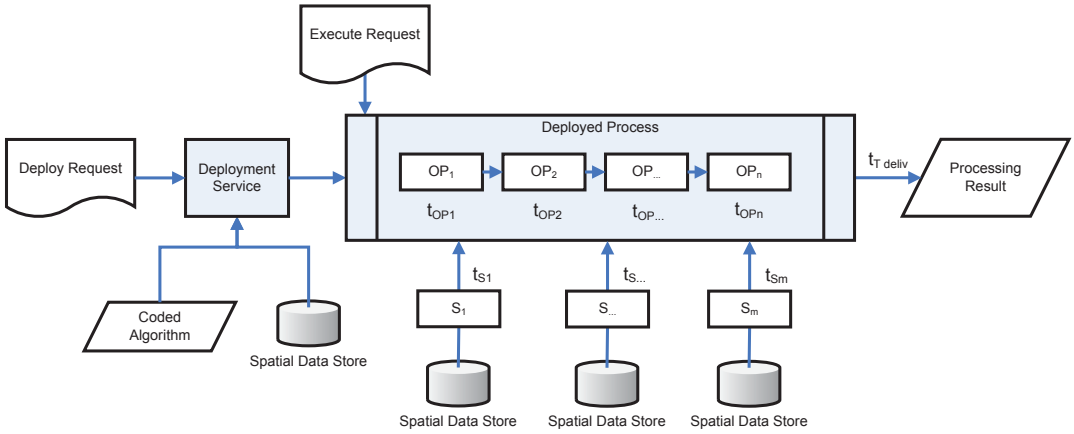


Figure 3 A hybrid DMC scenario

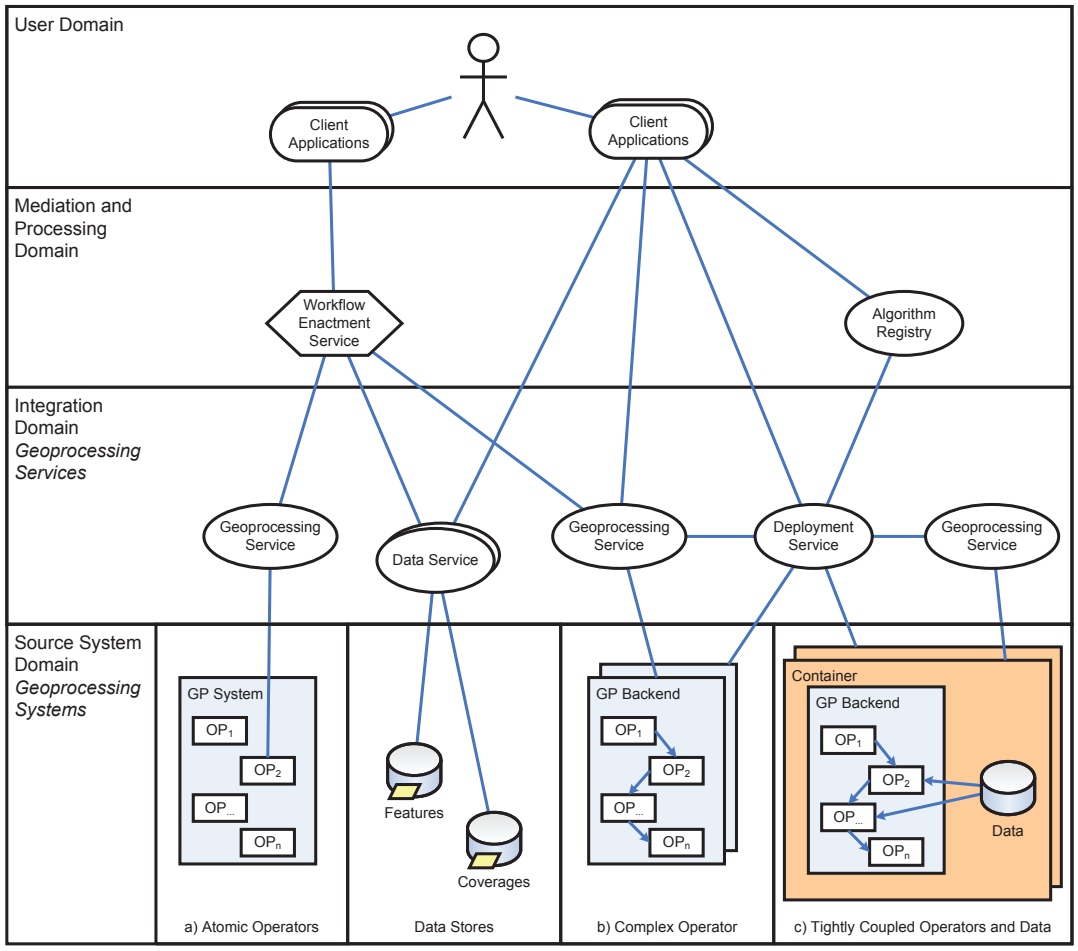


Figure 4 An abstract architecture for DMC according to the functional domains of the ORCHESTRA Service Network (OGC 2007a)

holds different geoprocessing systems. Geoprocessing operators offered there can be of three different types: Atomic operators (Figure 4a) are the smallest accessible geoprocessing functionality in a geoprocessing system, such as *intersect* or *shrink*. Complex operators (Figure 4b) are coarse-grained and defined through the combined functionality of the participating atomic operators. Additionally, in Figure 4c a container is introduced to allow the use of tightly coupled data.

Geoprocessing services offer either atomic or complex operators and provide direct access to geoprocessing systems in the source system domain. If a geoprocessing service is connected with a deployment service through an opaque link, it also allows the deployment of custom algorithms and their subsequent execution through the underlying geoprocessing system.

The act of workflow creation and execution (ISO 2005) is supported from within the mediation and processing domain. Here, services from the lower domains can be registered, searched and chained to workflows. These workflows can be made persistent and executed on a workflow enactment service. In order to support DMC, an algorithm registry is introduced in this domain to store and exchange predefined geoprocessing algorithms.

The user domain contains client applications that provide user interfaces to access any service in the lower domains directly or to control, to instantiate and to manipulate services in the mediation and processing domain. Infrastructures supporting DMC need also to provide the appropriate editors in the user domain to create and publish complex operators.

A DMC deployment procedure is illustrated in Figure 5. After the initial assembly, a geoprocessing algorithm is uploaded to the algorithm registry to allow Web-based access. Prior to the first execution, the code and data package is sent to a deployment service instance. On reception, the algorithm is analyzed and compiled into executable code that can be interpreted by an available geoprocessing backend at the service instance. If additional tightly coupled data items are included in the package, these can be parsed to a local data store which suits the present geoprocessing backends. Once deployed, the algorithm is made visible on a geoprocessing service in the integration domain as a new process and can be executed from a superseding domain. The new process remains accessible until the algorithm is un-deployed.

The architecture also enables algorithm management and maintenance. Depending on the chosen registry, algorithms can be updated and versioned making any revision

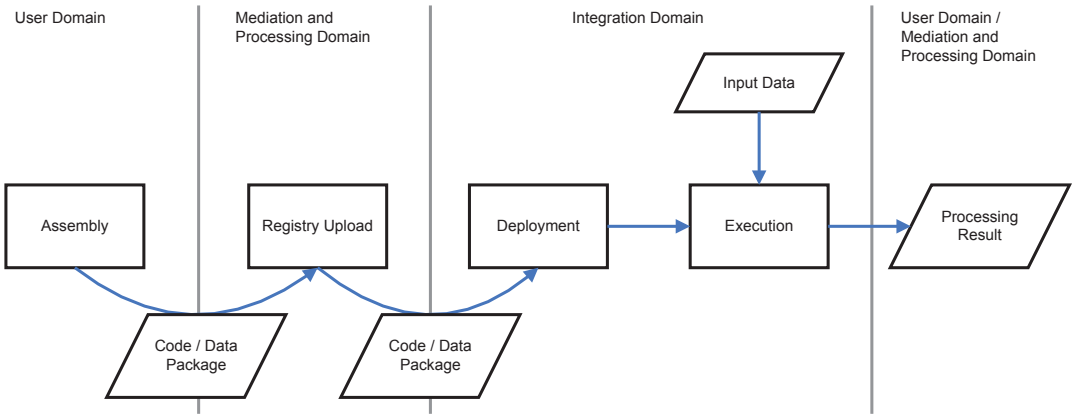


Figure 5 Code deployment and execution in a DMC scenario

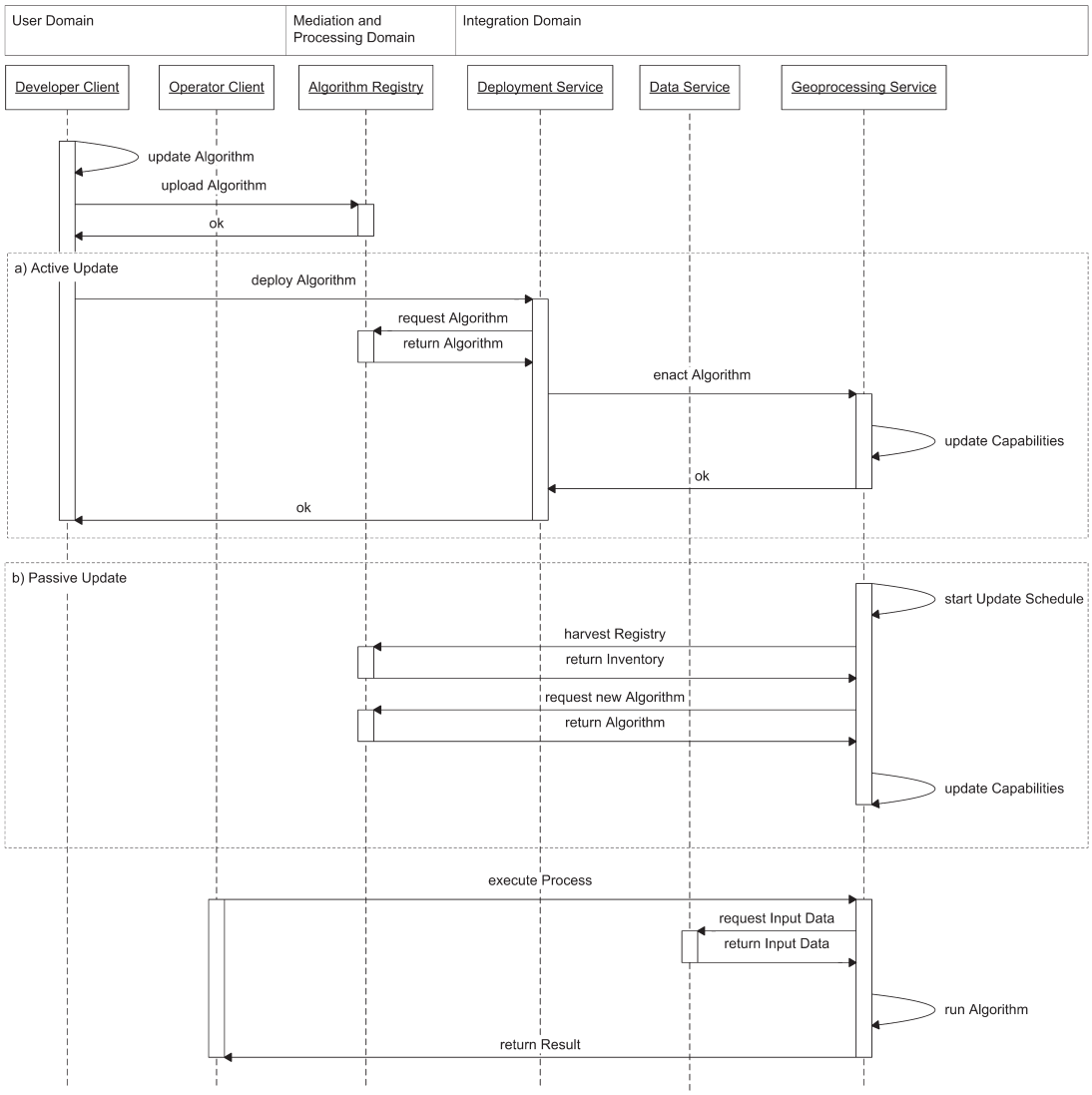


Figure 6 Algorithm update and re-deployment

transparent to other users and also to the deploying geoprocessing services. Depending on a specific setup, it may be required that updates are either forced by a supervising expert (active update) or drawn automatically by the involved geoprocessing services according to a defined schedule (passive update).

An active update (Figure 6a) is conducted by the developer client and similar to a deployment procedure. By issuing a deploy request to a deployment service, the current revision of an algorithm is fetched from the registry and uploaded to the geoprocessing service. Any previous version of that algorithm on the geoprocessing service instance will be overwritten. Such an active update requires an additional security layer at the Web Service level that prevents unauthorized users from writing to the deployment service.

A passive update is either triggered by a scheduler or an event (e.g. an execute request; to make sure that the latest available version is used). If the updates are scheduled (Figure 6b), the geoprocessing service can be synchronized with the whole

registry and update each provided process. In cases where large data sets are shipped with the algorithm, this data does not need to be retrieved at run time.

4 A Prototypical Implementation

The proposed architecture and service interactions have been implemented and tested in the SoKNOS project (Service-Oriented Architectures Supporting Networks of Public Security; SoKNOS 2010). It aims to establish a service platform as a technical basis for collaboration and decision support in emergency management. It is required to operate a Web Service environment that is capable of providing information in (near) real-time, offering failsafe and secure services, providing effortless maintenance and evolution of the decision support tools as well as the means for logging the decision processes for later analysis.

Parts of the presented work are also used in OGC's most recent Open Web Services testbed (OWS-7) to prototype geoprocessing services for feature fusion. These testbeds are part of the OGC Interoperability Program, serving standards development and testing.

4.1 Infrastructure

The hybrid DMC scenario has been prototyped in an SDI using OpenGIS Web Services (Figure 7). This infrastructure offers numerous data access and portrayal services. For service-based geoprocessing it contains several WPS instances and an algorithm registry, allowing a straight-forward realization of hybrid DMC. Algorithms are produced and published by a developer client in a defined GIS environment. This developer client also uploads the combined code and data packages into an algorithm registry.

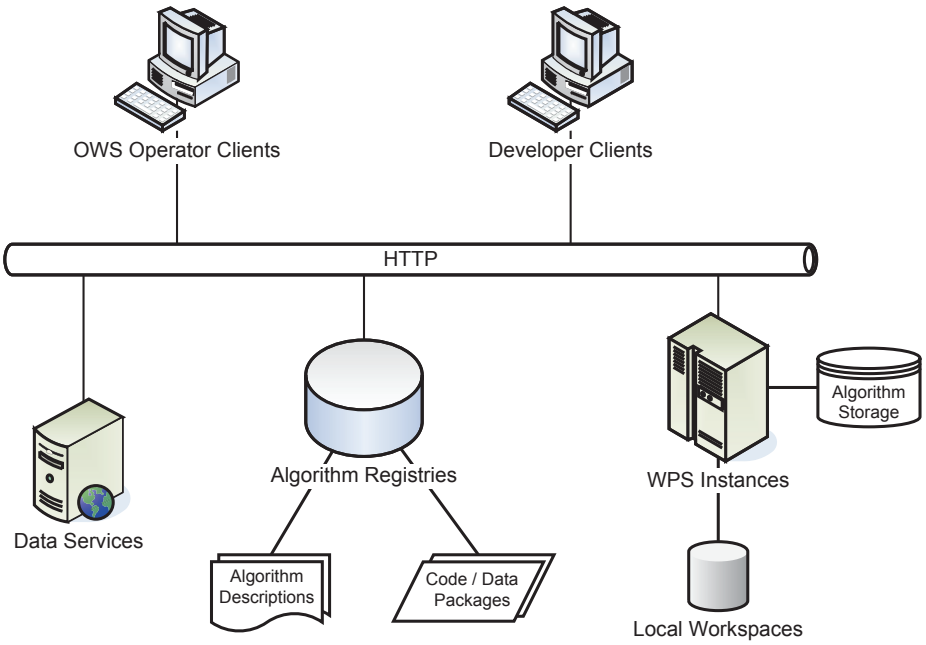


Figure 7 Prototypical implementation architecture

The operator clients in the SoKNOS system can communicate with OpenGIS Web Services in a generic way and have access to both data and processing services to retrieve the desired data and the processing results. Additionally, the WPS instances have their own local process description repositories that can be aligned with the remote registries' contents.

The prototypical algorithm registry is realized by a version management software (SVN – Apache Subversion) with an HTTP interface for public download access. Such a versioned registry allows concurrent development and operational deployment: Each operational WPS may deploy only those algorithms that have been tested thoroughly and are proven to produce reliable and valid results. A versioned algorithm registry also helps to perform forensics of previous decision situations by the possibility to reload the previously used code and data packages.

The prototype was implemented in a coupled WPS/ArcGIS Server environment (Müller et al. 2010). For the server part, the Open Source WPS framework from 52°North has been adapted and extended with connectors to support third party geoprocessing systems and remote algorithm registries. This work was also contributed to the 52°North WPS community that hosts a publicly available demo server (52°North 2010).

4.2 Mapping from Geoprocessing Algorithms to Service Interfaces

Exchanging geoprocessing algorithms between different processing systems requires an established and commonly accepted modeling language or algebra. As today's algorithms for geoprocessing are usually tightly coupled to the characteristics of specific geoprocessing systems (Alarcon et al. 2007), there is no universal modeling algebra that can be used to exchange geoprocessing code.

Lacking a generic, well-defined and standardized way to encode the concrete geoprocessing algorithms, different GIS platform dependent coding languages (here ArcToolbox, Python and GRASS scripts) have been used for the prototypical implementation. Consequently, it is necessary to describe the required runtime environment to such an extent that it is possible to decide whether a piece of geoprocessing code can be executed on a target platform or not. From a GIS perspective, geoprocessing systems like ArcGIS, GRASS, FME or IDRISI have standard APIs that support a set of programming or scripting languages as well as convenient proprietary formats like ArcToolbox or an FME Workbench. From a programming or scripting language perspective, an implemented algorithm can require multiple libraries like GDAL, NumPy or R and multiple geoprocessing systems. Given these assumptions, geoprocessing code can be encoded for a cross platform exchange. Knowing the required runtime environment the receiver can decide whether s/he can deploy and execute that code or not.

Depending on the data types supported by the Web Service, the required processing backend and the chosen API, a mapping is not always possible. As the process interface on a Web Service provides the facade for the desired functionality, the algorithm developer is responsible to ensure an unambiguous mapping between the backend and the Web Service interface.

4.3 Workspace Encoding for Hybrid DMC

In practice, the cross-platform exchange of custom geoprocessing tools is mostly accomplished using the workspace concept. Workspaces may contain a basic file structure that

is required to operate the geoprocessing tool and possibly includes one or more data sets thus paralleling the combined code and data package proposed in Section 3. The compiled algorithm, data items and the required working directory layout are assembled into a file structure that can be used across platforms if a suitable runtime environment is provided. In ISO 19118 (ISO 2006) a syntactical pattern for data interchange is described. This pattern can be adapted to interchange workspaces and executable code among two systems by creating an appropriate interchange schema for geoprocessing workspaces. The description of the required processing platform can be accomplished by using unique identifiers for the required software components. A universal means to communicate the structure of a workspace is the use of Uniform Resource Locators (URL; IETF 2005).

The interchange schema for hybrid DMC is defined in accordance with the WPS interface specification. It uses a common WPS *ProcessDescription* document to describe the desired process interface and is augmented with a referenced XSLT document. Thus, the *ProcessDescription* document can be interpreted in two different ways: As a plain *ProcessDescription* document providing a valid WPS process interface description, or as an *AlgorithmDescription* containing necessary information for successful algorithm deployment and parameter mapping (Figure 8).

A WPS instance pulling its code from a central registry is shown in Figure 9. The process description is loaded from a local or remote repository and parsed for an XSLT reference. After applying a style sheet transformation, an algorithm description is

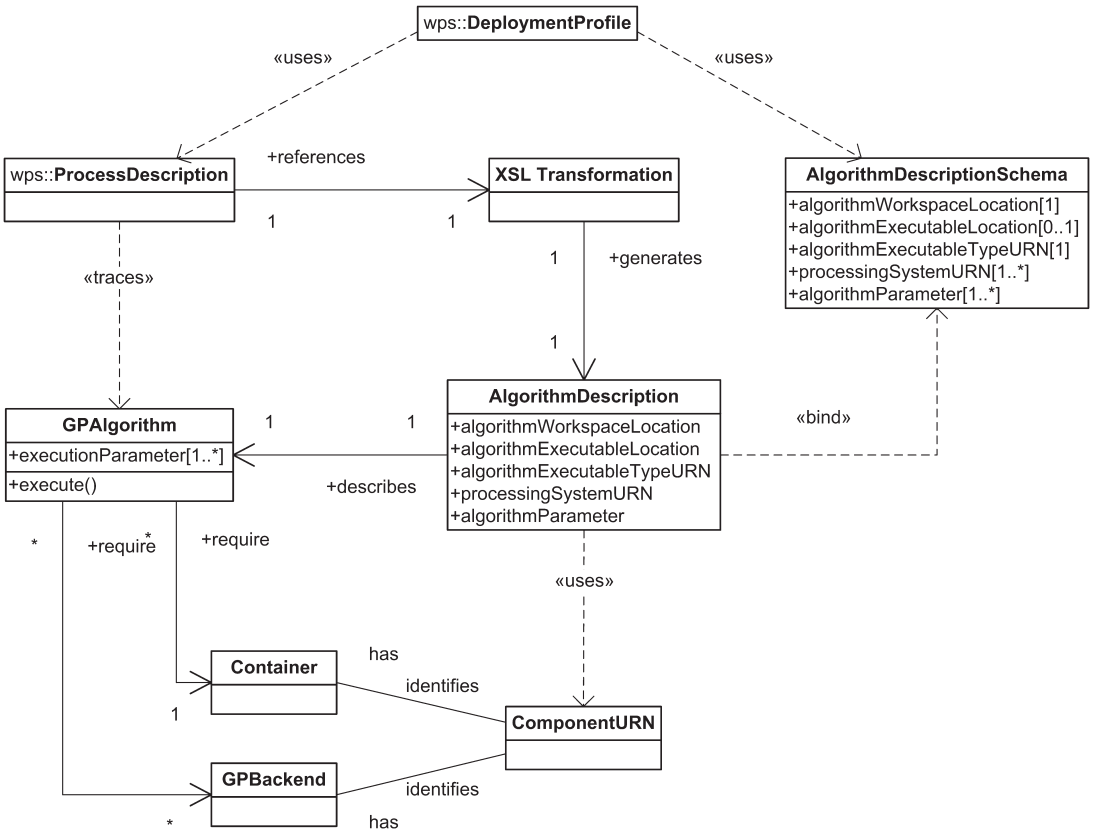


Figure 8 Deployment classes for platform specific geoprocessing algorithms

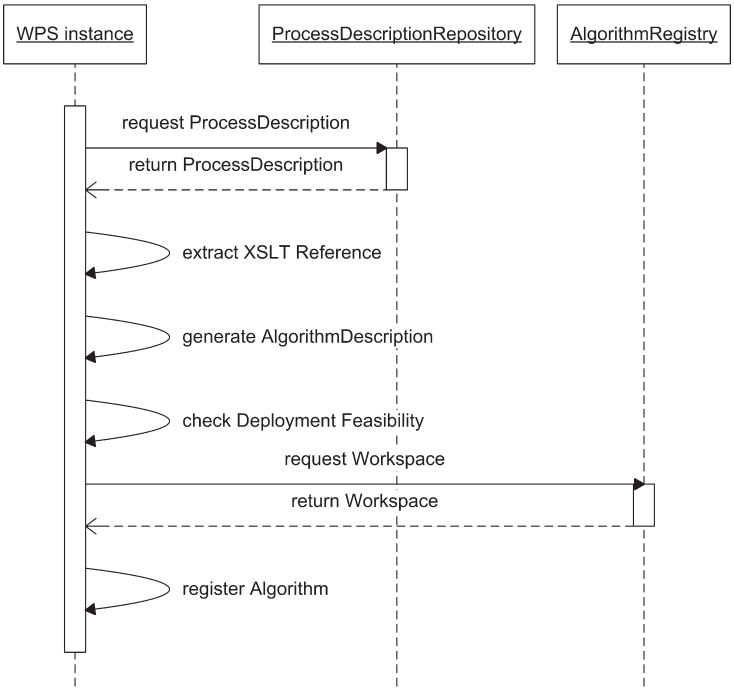


Figure 9 Sequence diagram for a pull-deployment

obtained and evaluated for deployment feasibility. The intended runtime environment is captured with Component URNs that indicate the required geoprocessing libraries or systems as well as the format of the executable algorithm.

If the WPS instance can provide an appropriate runtime environment for the described algorithm, the workspace is loaded and stored locally for execution. With the concept of a transactional WPS, the algorithm can also be pushed to the service assuming a proper advertisement of the supported runtime environment in the service capabilities. In this case, the sequence is not initiated by pulling a description from a registry, but a *deployProcess* operation that passes the augmented *ProcessDescription* document directly to the desired WPS instance.

4.4 Use Case

Two service-based decision support tools have been developed for the SoKNOS system:

1. An assessment algorithm that helps decision makers in assessing the threat imposed by escaping gaseous contaminants on the local population.
2. A delineation algorithm that roughly extracts flooded areas from flooding simulations and was used for visual analysis and resource planning.

The assessment algorithm was realized as a Python script with a reduced set of input parameters:

- a coverage from a simulation service, indicating the expected immission in the endangered area,
- a threshold value, indicating the hazardous contaminant concentration,
- a threshold value, indicating a possibly lethal contaminant concentration.

High resolution data about population distribution was stored with the Python code in an appropriate workspace to save bandwidth and keep the processing interface simple. On execution, the users obtain an output that was generated by a multi-criteria evaluation model, aggregating the population distribution and the variable immission levels provided at runtime.

The delineation algorithm was developed with the ArcGIS Model Builder and stored in ArcToolbox. It contains a chain of morphological, classification and generalization operators, performing the creation of flooding polygons and the merging or elimination of areas below a pre-defined size. To provide a simple interface one single synthetic parameter is used to control the vectorization process. As no constant geodata sets are required for the algorithm, it was deployed as a workspace that only contains the specific ArcToolbox.

During Web Service operation up-to-date workspace content was ensured by a scheduled synchronization at the participating WPS instances with the algorithm registry. If the use of the most recent version had to be guaranteed, it was optionally possible to check prior to each execution, whether an updated workspace and thus an updated algorithm is available at the registry.

4.5 Findings in Lifecycle Management and Operation

The architecture supports rapid prototyping in a distributed development process: Algorithm assembly in a Desktop GIS offers early results for discussion with the targeted users. Gradual adjustments to the algorithm as well as a recreation from scratch can be accomplished in the source system domain. Once a consolidated tool is created, the local workspace can be uploaded to the algorithm registry and is instantly available as a Web Service.

Having an early prototype is common practice in software development and a prerequisite to evolve the functionality in an iterative, yet timely manner (Figure 10). For both impact assessment and flooding area delineation, the applied methods had to be reworked and evolved to better suit the end-users' needs. Applying DMC, algorithmic changes become a matter of code replacement without touching the upper domains in the architecture (Section 3). To further decrease processing time, it is also possible to switch the processing backend later on or add more constant, thus cacheable, data sets to the workspace.

As Emergency Management requires failsafe and close to real-time data processing, redundant infrastructures are frequently found in this domain. Assuming a redundant WPS setup with the required algorithm available, Weiser and Zipf (2007) use a BPEL process for "secure Web Service Allocation" that retrieves a set of available WPS from a catalogue and checks a candidate service instance prior to execution for availability. With the ability to distribute algorithms among a set of processing services, it becomes possible to quickly deploy a redundant algorithm setup and thus ensure a desired degree of redundancy.

With the client's ability to select the WPS instance for deployment, it is possible to choose the service that is expected to provide the best performance. Data transportation time and available processing power are the primary determinants for the overall execution performance. Thus the considered WPS instance should be close to the required data sources in terms of available bandwidth and network topology and simultaneously offer

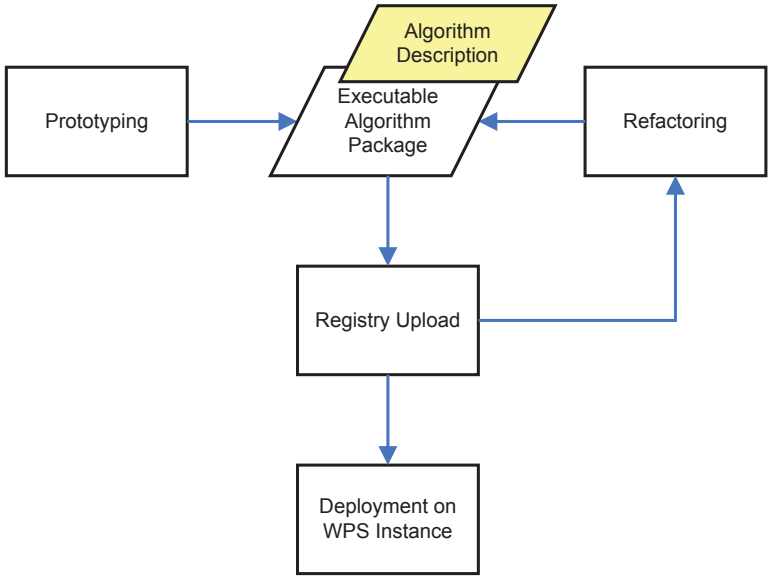


Figure 10 Lifecycle management

sufficient processing power to quickly compute the result. Dedicated performance tests are subject to future investigations.

5 Conclusions and Outlook

This article has demonstrated the feasibility and the benefits of "moving code" approaches in SDI. The applied paradigm was proven to improve performance in Web Service based geoprocessing tasks. The presented and implemented DMC approach also helps to reduce development efforts for new geoprocessing functionality, facilitates lifecycle management of geoprocessing algorithms and reduces the maintenance costs of geoprocessing services.

The prototypical implementation shows that there is almost no performance loss compared to a workstation GIS as data transportation and communication overheads are minimized. A pure service chaining approach has to retrieve all the involved data sets from remote resources at runtime, making it hard to reach close to real-time computation results. Using hybrid DMC with adequate deployment strategies to transmit a workspace, the involved constant data items are retrieved only once and most probably not at runtime.

For the SoKNOS use cases, "moving code" approaches with permanent deployment were proven useful in lifecycle management. During the development phase, new algorithms were created within a day. Assessing the deployment procedure, the refinement of WPS process interfaces was identified as the most expensive task in the development process. Once a stable interface was agreed upon, later updates of the algorithms were completely transparent to the WPS clients. An obvious means to create stable interfaces for common geoprocessing tasks is the development and implementation of WPS profiles

OGC (2007b): Complementary to an *AlgorithmDescription* stating the required runtime environment, WPS Profiles can be used to describe system-independent functionality and processing interfaces.

For general "moving code" scenarios it was found that all performance considerations clearly depend on the given modus operandi. In the case of one-time assembly and execution when no caching is possible, the performance gain compared to a data-driven approach will diminish. Thus, the SDI setup and its designated use cases favor either a data-driven or a "moving code" approach.

"Moving code" approaches are considered beneficial if:

- Algorithms are frequently changed and evolved
- Identical algorithms have to be deployed at or shared among several service instances
- A substantial amount of data can be shipped with the algorithm and stored prior to execution
- Some tightly coupled data sets can be used to increase performance (caching impact)
- Algorithms have to be placed at a processing service that resides "close" to the data (bandwidth impact)

Data-driven approaches have been proven beneficial in the following scenarios:

- One-time assembly and execution of workflows
- Real-time response for complex service chains is not required
- The required atomic operators are available at operational processing services
- The required simple operators are available at the data service level

The approach presented here lacks a rigid and well defined formalization mechanism to allow an interoperable encoding and deployment of the transported geoprocessing algorithm. Progressing in the design of such well defined geoprocessing algebras is one of the future research challenges (Brauner et al. 2009). Valuable input and starting points to this research are: Map Algebra (Tomlin 1990), Multidimensional Map Algebra (Mennis 2010), OpenGIS Filter Encoding (OGC 2005), WCPS Language (OGC 2009), the Spatial Extension to SQL (Egenhofer 1994), existing Geoprocessing libraries and scripting languages as well as approaches towards model sharing in Spatial Decision Support Systems (El-Gayar and Tandekar 2007, Power and Ramesh 2007).

The described approaches combined with the potentialities of a Grid infrastructure mark a further challenge for future investigations. The Grid middleware and the underlying computing resources can be considered as an additional source system that has to be integrated. A processing service interface and an appropriate deployment service provide access to the Grid environment. Alternatively, it is also possible to create a service based Grid infrastructure, where each node holds a processing service. "Moving code" could be deployed on these individual services thus forming a large service array for parallel processing.

Acknowledgements

Parts of the research presented here has been conducted in the frame of the project SoKNOS (project number 01ISO7009) funded by the German Federal Ministry of Education and Research to foster the development of service oriented architectures for

risk management. Additional thanks go to the anonymous reviewers who helped to improve this paper. We would also like to thank Stephan Mäs for helping to prepare the final article.

References

52°North 2010 The 52°North WPS Framework. WWW document, http://52north.org/wps

Alarcon V J, O'Hara C G, Viger R, Shrestha B, and Mali P 2007 Using an interoperable geoprocessing system for hydrological simulation. In *Proceedings of the International Conference on Computational Methods in Science and Engineering (ICCMSE 2007)*, Corfu, Greece: 1136–40

Baranski B 2008 Grid computing enabled web processing service. In Pebesma E, Bishr M, and Bartoschek T (eds) *GI-Days 2008: Proceedings of the Sixth Geographic Information Days*. Münster, Institut für Geoinformatik: IfGIprints No 32: 243–56

Brauner J, Foerster T, Schaeffer B, and Baranski B 2009 Towards a research agenda for geoprocessing services. In *Proceedings of the Twelfth Annual AGILE Conference*, Hanover, Germany

Craglia M, Goodchild M F, Annoni A, Camara G, Gould M, Kuhn W, Mark D, Masser I, Maguire D, Liang S, and Parsons E 2008 Next-generation Digital Earth: A position paper from the Vespucci Initiative for the Advancement of Geographic Information Science. *International Journal of Spatial Data Infrastructures Research* 3: 146–67

Egenhofer M 1994 Spatial SQL: A query and presentation language. *IEEE Transactions on Knowledge and Data Engineering* 6: 86–95

El-Gayar O and Tandekar K 2007 An XML-based schema definition for model sharing and re-use in a distributed environment. *Decision Support Systems* 43: 791–808

Foster I, Kesselman C, and Tuecke S 2001 The anatomy of the Grid: Enabling scalable virtual organizations. *International Journal of Supercomputer Applications* 15: 200–22

Friis-Christensen A, Lutz M, Ostländer N, and Bernard L 2007 Designing service architectures for distributed geoprocessing: Challenges and future directions. *Transactions in GIS* 11: 799–818

Hobona G, Faribairn D, and James P 2007 Workflow enactment of Grid-enabled geospatial Web services. In *Proceedings of the Sixth UK e-Science All Hands Meeting*, Nottingham, United Kingdom

IETF 2005 Uniform Resource Identifier (URI): Generic syntax. WWW document, http://tools.ietf.org/html/rfc3986/rfc3986.htm

ISO 2005 ISO 19119:2005 geographic information services. International Standard, ISO TC 211

ISO 2006 ISO 19118:2006 geographic information – encoding. International Standard, ISO TC 211

Kiehle C, Greve K, and Heier C 2007 Requirements for next generation spatial data infrastructures: Standardized Web-based geoprocessing and Web service orchestration. *Transactions in GIS* 11: 819–34

Mennis J 2010 Multidimensional map algebra: Design and implementation of a spatio-temporal GIS processing language. *Transactions in GIS* 14: 1–21

Müller M, Bernard L, and Vogel R 2010 Multi-criteria evaluation for emergency management in Spatial Data Infrastructures. In Zlatanova S, Konecny M, and Bandrova T (eds) *Cartography and Geoinformatics for Disaster Management: Developments and Trends*. Berlin, Springer: 273–86

OGC 2005 *OpenGIS Filter Encoding Implementation Specification 1.1.0*. Wayland, MA, OGC Document No 04-095

OGC 2006 *OpenGIS Web Feature Service (WFS) Implementation Specification, Corrigendum, 1.0.0*. Wayland, MA, OGC Document No 06-027

OGC 2007a *Reference Model for the ORCHESTRA Architecture (RM-OA) V2 (Rev 2.1)*. Wayland, MA, OGC Document No 07-097

OGC 2007b *OpenGIS Web Processing Service, Version 1.0.0*. Wayland, MA, OGC Document No 05-007r7

OGC 2008 *Web Coverage Service Implementation Standard*. Wayland, MA, OGC Document No 07-067r5

OGC 2009 *Web Coverage Processing Service Language Interface Standard*. Wayland, MA, OGC Document No 08-068r2

Power D J and Ramesh S 2007 Model-driven decision support systems: Concepts and research directions. *Decision Support Systems* 43: 1044–61

PyWPS 2010 The Python web processing service. WWW document, http://pywps.wald.intevation.org/

SANY 2009 *SANY: An Open Service Architecture for Sensor Networks*. Vienna, SANY Consortium

Schaeffer B 2008 Towards a transactional WPS (WPS-T). In Pebesma E, Bishr M, and Bartoschek T (eds) *GI-Days 2008: Proceedings of the Sixth Geographic Information Days*. Münster, Institut für Geoinformatik: IfGIprints No 32: 91–116

Scholten M, Klamma R, and Kiehle C 2006 Performance evaluation of spatial data infrastructures for geoprocessing. *IEEE Internet Computing* 10(5): 34–41

SoKNOS 2010 SoKNOS: Service-orientierte ArchiteKturen zur Unterstützung von Netzwerken im Rahmen Oeffentlicher Sicherheit (Service-Oriented ArchiteCtures Supporting Networks of Public Security). WWW document, http://www.soknos.de/

Tomlin D 1990 *Geographic Information Systems and Cartographic Modelling*. Englewood Cliffs, NJ, Prentice-Hall

Weiser A and Zipf A 2007 Web service orchestration of OGC web services for disaster management. In Li J, Zlatanova S, and Fabbri A (eds) *Geomatics Solutions for Disaster Management*. Berlin, Springer Lecture Notes in Geoinformation and Cartography: 239–54

Zaharia R, Vasiliu L, Hoffman J, and Klien E 2008 Semantic execution meets geospatial Web services: A pilot application. *Transactions in GIS* 12: 59–73

3. Moving Code – Sharing Geoprocessing Logic on the Web

Müller, Matthias; Bernard, Lars; Kadner, Daniel: Moving code – Sharing geoprocessing logic on the Web. In: ISPRS Journal of Photogrammetry and Remote Sensing, 83, 2013, pp. 193–203.

Abstract. Efficient data processing is a long-standing challenge in remote sensing. Effective and efficient algorithms are required for product generation in ground processing systems, event-based or on-demand analysis, environmental monitoring, and data mining. Furthermore, the increasing number of survey missions and the exponentially growing data volume in recent years have created demand for better software reuse as well as an efficient use of scalable processing infrastructures. Solutions that address both demands simultaneously have begun to slowly appear, but they seldom consider the possibility to coordinate development and maintenance efforts across different institutions, community projects, and software vendors.

This paper presents a new approach to share, reuse, and possibly standardise geoprocessing logic in the field of remote sensing. Drawing from the principles of service-oriented design and distributed processing, this paper introduces moving-code packages as self-describing software components that contain algorithmic code and machine-readable descriptions of the provided functionality, platform, and infrastructure, as well as basic information about exploitation rights. Furthermore, the paper presents a lean publishing mechanism by which to distribute these packages on the Web and to integrate them in different processing environments ranging from monolithic workstations to elastic computational environments or "clouds". The paper concludes with an outlook toward community repositories for reusable geoprocessing logic and their possible impact on data-driven science in general.

ISPRS Journal of Photogrammetry and Remote Sensing 83 (2013) 193–203

Contents lists available at SciVerse ScienceDirect

ISPRS Journal of Photogrammetry and Remote Sensing

journal homepage: www.elsevier.com/locate/isprsjprs

Moving code – Sharing geoprocessing logic on the Web

Matthias Müller*, Lars Bernard, Daniel Kadner

Technische Universität Dresden, Geoinformation Systems, 01062 Dresden, Germany

ARTICLE INFO

Article history:
Available online 29 March 2013

Keywords:
Geoprocessing
Software reuse
Interoperability
Spatial data Infrastructures
Standards

ABSTRACT

Efficient data processing is a long-standing challenge in remote sensing. Effective and efficient algorithms are required for product generation in ground processing systems, event-based or on-demand analysis, environmental monitoring, and data mining. Furthermore, the increasing number of survey missions and the exponentially growing data volume in recent years have created demand for better software reuse as well as an efficient use of scalable processing infrastructures. Solutions that address both demands simultaneously have begun to slowly appear, but they seldom consider the possibility to coordinate development and maintenance efforts across different institutions, community projects, and software vendors.

This paper presents a new approach to share, reuse, and possibly standardise geoprocessing logic in the field of remote sensing. Drawing from the principles of service-oriented design and distributed processing, this paper introduces moving-code packages as self-describing software components that contain algorithmic code and machine-readable descriptions of the provided functionality, platform, and infrastructure, as well as basic information about exploitation rights. Furthermore, the paper presents a lean publishing mechanism by which to distribute these packages on the Web and to integrate them in different processing environments ranging from monolithic workstations to elastic computational environments or "clouds". The paper concludes with an outlook toward community repositories for reusable geoprocessing logic and their possible impact on data-driven science in general.

1. Introduction

When the use of computers in remote sensing gained momentum in the late 1960s, an immediate need for computational methods for the timely handling, analysis, and dissemination of data became apparent (Simonett, 1969). Although the technology has evolved significantly since this time period, the effective and efficient processing of data originating from various sensors or data sources is still a topic in recent publications (e.g., Deqiang et al., 2012; Happ et al., 2010; Plaza, 2006).

The evolution of internet and Web technologies beginning in the late 1990s has also heavily influenced the methods of remote-sensing data management and distribution. Today, several Web portals provide access to remote-sensing data catalogues and data stores. These portals allow for the easy discovery and access (i.e., download) of remote-sensing data. Prominent examples are the remote-sensing portals of NASA (NASA, 2012) and ESA (ESA, 2012), as well as the Portal on the European Land Cover Data Corine (Eionet, 2012).

Global and regional initiatives, such as Digital Earth (Craglia et al., 2012), GEOSS (GEO, 2012) and GMES (GMES, 2012), have been established not only to exploit and encourage the development of remote-sensing applications but also to realise and improve global infrastructures, allowing for an easy and efficient exchange and utilisation of remote-sensing data and the derived information products. Additional concepts for infrastructures in eScience or so-called cyberinfrastructures have arisen and are envisioned as "... a combination of data, resources, network protocols, computing platforms, and computing services that brings people, information, and computational tools together to perform science or other data-rich applications in this information-driven world" (Yang et al., 2010). With spatial or geo-referenced data being one of the driving information resources, geospatial cyberinfrastructures are framed around the characteristics of geospatial data (including Earth observation data and sensors), geospatial data processing, geo-visualisation, and spatial decision support. In their conceptual framework, Yang et al. (2010) identify distributed geographic information processing and high-performance computing as key technologies necessary to achieve a geospatial cyberinfrastructure. In parallel, it is suggested to have these capabilities available in a service-oriented manner to facilitate rapid application development. Gray (2009) demands more efficient, easy to use data analysis tools for eScience in general, and Craglia et al. (2012) articulate a similar demand for the future Digital Earth.

* Corresponding author. Tel.: +49 351 463 31953; fax: +49 351 463 35879.
E-mail address: Matthias_Mueller@tu-dresden.de (M. Müller).

0924-2716/$ - see front matter © 2013 International Society for Photogrammetry and Remote Sensing, Inc. (ISPRS) Published by Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.isprsjprs.2013.02.011

Industry and de-jure standards, most prominently from the Open Geospatial Consortium (OGC), the International Organization for Standardisation (ISO), or the Institute of Electrical and Electronics Engineers (IEEE), allow for harmonised data formats and interoperable discovery, viewing, and download services on the Web. Standards such as the interface specifications for Web Coverage Services and Sensor Observation Services are rooted in remote-sensing applications and are widely used today. However, current developments mainly focus on data exchange in distributed environments and only a few authors have considered the possibility to exchange geoprocessing logic in Web-based environments (Friis-Christensen et al., 2007; Müller et al., 2010; Kiehle et al., 2007). There are three main facets associated with these developments. First, there is a focus on reusing computational algorithms not only to avoid duplicate implementations but also to allow peer reviews of algorithms and, in the long term, gain efficiency and better quality algorithms. Second, as the quantity and size of remote-sensing datasets challenge computing power and internet bandwidths even with today's technology, mechanisms are required that consider moving the computational code closer to the data rather than moving excessive amounts of data to a processing service. Third, elastic computing environments, such as grid and cloud computing, call for techniques to enable the easy distribution of geoprocessing logic in scalable environments. These developments face two main issues: first, there is a lack of a well-established common understanding of the functional granularity necessary to support recurring geo-computational tasks, and, second, there is a lack of mechanisms that support the exchange of processing functionality across different distributed computational platforms (Gray, 2009; Yang and Liu, 2012).

This paper proposes a lightweight framework that contracts geoprocessing logic in the following four dimensions. (1) A functional description provides a formalisation of the input and output data with respect to structure and content and a comprehensible description of the procedure used to derive the outputs from the inputs. (2) A platform description states the dependencies on other software that needs to be available for deploying and executing the computing logic. Such information enables the determination of a proper software platform for deployment or probing the compatibility of existing configurations. (3) An infrastructure description states the hardware requirements needed to execute the algorithmic code reliably. (4) The exploitation rights associated with the computing logic constitute the fourth dimension.

The paper proceeds with a review of existing approaches to the processing of large data and their application in the remote-sensing domain, with a special focus on code reuse and code sharing approaches to better deal with big data. The next section presents a novel approach for packaging and sharing reusable code based on the requirements of service-oriented software design and cloud computing service models. To prove the feasibility of the approach, the concept is then applied to a data-intensive anomaly detection workflow in a distributed geodata infrastructure and is realised using open standards wherever possible. The paper concludes with an outlook on community repositories for reusable algorithmic code and some thoughts regarding a code-sharing standard.

2. Approaches to big data processing

Data processing challenges in remote sensing are manifold. Datasets are becoming larger with each generation of sensors due to their increased radiometric, spectral, and geometric resolution. At the same time, the number of datasets is growing rapidly with each new space or airborne observation platform (Pallickara et al., 2011). These problems are not unique to remote sensing but are characteristic of all data-driven research and analysis activities (Clery and Voss, 2005; Gray, 2009; Yang et al., 2010). The problem is approached in two different ways. Data partitioning and parallelisation efforts attempt to take advantage of multi-core systems and computer clusters by splitting computations across a large number of processors. Additionally, software reuse efforts are an approach to deal with the large variety of observation data and analysis tasks by providing pluggable, reusable building blocks as a foundation for new processing workflows. When grid and cloud computing infrastructures began to appear, it became clear that the concept of pluggable, loosely coupled atomic processes could also be used to accomplish parallel processing workflows in these new scalable computing environments.

The following sections summarise existing approaches to address scalability issues and efforts to increase software reuse in the remote-sensing domain.

2.1. Parallel data processing

Partitioning techniques are the most obvious approach to mass data handling. They are essential to process very large datasets that would not fit into a computer's memory. Incremental algorithms are designed to be used with sequential data subsets or continuous data streams, and there is a large body of knowledge in this domain of computer science (e.g., Knuth, 1997). These algorithms are frequently used for computing basic statistical parameters of large satellite images, time series data, or large multipoint samplings.

Most parallelisation techniques to reduce computational costs rely on partitioned datasets. Numerous publications have presented innovative approaches for the parallel processing of remote-sensing data. A general discussion on data partitioning approaches and parallel processing techniques for hyperspectral images is presented by Plaza (2006). Happ et al. (2010) and Korting et al. (2011) focus on the shortcomings of existing image segmentation algorithms. These authors use partitioning approaches along with a parallel re-implementation of established algorithms to reduce the computational costs on a multi-core system. A software framework for parallel image classification, including pre- and post-processing steps, is presented by Bernabé et al. (2012). Teutsch et al. (2011) present a partitioning and parallelisation approach for clustering point cloud data that may originate from laser scanning devices. Some of these processing tasks can be considered "embarrassingly parallel" and thus require very little data partitioning and re-assembly effort (Raicu et al, 2008). However, as partitions and parallel threads are increasing in number, saturation effects may be observed (Muñoz-Marí et al., 2009) due to communication and coordination efforts. In some cases, the peak performance is limited by the underlying storage system because the data cannot be transferred quickly enough to the computer's memory.

For completeness, it should also be noted that there are additional approaches that take advantage of dedicated hardware. An interesting idea is the use of graphics hardware and specialised algorithms for computationally intensive processing tasks (Xiaoshu and Hong, 2010; Christophe et al., 2011). Instead of explicit parallel programming, the graphics processors internally execute single instructions on multiple data (SIMD). This hardware architecture is ideal for processing array-like data structures in cases where identical operations must be performed on each of the elements.

2.2. Software reuse in data processing workflows

Software reuse is an approach that helps deal with large numbers of similar datasets. Remote-sensing data are created by the

various instruments that sense similar physical parameters, but these instruments provide different geometric, spectral, and radiometric resolutions, or they simply have different data formats when delivered by a data centre. The naive approach is the development of stove-piped analysis and monitoring software that can address the subtleties of the data. Approaching the problem in such a singular manner gradually leads to an unmanageable amount of software with essentially similar functionality. Moreover, due to the sheer number of new datasets, big data can hardly be dealt with by per-task implementations of analysis functions. Software reuse strategies aim to use existing functionality for future tasks; therefore, frequently used procedures in data processing workflows have inherent potential for reuse. NASA has started an internal working group to foster software reuse across their present and foreseen Earth observing missions (Marshall et al., 2011; Mattmann et al., 2010). In Europe, ESA is undertaking efforts to automate user-driven data access, preprocessing, and analysis by establishing a community toolbox of best-practice analysis functions (Meijer et al., 2009).

The reuse of functionality and the ability to invoke it into new workflows is one of the core assets of service-oriented design, where "service" can be defined as a software component that encapsulates functionality behind a standardised interface. Services hide implementation details from their clients and can thus be accessed without knowledge about the internal workings using standard protocols (Erl, 2007; Foster, 2005). Functional abstraction ensures discoverability, i.e., a potential client may inquire about a service catalogue to retrieve a service instance that performs the requested function. To increase the potential for reuse, services mostly rely on established conventions and media formats for data exchange. Georeferenced Tagged Image File Format (GeoTiff), Hierarchical Data Format (HDF)(), Network Common Data Form (NetCDF), Gridded Binary (GRIB), and Geography Markup Language (GML) are some examples of well-known and even standardised formats of spatial data.

Web services for data access have become relatively commonplace. Operational service-oriented architectures (SOAs) that include data processing services are less common. Eberle and Strobl (2012) present a processing-centric SOA to generate remote-sensing products. They are encapsulating NASA software for the Moderate-resolution Imaging Spectroradiometer (MODIS) raw data processing and heat detection functions within a Web Processing Service (WPS; OGC, 2007).

Anselin (2012) discusses a service-oriented workbench for spatial statistics that provides the required analysis functions as Web services. He finds that encapsulating existing functionality into loosely coupled data processing services is technically feasible but not trivial from an application perspective. Therefore, careful choices about the appropriate granularity for atomic functions must be made to ensure their reusability and composability into meaningful workflows.

Several authors (Anselin, 2012; Brauner et al., 2009; Friis-Christensen et al., 2007) discuss issues with large datasets in Web service-based data processing. Invoking functionality from a remote Web service is convenient, but shipping large amounts of data between distant data services, processing services, and clients can become prohibitively expensive in terms of transmission time. Additional applications may only require a subset or an aggregation of the considered datasets but do not require sending of the complete full-resolution data. Approaches to move the algorithmic code to the data rather than the other way around have been discussed by Müller et al. (2010).

Functionality alone is not the only reason for software reuse. Achieving stability, robustness, and correctness for a software component requires significant development efforts. Once available, it is desirable to reuse this software in different application contexts. However, the stove-piped development and unnecessarily tight coupling of components to proprietary data formats or processing platforms are major obstacles for code and component interchange.

2.3. Scalable infrastructures for data-driven analyses

Both grid computing and cloud computing share the same goal of delivering computational and data storage facilities to users and outsourcing the operation of these resources to third parties (Foster et al., 2008). A common purpose of clouds and grids is large distributed data storage and parallel computing.

In a grid, resources are shared among members of virtual organisations that often participate in a joint project. Historically, grids have been viewed as an effort to make consolidated use of expensive computational infrastructure. In remote sensing, grid infrastructures have been applied for recurrent data processing tasks (e.g., Gasster et al., 2008; Hu et al., 2005). Based on the idea of service-oriented science, Foster (2005) developed an outsourcing strategy to allow the distribution of discipline-specific content (i.e., data, software and processes) that is developed and provided by scientists. The provision of lower level, general-purpose infrastructures, such as domain-independent software and hardware, may be handed over to specialist providers. For remote sensing, Aloisio et al. (2004) implemented such a system in a shared grid infrastructure for high-performance computing.

In contrast to grids, cloud computing providers lend their resources to consumers in an ad hoc manner. Cloud computing is a model for on-demand access to "... a shared pool of computing resources [...] that can be rapidly provisioned and released with minimal management effort" (Mell and Grance, 2011). An essential feature of cloud computing is the elasticity of the provided resources that scale rapidly with a changing demand. Ideally, a consumer may request arbitrary computational resources from a virtually unlimited resource pool at any given time. Due to the dynamic up and down scaling mechanism, consumers only pay for the resources when they actually use them. The cloud computing model is comprised of three service models (Mell and Grance, 2011):

Software as a Service (SaaS) allows a consumer to use the provider's application in a cloud infrastructure. The user is completely unaware of the underlying physical hardware resources and is unable to reconfigure the provided software. Depending on the particular software delivered, cloud consumers access SaaS through Web browsers, Web services, or full-fledged remote desktop sessions.

Platform as a Service (PaaS) allows consumers to deploy their own applications and software in a pre-defined environment using pre-defined programming languages, libraries, and APIs supported by the PaaS provider. Consumers still do not have control over physical hardware resources but do have full control over the deployed software.

Infrastructure as a Service (IaaS) provides consumers with physical resources for processing, storage, and networking. Consumers may deploy and run arbitrary software in an IaaS environment, including operating systems and low-level applications.

Applications of cloud services naturally resemble grid computing applications. Assessments of cloud computing for eScience in general (Ramakrishnan et al., 2010) and remote sensing in particular (Dong et al., 2011) yield encouraging results in terms of computing performance and scalability. Considering the flexibility and portability of runtime environments, which is an essential foundation for the intended code sharing, cloud computing outperforms grid computing (Ramakrishnan et al., 2010).

Yang et al., 2011 propose a classification of information technology challenges in geosciences which can be met by cloud

computing. These challenges are data intensity, computing intensity, concurrent access intensity and spatiotemporal intensity. The data intensity challenge is related to the organisation and efficient distribution of large, multi-dimensional, and globally distributed data. Here, cloud computing usually supports large and distributed databases in the backend of data management services (Blower, 2010; Cary et al., 2010). Computing intensity refers to computationally expensive geoprocessing tasks, which frequently occur in data mining or geospatial simulation. Elastic cloud environments provide processing nodes in large numbers and support parallelisation through frameworks such as MapReduce (Dean and Ghemawat, 2008; Deqiang et al., 2012). Since an efficient scheduling mechanism requires control of hardware resources and the software stack the PaaS and IaaS service models are considered to support computing intensive scenarios (Yang et al., 2011). Concurrent access intensity and spatiotemporal intensity refer to the occurrence of peak demands for geodata or geoprocessing tasks. Duty cycles and peak loads for computing resources may occur randomly or follow certain geographic and temporal patterns since users live in different places and time zones. Due to the immanent elasticity of cloud environments, resources can be dynamically provided and released in response to the current demand. Cloud computing has been applied in the backend of data access services and processing services to guarantee minimal response times (Kussul et al., 2012; Schäffer et al., 2010).

3. An interchange model for reusable geoprocessing logic

The previous review demonstrates that code sharing concepts are complementary to data sharing and thus may help deal with big data by moving algorithmic code closer to the data rather than the other way around. Code-sharing concepts can also be employed for the distribution of data across a large array of processing nodes, thus supporting some parallelisation scenarios. The state-of-the-art on big data processing also shows that there is a persistent demand for better interoperability and a standards-based exchange of code to support cloud computing.

Sharing and reusing code at a larger scale may yield community repositories of frequently used data processing functions. Many of the current data infrastructures resemble specialised "workbenches" that deliver analysis capabilities through Web services, Web applications, and possibly specialised middleware. Going even further by not only delivering the pre-configured workbench to the user but also giving him or her the ability to comfortably assemble, configure, and extend his or her own workbenches would be the next logical step. Among other tasks, this step would involve the development of loosely coupled processing services that can be deployed and invoked in a variety of software environments. The following sections present an integrated concept for the sharing and deployment of reusable data processing functions.

3.1. A four-dimensional descriptive model for sharing algorithmic code

The research and development activities associated with service-oriented spatial data processing have led to standardised service interface descriptions of processing services in a distributed environment. The primary subject of the resulting service descriptions is the provided functionality, while the properties of the underlying implementation remain unconsidered. Sharing algorithmic code among different processing nodes, which is the key to achieving elasticity in cloud computing, necessitates a more holistic description for service-oriented software components with a particular consideration of the required runtime environment, i.e., required software platforms and computing hardware.

Fig. 1 presents an abstract model for describing exchangeable geoprocessing logic along four dimensions. The contracted functionality (1) is a machine-readable description of the provided functionality. It describes the interface of the implemented algorithmic code in a standardised manner and may contain additional information concerning the algorithms applied. In cloud computing, a functional description can be placed on the same level as the SaaS service model. The contracted platform (2) characterises the runtime environments in terms of basic software platforms. This dimension corresponds to the PaaS service model in cloud computing. Coarse-grained software dependencies (e.g., particular sets of installed libraries and interpreters) of the code that is to be shared are specified by this dimension. The contracted infrastructure (3) refers to particular hardware resources that are required at runtime for a reliable execution and corresponds to the IaaS service model. The code to be shared could be resource demanding and is intended to be run on a high-memory, multi-core server, or it could be resource-efficient, allowing execution on low-memory, single-core computers. Resource-efficient code is especially interesting in concurrent runtime environments, where the same function is executed simultaneously for parallelisation. Resource-efficient implementations are found in the backend of application servers but may also offer cost-efficient data processing in small cloud computing instances. The legal contract (4) refers to the exploitation rights associated with the shared algorithmic code. It covers all legal aspects, from usage rights to liabilities. Producers may grant free use of their software for non-commercial purposes but demand a license fee for commercial applications.

Such a unified description model allows software developers to specify exactly what function the components perform and in which environments they can be run. Software inventories building on the proposed descriptive model will be able to deliver reusable software components that meet given criteria concerning exploitation rights, hardware requirements, platform dependencies, and functionality. Users may query those inventories to search for existing functionality that suits their needs and fits into their existing infrastructure. Another possible scenario would be an inquiry for cloud services that provide sufficient capabilities to run a particular component.

3.2. Conceptual architecture

The conceptual architecture in Fig. 2 presents an overview of the components required for the service-oriented exchange of geoprocessing logic in a distributed data infrastructure. The main building blocks are code management services, data management services, and workflow and processing facilities that finally execute the computing logic on the data. The architecture supports typical publish-find-bind scenarios; for example, download services for geoprocessing logic are registered with independent discovery services (publish). Consumers of geoprocessing logic may query the

Fig. 1. Dimensions used to describe algorithmic code: (1) contracted functionality, (2) platform, (3) infrastructure and (4) license or legal contract associated with a particular implementation of an abstract functionality.

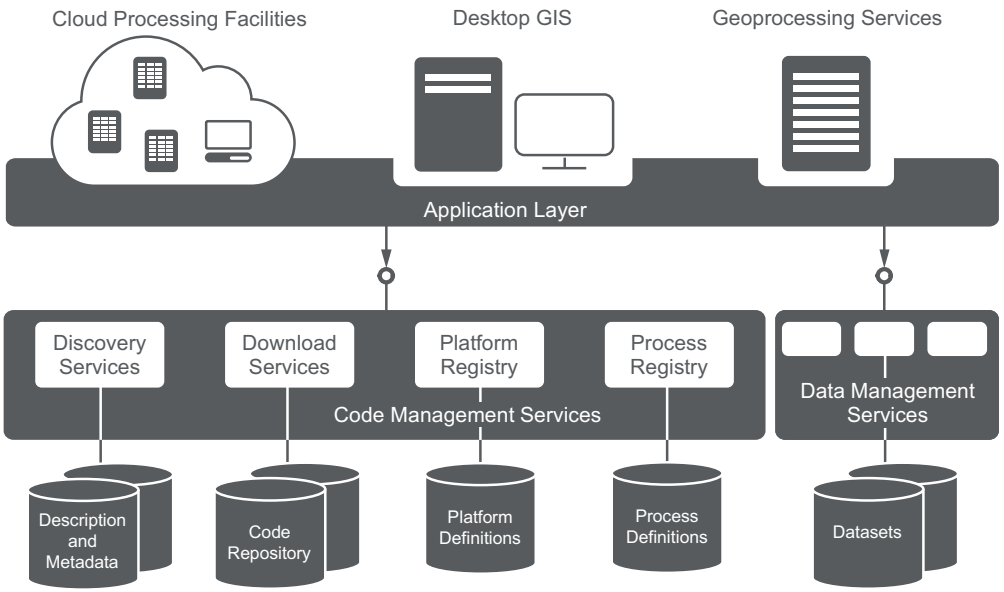


Fig. 2. Conceptual architecture for a distributed code-sharing and geoprocessing environment.

catalogue to find geoprocessing logic that suits their needs (find). Harmonised identifiers and definitions of functionality and processing platforms are stored in common registries. After a suitable implementation has been found, the consumer's computing site (located in the application layer) can be connected to a particular provider's download service (bind). Depending on the specific requirements, this binding may be permanent (e.g., to receive automatic updates) or temporary (e.g., for one-time executions). A platform registry may also be used to establish a well-known set of platform definitions. PaaS providers may use such a registry to provide frequently used pre-configured processing platforms to their customers.

Possible processing facilities range from stand-alone applications in desktop GIS, via geoprocessing services where clients invoke remote services to perform data analysis tasks, to cloud environments offering computational power and bandwidth to process large datasets.

3.3. A packaging scheme for reusable geoprocessing logic

Similarly to data exchange among different systems (ISO, 2005), the redistribution of geoprocessing logic requires a common packaging and exchange format. The approach presented in this chapter is based on the work of Müller et al. (2010), who introduced moving-code packages as structured zip files that contain a description of the provided geoprocessing logic and the platform requirements, as well as the specific implementation. In this paper, that approach is extended and modularised to support the description of PaaS, IaaS, and licenses, as well as the ability to use arbitrary standards for the functional description. The moving-code concept uses a workspace to store the algorithmic code and possible tightly coupled datasets. Workspace concepts are known to many GIS and other data processing software programs, thus making moving-code packages a versatile approach that covers a broad range of applications. In addressing the four dimensions shown in Fig. 1, the class diagram in Fig. 3 presents a conceptual model for moving-code packages.

The schema has been designed for extensibility, allowing different functional representations of the provided logic. For the functional contract, the WPS standard is used as a basic means to communicate a declarative specification of the geoprocessing logic. However, other widely adopted standards, such as the Web Service Definition Language (WSDL; W3C, 2007), may also be used for this purpose.

The contracted platform is formalised as a list of unique identifiers of runtime components that have to be provided by a deployment platform. The contracted platform describes high-level dependencies, such as third-party libraries and software packages that must be available on the target platform. Individual runtime components are represented by unique identifiers, which are registered and further described at the platform registry in Fig. 2. Runtime components are self-aggregating objects; namely, a coarse-grained runtime component may aggregate other, more finely grained runtime components. For example, different versions of the well-known library GDAL are aggregated by a variety of other software programs, such as GRASS and ArcGIS.

Describing the contracted infrastructure requires a compromise between detail and manageability. Detailed information on hardware properties is often desirable in high-performance computing, but it introduces a significant overhead for general-purpose applications. In favour of manageability, the contracted infrastructure is described in line with the Open Cloud Computing Interface (OCCI, OGF, 2011a) specification, an emerging standard in cloud computing that will be supported by numerous cloud infrastructure providers. This standard also allows for the description of computation infrastructure at a more general and manageable level (OGF, 2011b) and focuses mainly on computing architecture, processing power, and memory consumption. An additional advantage of using OCCI lies in the direct comparability of hardware capabilities and requirements, allowing an instant decision to be made on deployment feasibility. Users may also decide to select resource-efficient implementations if multiple moving-code packages offer the same functionality.

An interoperable approach to the exchange and communication of exploitation rights is the Creative Commons license (Abelson et al., 2008). The standardised modular licensing scheme is applicable to licensing requirements of open-source software and is the preferred mechanism to describe the exploitation rights for moving-code packages. In catalogues or inventories, this scheme allows users to query desired licensing requirements as search

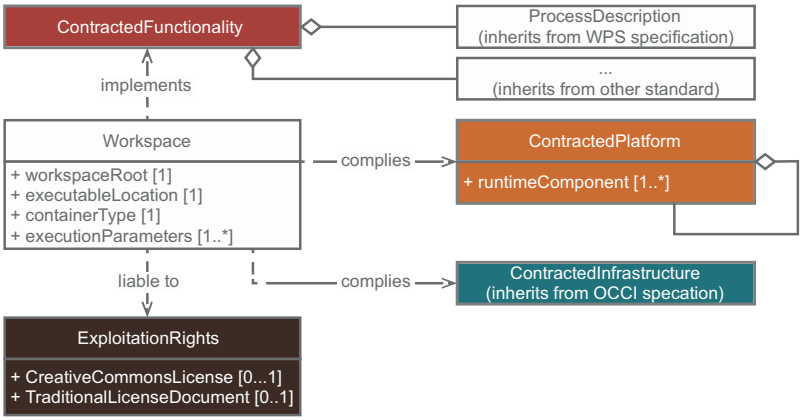


Fig. 3. Structural view of a moving-code package.

criteria for reusable code. As not all implementations use Creative Common license types, practical considerations demand a way to express commercial licenses and other types of open source licenses that are neither modular nor standardised. In this case, the legal text is supplied by a URL reference where the license document can be obtained.

The workspace description holds information about the workspace structure, the location of the executable code (e.g. a particular file), and instructions for the parameter mapping between the executable code and the service-oriented description. Workspaces contain the algorithmic code that realises the functional contract and must comply with the contracted platforms and the contracted types of infrastructure.

3.4. Machine-readable feeds for software discovery and download

Existing ISO metadata standards are suitable for the description of geospatial data and services. Today, ISO 19115 (ISO, 2009) forms the basis of the metadata models of most geodata catalogues. However, the description of processing functionality in standardised catalogues is still an issue. The ISO 19115 standard only allows processing operations to be described as supplementary information in the lineage statement of a particular dataset. Cataloguing generic and independent processing operations is not foreseen. Lacking a standardised concept for a processing inventory, a hybrid approach was chosen that uses the Atom Syndication Format (IETF, 2005a) for the discovery and download of geoprocessing functionality in the code management services (compare with Fig. 2). It is a lean, extensible XML-based format for platform-independent information exchange, and its usage ranges from exchanging simple text news to sharing multimedia content or complex data. News readers as well as other aggregator software may subscribe to Atom feeds and obtain new or updated content by a pull mechanism. The format's capabilities to exchange information in multiple representations, being human- as well as machine-readable, and the provision of a simple versioning mechanism make it a suitable exchange vehicle for moving-code packages. Atom feeds delivering geoprocessing functionality are called geoprocessing feeds in this paper. The human-readable part of the feed's content allows users to study the provided functionality and pick those packages that suit their needs. The machine-readable part of the same feed can be invoked by any of the application environments in Fig. 2. Additionally, after subscription, these machines may occasionally synchronise with the geoprocessing feed to obtain an updated algorithmic code.

Geoprocessing feeds are also a convenient instrument with which to distribute moving-code packages in a dedicated, private network that consists of multiple processing nodes. Similarly to newsfeeds, a customised or private geoprocessing feed may contain unions or filtered subsets of external feeds. Selection criteria may relate to any of the four dimensions discussed in Section 3.1. Hosting the moving-code packages in-house also increases resilience to external server outages and speeds up the internal component distribution process.

4. Proof-of-concept: An anomaly detection application

In this section, a real-world example will be presented that uses moving-code packages and processing feeds to describe and exchange reusable functionality in a remote-sensing analysis. The related proof-of-concept implementation has been realised within the framework of the OGC Web Services Phase 8 testbed (OWS-8, OGC, 2012).

Anomaly detection in time series data is a common task in data mining and monitoring applications. The term "anomaly" may relate to the shape, size, and duration of spatial phenomena or their intensity within a certain region (McKenna and Gutierrez, 2011). The following example presents a procedure for detecting intensity anomalies in vegetation indices. The calculation procedure was originally developed by Samanta et al. (2011) to assess the sensitivity of Amazon rainforests to dry-season droughts. The available MATLAB process was analysed and decomposed into reusable pieces of geoprocessing logic for subsequent implementation as reusable moving-code packages (Fig. 4).

Inputs to the process chain are the MODIS MOD13A2 and MOD12Q1 products. The MOD13A2 product contains the pre-computed Enhanced Vegetation Index (EVI), an index that is related to photosynthetic activity and greening, and some quality information that can be used as a measure of reliability for the EVI value. The MOD12Q1 contains a forest mask that is used to sift out all non-forest pixels. Both inputs are so-called MODIS granules, which are spatio-temporal tiles containing data for a particular spatial and temporal extent. These granules are somewhat specific to the orbital parameters of NASA's Terra satellite and must first be aggregated to monthly values to perform any further computations. A preprocessing step involves all necessary steps that are particularly related to the HDF-EOS data format and metadata as well as the product-specific processing steps.

The subsequent processing steps are local operations on raster data or more formally discrete surface coverages with quadrilateral grid geometry (compare with ISO, 2007), which can be handled within Tomlin's map algebra operators (Tomlin, 1990). From the monthly EVI values, the quarterly data for the summer season are obtained through a local mean function. Based on these annual quarterly data, a *z*-score function computes the pixel-wise

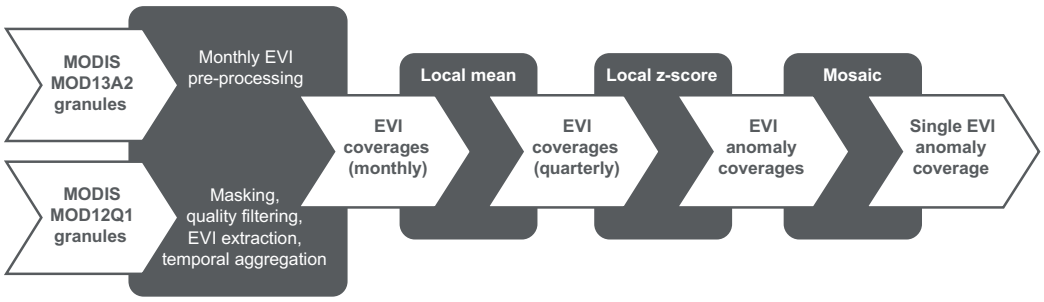


Fig. 4. Processing workflow for the anomaly calculation procedure (adapted from Samanta et al., 2011).

difference between the current year's EVI and the corresponding values from a reference period in standard deviations. Values below -1σ are considered as "browning", indicating areas with significantly less photosynthetic activity than that in the reference period and that are, therefore, potentially suffering from drought effects (Samanta et al., 2011). The final process step is a common mosaicking operation that assembles the individual tiles, each corresponding to the geographic extent of the respective input granules to a larger raster covering the entire Amazon region in South America.

This use case is a good example of an embarrassingly parallel problem: most functions are independently performed on a large number of individual granules and can thus be easily processed in parallel. From the many possible decompositions of the original procedure, the particular decomposition in Fig. 4 yields quite generic functions that have a high potential for reuse and allow for a high degree of parallelisation.

The local mean is a generic operation from Tomlin's Map Algebra and a general-purpose function in many workflows. The *z*-score process is similarly versatile for anomaly detections; specifically, the applied standardisation procedure is a common subtask in spatial multi-criteria evaluation workflows; a preprocessing step in visual analysis; or, as in this case, a statistical transformation for anomaly detection. The last sub-process for mosaicking is a frequently applied stitching operation. Many data producers provide tiled data because they are easier to handle and index than very large datasets. Consequently, when a larger geographic area is to be covered with spatially partitioned data, it must be reassembled with a mosaicking operation. The first step in the chain is very specific to the input data format (HDF-EOS) and the supplied metadata in the MODIS granules. It is therefore advantageous to perform granule-dependent masking, quality filtering and temporal aggregation in a single step that delivers the refined data in a more common format.

The suggested workflow exploits the tiled structure of the original data and allows a high degree of parallelisation in the first steps of the workflow. For a typical run, 336 pairs of MOD13A2 granules with an overall size of 13.4 Gigabytes have to be processed. This initial number is gradually reduced to a single anomaly coverage of 16 Megabytes in the last step. The frequency and duration of each sub-process are shown in Fig. 5. For a sequential execution, the overall duration amounts to approximately 26 min. In contrast, for a parallel execution, the theoretical minimum duration is no more than 16 s, a decrease of two orders of magnitude. In the latter case, the calculation procedure will become *I*/*O* bound, which means that obtaining access to a physical storage device will become the main bottleneck. Clusters or cloud infrastructures with distributed, shared storage are the only technical means to counter the expected *I*/*O* bandwidth issues. By using moving-code packages for the sub-process implementations, the components can be deployed in arbitrary infrastructures, including a local workstation or desktop computer, a geoprocessing server, or a high-performance cloud with a large number of processing nodes. The capabilities of the infrastructure dictate the optimal workflow and the appropriate degree of parallelisation.

All steps were re-implemented in GDAL/Python using Numpy array operations for speed and tiling mechanisms for a low memory footprint. The performance data were obtained on a Core2Duo processor (single core use) at a 2.4-GHz clock speed. Due to the use of open-source software instead of the original MATLAB scripts and the suitability for execution, even in older or poorly performing systems, the sub-processes are potentially valuable for a broad range of users. At the same time, the low memory footprint and the robust and parallel design suggest a deployment behind Web service facades or in cloud environments where the individual executions can be split among a large number of processing nodes.

Experiences with geoprocessing feeds and the atom protocol for publishing, downloading, and announcing updates of computing logic are generally good. Despite the lack of a full-fledged management infrastructure for code-sharing, process feeds were found a convenient way to deliver basic functional descriptions as well as machine-readable moving-code packages to the users. A package description example of one of the sub-processes can be found in the appendix. To further assess scalability, this first prototype needs to be instantiated for different scenarios and in different cloud computing environments. In this way, potential scaling limits could be explored and indicators could be derived to better rate cloud computing platforms for data-intensive processing.

5. Conclusion and outlook

Making progress in the design of well-defined descriptions of geoprocessing logic is one of the remaining research challenges in geoinformation science and will contribute to making the proposed concept a powerful tool for seamless and comprehensive documentation in geospatial data handling. The four-dimensional contract for geoprocessing logic aids in the sharing of existing implementations with many users and removes obstacles that keep people from reusing each other's codes. However, the lack of common standards for catalogues of processing functions is an obstacle for interoperability. A unified approach to aligning the requirements for free or structured search interfaces, in-depth functional documentation, and comparability of the provided functionality is needed to further enhance the approach proposed in this paper.

Considering the various opportunities associated with mobile and reusable code, it may be useful to standardise the packaging format further. Fig. 6 shows how the different parts of the package description relate to existing standards. For infrastructure and functionality, a fully standardised description is possible. Currently, platforms cannot be described in a fully structured manner

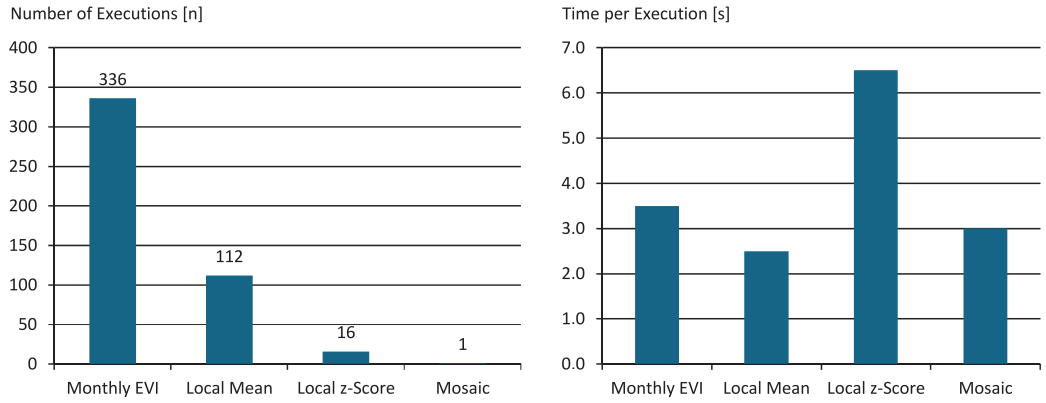


Fig. 5. Performance and data reduction data associated with the Amazon drought calculation workflow (Fig. 4).

and must, therefore, be identified by unique resource identifiers (i.e., Uniform Resource Locators, URLs; IETF, 2005b). Exploitation rights can be specified by using machine-readable licenses such as Creative Commons. Standardising the workspace structure and description is probably the most difficult task. Workspace concepts are widely used in practice and there is no dispute about their versatility and usefulness. However, there is no real standard or single best practice regarding how a portable workspace should be designed. Moving-code packages make only very generic assumptions about the workspace, e.g., that it must contain an executable script or binary file in a well-known format. Based on that format, the parameter mapping between the functional description and the algorithmic code is defined. Similarly to mapping conventions between JAVA methods and WSDL interfaces, these mappings are a further subject for standardisation.

Considering the efforts by individual institutions to foster software reuse, sharing software in a larger ecosystem seems to be the next logical step. Software that works well is a valuable resource, and its maintenance requires resources; therefore, splitting these efforts among community members is reasonable. Communities that are already discussing best-practice toolboxes for data analysis (e.g., Meijer et al., 2009) may benefit from the presented code sharing approach. With the ability to investigate catalogues for existing processing functions, users and software developers can save time and effort by assessing existing functions for their purposes and make well-informed make-or-buy decisions.

In addition, code sharing could be used in conjunction with workflow sharing concepts. Internet platforms such as myExperiment already promote the exchange of scientific workflows to facilitate the assessment and reproducibility of data-intensive research (De Roure et al., 2009; Missier et al., 2010). As in any other workflow system, task-specific functionality is obtained by orchestrating reusable atomic components, many of which are Web services. Coupling a workflow management system with a code sharing infrastructure would provide a flexible extension mechanism that allows users to take advantage of new basic functions and their execution in high-performance environments.

Two problems associated with the act of code sharing could not be solved within the framework presented in this paper: trust and security. Moving-code packages per se cannot guarantee that an

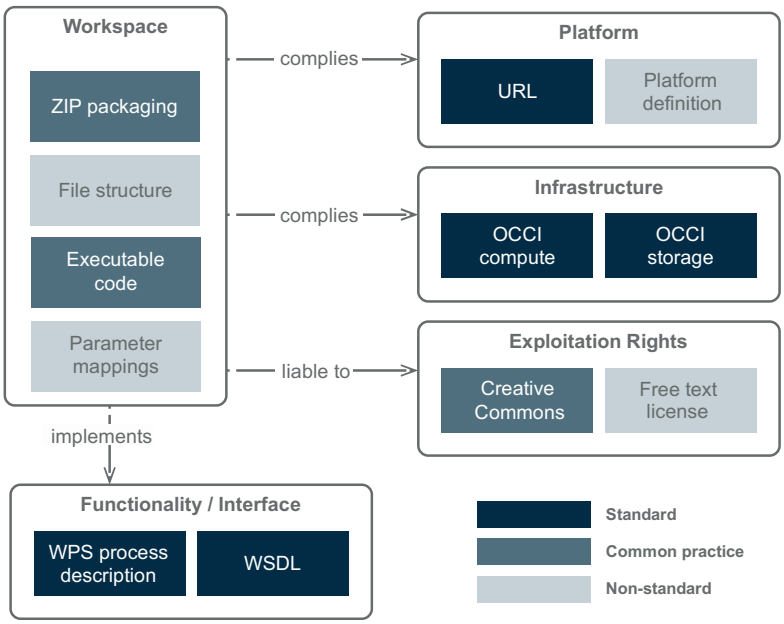


Fig. 6. A standards-based view of the code-packaging scheme.

implementation complies with the functional contract, behaves well in other environments, or does not contain harmful code. Building trust and enforcing implementation quality as well as security require a good review system or a community platform. Checking for malicious code requires at least the invocation of a virus and malware scanner in the publication process, and more sophisticated approaches may involve test runs in a sandbox environment. The community building process may very well take place on a Web platform such as the Apple iTunes App Store or the Android Market. Providers of geoprocessing logic can publish their algorithms in such a marketplace, and all other users will be able to utilise them. An integrated review process allows all par-

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- shortened package description example -->
<mc:packageDescription ...>
  <!-- Functional description based on the WPS 1.0 standard -->
  <mc:contractedFunctionality>
     <mc:wpsProcessDescription wps:processVersion="1">
        <ows:Identifier>org.somenamespace.algorithms.raster. zscore</ows:Identifier>
        <ows:Title>Cell wise z-score algorithm</ows:Title>
        <ows:Abstract>Computes a z-score cell by cell</ows:Abstract>
        <DataInputs>
           <Input minOccurs="1" maxOccurs="1000">
              <ows:Identifier>sample</ows:Identifier>
              <ows:Title>Sample Data</ows:Title>
              <ows:Abstract>Collection of single band rasters to compute sigma and mean from.</ows:Abstract>
                ... <!-- data format specification -->
           </Input>
           <Input minOccurs="1" maxOccurs="1">
              <ows:Identifier>rawscore</ows:Identifier>
              <ows:Title>Raw Score</ows:Title>
              <ows:Abstract>A single band raster with raw values that shall be transformed to a z-score.</ows:Abstract>
                ... <!-- data format specification -->
           </Input>
        </DataInputs>
        <ProcessOutputs>
           <Output>
              <ows:Identifier>zscore</ows:Identifier>
              <ows:Title>Z-score</ows:Title>
              <ows:Abstract>A single band raster with the z-score values computed from the raw score.</ows:Abstract>
                ... <!-- data format specification -->
           </Output>
        </ProcessOutputs>
     </mc:wpsProcessDescription>
  </mc:contractedFunctionality>
```

Contracted Functionality

```
  <!-- First valid platform -->
  <mc:contractedPlatform>
     <mc:requiredRuntimeComponent>http://someregistry.org/ python-2.5</mc:requiredRuntimeComponent>
     <mc:requiredRuntimeComponent>http://someregistry.org/ gdal-1.8</mc:requiredRuntimeComponent>
     <mc:requiredRuntimeComponent>http://someregistry.org/ gdal-1.8-python-2.5</mc:requiredRuntimeComponent>
  </mc:contractedPlatform>

  <!-- Other valid platforms, i.e. other compatible gdal and python versions -->
  <mc:contractedPlatform>
     ...
  </mc:contractedPlatform>
```

Contracted Platform

```
  <!-- Required computing resources -->
  <mc:contractedInfrastructure>
     <!-- Code runs in a single core environment -->
     <mc:occi.compute.cores>1</mc:occi.compute.cores>
     <!-- Consumes no more than 200 MB of system memory -->
     <mc:occi.compute.memory>0.2</mc:occi.compute.memory>
     <!-- Temporary storage space is not a critical factor for a reliable execution -->
     <!-- Processor speed is not a critical factor for a reliable execution -->
     <!-- Computing architecture is not a critical factor since python scripts are generally
        considered platform independent.-->
  </mc:contractedInfrastructure>
```

Contracted Infrastructure

```
  <!-- License associated wih this implementation -->
  <mc:exploitationRights>
     <mc:creativeCommonsLicense>
        <mc:cc.license>http://creativecommons.org/licenses/by-sa/3.0/ </mc:cc.license>
        <mc:dct.title>z-Score algorithm implementation for raster data</mc:dct.title>
        <mc:cc.attributionName>Author Name</mc:cc.attributionName>
        <mc:cc.attributionURL>http://somesite.org</mc:cc.attributionURL>
     </mc:creativeCommonsLicense>
  </mc:exploitationRights>
```

Legal Contract

```
  <!-- Description of the workspace structure -->
  <mc:workspace>
     <!-- Workspace root and location of the executable -->
     <mc:workspaceRoot>./ztransform</mc:workspaceRoot>
     <mc:executableLocation>./ztransform.py</mc:executableLocation>
     <!-- Type of executable -->
     <mc:containerType>http://someregistry.org/ pythonscript-2.5</mc:containerType>
     <mc:executionParameters>
        <!-- Mapping instructions for the first input -->
        <mc:parameter>
           <mc:prefixString/>
           <mc:suffixString/>
           <mc:separatorString/>
           <mc:positionID>3</mc:positionID>
           <mc:functionalInputID>sample</mc:functionalInputID>
           <mc:functionalOutputID/>
        </mc:parameter>
        <mc:parameter>
           ... <!-- Other parameter mappings -->
        </mc:parameter>
     </mc:executionParameters>
  </mc:workspace>
</mc:packageDescription>
```

Workspace

ticipants to comment on and rate the published algorithms. Such a review process can be extended toward quality guarantees in terms of correctness and robust execution. For the scientific community, the visibility and potential reuse of their software may lead to new output metrics in scientific research. Publishing software along with a written paper is already supported by some publishing companies. Lowering the barrier for reuse and ad hoc execution is a next step. Additional impact metrics for geoprocessing logic produced by science projects may help to tackle a persisting issue in data-driven science, namely the availability of versatile, peer-reviewed tools for data curation and analysis.

Acknowledgments

The research leading to these results has received funding from the German Federal Ministry of Education and Research under grant agreement n° 01LL0901C and the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n° 244100.

Appendix A

This appendix consists of a descriptive example of a moving code package containing an implementation of the *z*-score process presented in the paper. It consists of the following parts: contracted functionality, platform and infrastructure description, exploitation rights, and workspace description.

Contracted functionality follows the OpenGIS WPS standard. In this standard, the functionality is identified by some type of process signature that defines inputs and outputs and provides simple descriptions about the implemented algorithm. The given *z*-score process requires two inputs: a raw score and a sample set (both are raster data). The sample set is used to calculate the values for σ and μ; the raw score is then converted to a *z*-score which is returned by the service. Additionally, this interface description may point to a process profile that provides additional metadata about the algorithm, i.e., through hypertext documents. However, further research is required to fully capture the process semantics.

The contracted platform lists a set of platform components that must be available on the target platform to reliably execute the contained code. In the *z*-score example, it is demanded that the platform must have a Python 2.5 runtime environment and a GDAL 1.8 standard library installed. There must also be a Python to GDAL connector in place. Each of the runtime components is specified by a resolvable URL that can be used to unambiguously identify each software component. The URL links to a hypertext document that provides the required description and definition of each component. There may be multiple contracted platforms, each containing a confirmed combination of runtime components. The code contained in the package is expected to run in any of these contracted platforms.

The contracted infrastructure is a subset of the elements defined in the OCCI specification. The OCCI elements are used to characterise the resource consumption of the code. Running the example *z*-score code requires one processor core and guarantees memory consumption below or equal to 200 Megabytes during execution (achieved by adaptive tiling mechanism in the implementation). The example does not require a certain computing architecture, clock speed, or extra amount of temporary disc space for reliable execution.

The example package comes with a Creative Commons sharealike license, which is stated in the exploitation rights section. This particular license permits free use of the code as well as the ability to alter or enhance it as long as it is re-published under the same type of license. This type of licence is frequently found in association with open-source software to ensure that enhancements are transferred back into the open source community.

The workspace description contains information about the type and location of the executable code in the package. In the *z*-score process example, the executable is a plain python script that can be called in a command line fashion. Additionally, the workspace description provides a syntax mapping between the WPS interface and the command line.

References

Abelson, H., Adida, B., Linksvayer, M., Yergler, N., 2008. ccREL: The creative commons rights expression language. http://wiki.creativecommons.org/images/d/d6/Ccrel-1.0.pdf (accessed 12.11.12).

Aloisio, G., Cafaro, M., Epicoco, I., Quarta, G.C., 2004. A problem solving environment for remote sensing data processing. In: Proceedings of the International Conference on Information Technology: Coding Computing (ITCC), Lecce, Italy, 5–7 April, vol. 2, 2004, pp. 56–61.

Anselin, L., 2012. From Space Stat to CyberGIS. International Regional Science Review 35 (2), 131–157.

Bernabé, S., Plaza, A., Reddy Marpu, P., Atli Benediktsson, J., 2012. A new parallel tool for classification of remotely sensed imagery. Computers & Geosciences 46, 208–218.

Blower, J.D., 2010. GIS in the cloud: implementing a web map service on Google App Engine. In: Proceedings of the 1st International Conference and Exhibition on Computing for Geospatial Research & Application (COM.Geo), Bethesda, MD, USA, 21–23 June, 2010, pp. 34:1–4.

Brauner, J., Foerster, T., Schaeffer, B., Baranski, B., 2009. Towards a Research Agenda for Geoprocessing Services. In: Proceedings of the 12th AGILE Conference, Hannover, Germany, 2–5 June, pp. 124:1–12.

Cary, A., Yesha, Y., Adjouadi, M. ,Rishe, N., 2010. Leveraging Cloud Computing in Geodatabase Management. In: Proceedings of the IEEE International Conference on Granular Computing (GrC) 2010, San Jose, CA, USA, pp. 73-78.

Christophe, E., Michel, J., Inglada, J., 2011. Remote sensing processing: from multicore to GPU. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 4 (3), 643–652.

Clery, D., Voss, D., 2005. All for one and one for all. Science 308 (5723), 809.

Craglia, M., de Bie, K., Jackson, D., Pesaresi, M., Remetey-Fülöpp, G., Wang, C., Annoni, A., Bian, L., Campbell, F., Ehlers, M., van Genderen, J., Goodchild, M., Guo, H., Lewis, A., Simpson, R., Skidmore, A., Woodgate, P., 2012. Digital Earth 2020: towards the vision for the next decade. International Journal of Digital Earth 5 (1), 4–21.

De Roure, D., Goble, C., Stevens, R., 2009. The design and realisation of the Virtual research environment for social sharing of workflows. Future Generation Computer Systems 25 (5), 561–567.

Dean, J., Ghemawat, S., 2008. MapReduce: simplified data processing on large clusters. Communications of the ACM 51 (1), 107–113.

Deqiang, G., Keping, D., Yonghua, Q., Yuzhen, Z., Linli, L., 2012. Remote sensing algorithm platform in Windows Azure. In: Proceedings of the 20th International Conference on Geoinformatics (GEOINFORMATICS), Hong Kong, China, (doi:10.1109/Geoinformatics.2012.6270351).

Dong, J., Xue, Y., Chen, Z., Xu, H., Li, Y.C.I., 2011. Analysis of remote sensing quantitative inversion in cloud computing. In: Proceedings of the IEEE Geoscience and Remote Sensing Symposium (IGARSS), Vancouver, BC, Canada, pp. 4348-4351.

Eberle, J., Strobl, C., 2012. Web-based geoprocessing and workflow creation for generating and providing remote sensing products. Geomatica 66 (1), 13–26.

Eionet, 2012. Corine Land Cover 2006. http://sia.eionet.europa.eu/CLC2006. (accessed 12.11.12).

Erl, T., 2007. SOA Principles of Service Design. Prentice Hall PTR Upper Saddle River, NJ, USA, pp. 573.

ESA, 2012. Earthnet 2012. http://earth.esa.int. (accessed 12.11.12).

Foster, I., 2005. Service-oriented science. Science 308 (5723), 814–817.

Foster, I., Yong, Z., Raicu, I., Lu, S., 2008. Cloud Computing and Grid Computing 360-Degree Compared. Grid Computing Environments Workshop, 2008 (GCE '08), (doi:10.1109/GCE.2008.4738445).

Friis-Christensen, A., Ostländer, N., Lutz, M., Bernard, L., 2007. Designing service architectures for distributed geoprocessing: challenges and future directions. Transactions in GIS 11 (6), 799–818.

Gasster, S.D., Lee, C.A., Palko, J.W., 2008. Remote sensing Grids: architecture and implementation. In: Plaza, A., Chang, C. (Eds.), High Performance Computing in Remote Sensing. Chapman & Hall, Boca Raton, FL, USA, pp. 203–236.

GEO, 2012. The global earth observation system of systems, http://www.earthobservations.org/geoss.shtml. (accessed 12.11.12).

GMES, 2012. GMES.info, http://www.gmes.info/. (accessed 12.11.12).

Gray, J., 2009. eScience: a transformed scientific method. In: A.J.G. Hey, S. Tansley, K.M. Tolle (Eds.), The fourth paradigm: data-intensive scientific discovery. Microsoft Research, Redmond, WA, pp. xvii–xxxi.

Happ, P.N., Ferreira, R.S., Bentes, C., Costa, G.A.O.P., Feitosa, R.Q., 2010. Multiresolution Segmentation: a Parallel Approach for High Resolution Image Segmentation in Multicore Architectures. In: E.A. Addink, F.M.B. Van Coillie (Eds.), The International Archives of the Photogrammetry, Remote Sensing and

Spatial Information Sciences, vol. XXXVIII-4/C7. http://www.isprs.org/proceedings/xxxviii/4-c7/pdf/Happ_143.pdf (accessed 12.11.12).
Hu, Y., Xue, Y., Tang, J., Zhong, S., Cai, G.C., 2005. Data-Parallel Method for Georeferencing of MODIS Level 1b Data Using Grid Computing. In: Sunderam, V.S., Albada, G.D., Sloot, P.M.A., Dongarra, J. (Eds.), Computational Science – ICCS 2005. Springer, Berlin, pp. 883–886.
IETF, 2005a. The Atom Syndication Format. http://www.ietf.org/rfc/rfc4287. (accessed 12.11.12).
IETF, 2005b. Uniform Resource Identifier (URI): Generic Syntax. http://tools.ietf.org/html/rfc3986. (accessed 12.11.12).
ISO, 2005. Geographic information – Encoding. International standard, ISO 19118.
ISO, 2007. Geographic information – Schema for coverage geometry and functions. International standard, ISO 191123.
ISO, 2009. Geographic information – metadata – part 2: extensions for imagery and gridded data. International standard ISO 19115–2, 2009.
Kiehle, C., Greve, K., Heier, C., 2007. Requirements for Next Generation Spatial Data Infrastructures - Standardized Web Based Geoprocessing and Web Service Orchestration. Transactions in GIS 11 (6), 819–834.
Knuth, D., 1997. Art of computer programming, volume 2: seminumerical algorithms (3rd Edition). Addison-Wesley Professional, Reading, MA, USA, pp. 784.
Korting, T.S., Castejon, E.F., Fonseca, L.M.G., 2011. Divide and Segment – An alternative for parallel segmentation. In: Proceedings XII Brazilian Symposium on Geoinformatics (GEOINFO), Campos do Jordão, Brazil, 27–29 November, pp. 97–104.
Kussul, N., Mandl, D., Moe, K., Mund, J., Post, J., Shelestov, A., Skakun, S., Szarzynski, J., Van Langenhove, G., Handy, M., 2012. Interoperable infrastructure for flood monitoring: sensorweb, grid and cloud. IEEE Journal of Selected Topics in Applied Earth Observations and Remote Sensing 5 (6), 1740–1745.
Marshall, J.J., Downs, R.R., Mattmann, C.A., 2011. Software reuse methods to improve technological infrastructure for e-Science. In: Proceedings of the IEEE International Conference on Information Reuse and Integration (IRI), Las Vegas, NV, USA, 3–5 August, pp. 528–532.
Mattmann, C.A., Downs, R.R., Marshall, J.J., Most, N.F., Samadi, S., 2010. Reuse of software assets for the NASA Earth science decadal survey missions. In: Proceedings of the IEEE Geoscience and Remote Sensing Symposium (IGARSS), Honolulu, HI, USA, 25–30 July, pp. 1687–1690.
McKenna, S.A., Gutierrez, K., 2011. Spatial-Temporal Event Detection in Climate Parameter Imagery. Sandia National Laboratories, Department of Commerce. Springfield, VA, USA, pp. 42.
Meijer, Y.J., Fehr, T., Koopman, R.M., Pellegrini, A., Busswell, G., Williams, I., De Mazière, M., Niemeijer, S., van Deelen, R., 2009. GECA: ESA's Next Generation Validation Data Centre. In: Proceedings of the Fringe 2009 Workshop, Frascati, Italy, 30 Novenber - 4 December. http://earth.eo.esa.int/workshops/fringe09/YaskaMeijer.pdf (accessed 12.11.12).
Mell, P., Grance, T., 2011. The NIST Definition of Cloud Computing, Recommendations of the National Institute of Standards and Technology. National Institute of Standards and Technology, Gaithersburg, MD, USA, 7p. http://csrc.nist.gov/publications/nistpubs/800-145/SP800-145.pdf (accessed 12.11.12).
Missier, P., Soiland-Reyes, S., Owen, S., Tan, W., Nenadic, A., Dunlop, I., Williams, A., Oinn, T., Goble, C., 2010. Taverna, reloaded. In: Scientific and Statistical Database Management: 22nd international conference on Scientific and statistical database management (SSDBM), Springer, Berlin, pp. 471–481.
Müller, M., Bernard, L. and Brauner, J., 2010. Moving code in spatial data infrastructures – web service based deployment of geoprocessing algorithms. Transactions in GIS, vol. 14(S1), pp. 101–118.
Muñoz-Marí, J., Plaza, A.J., Gualtieri, J.A., Camps-Valls, G., 2009. Parallel Implementations of SVM for Earth Observation. In: Xhafa, F. (Ed.), Parallel Programming and Applications in Grid, P2P and Networking systems. IOS Press, Amsterdam, Netherlands, pp. 292–312.
NASA, 2012. data.NASA, http://data.nasa.gov/. (accessed 12.11.12).
OGC, 2007. OpenGIS Web Processing Service, Version 1.0.0. Wayland, MA, USA, OGC document 05-007r7. http://portal.opengeospatial.org/files/?artifact_id=24151 (accessed 12.11.12).
OGC, 2012. OGC Web Services, Phase 8, http://www.opengeospatial.org/projects/initiatives/ows-8 (accessed 12.11.12).
OGF, 2011a. Open Cloud Computing Interface Specification. http://occi-wg.org/about/specification/ (accessed 12.11.12).
OGF, 2011b. Open Cloud Computing Interface Specification – Infrastructure. OGF document GFD-P-R.184. http://www.ogf.org/documents/GFD.184.pdf (accessed 12.11.12).
Pallickara, S.L., Malensek, M., Pallickara, S., 2011. On the Processing of Extreme Scale Datasets in the Geosciences. In: Furht, B., Escalante, A. (Eds.), Handbook of Data Intensive Computing. Springer, New York, pp. 521–537.
Plaza, A.J., 2006. Heterogeneous Parallel Computing in Remote Sensing Applications: Current Trends and Future Perspectives. IEEE International Conference on Cluster Computing, Barcelona, Spain, 25–28 September. (doi:/10.1109/CLUSTR.2006.311903).
Raicu, I., Zhang, Z., Wilde, M., Foster, I., Beckman, P., Iskra, K., Clifford, B., 2008. Toward loosely coupled programming on petascale systems. In: Proceedings of the ACM/IEEE conference on Supercomputing 2008 (SC'08), Austin, TX, USA, 15–21 November, pp. 22:1–12.
Ramakrishnan, L., Jackson, K.R., Canon, S., Cholia, S., Shalf, J., 2010. Defining future platform requirements for e-Science clouds. In: Proceedings of the 1st ACM symposium on Cloud computing (SoCC), 10-11 June, Indianapolis, IN, USA, pp. 101-106.
Samanta, A., Ganguly, S., Myneni, R.B., 2011. MODIS Enhanced Vegetation Index data do not show greening of Amazon forests during the 2005 drought. New Phytologist 189 (1), 11–15.
Schäffer, B., Baranski, B., Foerster, T., 2010. Towards Spatial Data Infrastructures in the Clouds. In: Painho, M., Santos, M.Y., Pundt, H. (Eds.), Geospatial Thinking. Springer, Berlin, pp. 399–418.
Simonett, D.S., 1969. Editor's preface. Remote Sensing of Environment, 1(1):v.
Teutsch, C., Trostmann, E., Berndt, D., 2011. A parallel point cloud clustering algorithm for subset segmentation and outlier detection. In: Proceedings of SPIE Volume 8085 Videometrics, Range Imaging, and Applications XI, Munich, Germany, 23 May, (doi:10.1117/12.888654).
Tomlin, D., 1990. Geographic Information Systems and Cartographic Modeling. Prentice-Hall, Englewood Cliffs, NY, USA, pp. 249.
W3C, 2007. Web Services Description Language (WSDL) Version 2.0. W3C Recommendation. http://www.w3.org/TR/wsdl20/(accessed 12.11.12).
Xiaoshu, S., Hong, Z., 2010. High Performance Remote Sensing Image Processing Using CUDA. In: Proceedings of the 3rd International Symposium on Electronic Commerce and Security (ISECS) 2010, Wuhan, China, 29–31 July, pp. 121–125.
Yang, H., Liu, X., 2012. Software Reuse in the Emerging Cloud Computing Era. IGI Global, Hershey, PA, USA, pp. 270.
Yang, C., Raskin, R., Goodchild, M., Gahegan, M., 2010. Geospatial cyberinfrastructure: past, present and future. Geospatial Cyberinfrastructure 34 (4), 264–277.
Yang, C., Goodchild, M., Huang, Q., Nebert, D., Raskin, R., Xu, Y., Bambacus, M., Fay, D., 2011. Spatial cloud computing: how can the geospatial sciences use and help shape cloud computing? International Journal of Digital Earth 4 (4), 305–329.

4. Hierarchical Profiling of Geoprocessing Services

Müller, Matthias: Hierarchical Profiling of Geoprocessing Services. Computers and Geosciences, 82, 2015, pp. 68–77.

Abstract. Analysis workflows in geoinformation systems and geodata infrastructures are built from reusable geoprocessing services. Ideally, these services are well-defined implementations which can be readily understood by clients in order to find and invoke the right service for a particular task. Despite technological advances towards service-oriented architectures, implementation uncertainty is still an issue and most geoprocessing services lack solid and meaningful descriptions of the provided functionality.

This paper reviews previous work in the field of service-oriented geoprocessing and discusses their contributions towards interoperable and well-defined processing services. Based on these findings, a framework is proposed that captures both semantic and syntactic properties of geoprocessing functions at different levels of granularity. Each of the levels is associated with a set of descriptive artifacts that refine the definitions of coarser levels, ultimately leading to well-defined implementations. The utility of the framework is illustrated for task-oriented search and workflow verification. Finally the paper discusses possible limitations of the presented approach and provides suggestions for future work.

Computers & Geosciences 82 (2015) 68–77

Contents lists available at ScienceDirect

Computers & Geosciences

journal homepage: www.elsevier.com/locate/cageo

Hierarchical profiling of geoprocessing services

Matthias Müller*

Technische Universität Dresden, Geoinformation Systems, 01062 Dresden, Germany

ARTICLE INFO

Article history:
Received 29 January 2015
Received in revised form
21 April 2015
Accepted 27 May 2015
Available online 29 May 2015

Keywords:
Geoprocessing services
Interoperability
Profiling
Subtyping

ABSTRACT

Analysis workflows in geoinformation systems and geodata infrastructures are built from reusable geoprocessing services. Ideally, these services are well-defined implementations which can be readily understood by clients in order to find and invoke the right service for a particular task. Despite technological advances towards service-oriented architectures, implementation uncertainty is still an issue and most geoprocessing services lack solid and meaningful descriptions of the provided functionality.

This paper reviews previous work in the field of service-oriented geoprocessing and discusses their contributions towards interoperable and well-defined processing services. Based on these findings, a framework is proposed that captures both semantic and syntactic properties of geoprocessing functions at different levels of granularity. Each of the levels is associated with a set of descriptive artifacts that refine the definitions of coarser levels, ultimately leading to well-defined implementations. The utility of the framework is illustrated for task-oriented search and workflow verification. Finally the paper discusses possible limitations of the presented approach and provides suggestions for future work.

1. Introduction

Reuse, comparability, and task-oriented discovery of geoprocessing services and functions are persisting topics in geoinformation science and application (Anselin, 2012; Lutz et al., 2003; Müller et al., 2013; Yuan and Albrecht, 1995). Instead of having well-defined implementations at hand, users still face a range of software products and libraries that maintain their own set of tools with system-specific documentation and interfaces. Web services for geoprocessing such as the Web Processing Service (WPS; OGC, 2007) or vendor-driven services (e.g. ESRI's ArcGIS Server REST API) have been introduced to overcome these incompatibilities. They eliminate the need to install a range of GIS products and geospatial libraries at the client's computer in order to access a particular geoprocessing tool. However, occasional surveys reveal little progress towards well-defined and interoperable functionality. To understand what a particular geoprocessing Web service actually does to their data, clients must obtain background knowledge about the service's backend systems and libraries (Granell et al., 2010). This contradicts a fundamental assumption in service-oriented design. The source system that runs a service is expected to be hidden from the client. In this sense, Web services have hardly improved semantic interoperability for geoprocessing applications but made existing deficits more obvious. In general, the interoperability of geoprocessing services is still limited to the technical level (Friis-Christensen et al., 2007), i.e. the communication protocol, network infrastructure and data encoding.

Geoprocessing services in general lack the ability to express and compare the semantics of the provided functions and algorithms. At best, implementations of geoprocessing functions are provided with comprehensive interface documentation, algorithmic specifications, detailed behavioral description, supported data exchange formats and permitted data volumes. At worst none of this information is present. For apparently identical implementations of common algorithms occasional case studies report slightly and sometimes seriously diverging behavior (Fisher, 2006; Lutz et al., 2003). A stricter focus on well-defined implementations and semantically enhanced interfaces, which can reliably distinguish identical from differing implementations, would help to avoid such implementation uncertainty.

The following section reviews current international standards and scientific publications on meaningful descriptions for geoprocessing functions and services. Subsequently, four levels of granularity are identified at which meaningful descriptions for geoprocessing functions can be obtained. These levels are arranged in a hierarchical framework with clear inheritance structures. Each level in the framework is associated with a set of descriptive artifacts that characterize geoprocessing services with increasing detail. A profiling process compiles these artifacts in a strictly additive manner. The process finally delivers well-defined

* Fax: +49 351 463 35879.
E-mail address: Matthias_Mueller@tu-dresden.de

http://dx.doi.org/10.1016/j.cageo.2015.05.017
0098-3004/

implementations that are consistent with declarations and behavior specifications imposed by superior levels. The described framework is shown to be useful for search and retrieval tasks, service cataloging, service comparison, and service invocation. Further examples describe possible applications in provenance tracking and workflow verification.

2. Related work

Meaningful descriptions for geoprocessing functions have been approached from different directions, including interface standards for Web services, metadata standards, ontologies, mathematical formalizations, and free-text descriptions. For subsequent discussions, some important terms need to be established. *Geoprocessing* refers to the processing of spatial or geo-referenced data which relate to conceptual data models and encoding formats from geoscience and neighboring disciplines. In reference to previous works on the topic (Albrecht, 1996; Fitzner et al., 2011; Lutz et al., 2003) a *geoprocessing function* is defined as a stateless function that transforms a given set of input data into a set of output data. The data types of these inputs and outputs usually relate data models and schemas in geospatial applications and neighboring disciplines. With regard to the basic principles of service orientation (Erl et al., 2008; Kuhn, 2005) a *geoprocessing service* is defined as an implementation of a well-specified, interoperable, and reusable geoprocessing function that receives and generates data in a common encoding format and can be readily embedded into a workflow. A broad range of geographic analysis tasks may be solved by assembling arbitrarily complex workflows from a finite pool of geoprocessing services (de Jesus et al., 2012; Longley et al., 2011). In contrast, *geoprocessing Web services* are a technology that provides client–server interaction for performing geoprocessing tasks on the Internet.

2.1. Normative definitions of geospatial operations

A straight-forward path towards well-defined geoprocessing services is a standardization process which declares clear conformance criteria for compliant implementations. In geoinformatics this approach has been chosen in the Simple Features for SQL Specification (OGC, 1999) and its ISO counterpart (ISO, 2004). Both standards are grounded in the work of Egenhofer and others (e.g. Egenhofer and Herring, 1992) and describe seven so-called "methods for spatial analysis", like Buffer, ConvexHull or geometrical and topological overlay operations. Compliant implementations of these specifications can be found in various data base products, such as Postgres/PostGIS, Oracle Spatial or Microsoft SQL Server. While the original Simple Features specification had targeted data base implementations, the standardized operations also appear in geospatial libraries, such as the Java Topology Suite (JTS) and the Geometry Engine Open Source (GEOS), or as query predicates in geospatial data services (ISO, 2010). This is an ideal situation from user's perspective: No matter which implementation is chosen, the adherence to a common standard guarantees consistent behavior across all software systems. Although normative definitions automatically lead to well-defined implementations they have only been applied to a tiny number of geoprocessing functions.

2.2. Profiling of Web processing services

The widely used WPS standard (OGC, 2007) describes a service interface and communication protocol for processing data in a Web service environment. The specification is quite generic and WPS clients and servers provide and invoke a broad range of geoprocessing functions on the Web. Geoprocessing functions in WPS are specified by so-called process descriptions which roughly resemble method signatures in programming languages. Process descriptions cover syntactic properties of a function's arguments (inputs and outputs), the supported data schemas, and encoding formats. Simple human-readable free-text elements for titles and abstracts may be used to supply basic semantic descriptions about the provided functionality.

In order to support typical publish–find–bind scenarios, process profiles have been proposed as a means to contract and possibly standardize the content of WPS servers. Profiles and compliant services might be cataloged in public Web based repositories and potential clients would simply browse these repositories to find processing services that implement the desired functionality. Since the WPS 1.0 standard is quite generic about the profiling process, different interpretations have evolved over time. These can be roughly divided into three categories.

Process-level application profiles have been proposed as interface blueprints for process implementations (e.g. Lanig and Zipf, 2010; Ostländer, 2008). In essence, this kind of profile contracts the process definition, process inputs and outputs and their concrete data encodings formats. A globally unique identifier such as a Uniform Resource Name (URN) or Uniform Resource Locator (URL) serves as a common reference for particular profile. In this regard, each process-level application profile exists in complete isolation.

Based on the insight that many implementations from a family of similar algorithms share common properties, *structured profiles* introduce an inheritance mechanism between individual profiles. Commonalities between similar functions, such as identical input and output parameters or common data formats are collected in base classes that can be extended to obtain more specific profiles (Nash, 2008). In contrast to process-level application profiles, structured profiles recognize syntactic and semantic similarities between geoprocessing functions.

The third category regards profiles as *compilations of processing functions*. For instance, Walenciak and Zipf (2010) and Müller et al. (2010) propose grouping specialized processing functions into thematic or domain-specific collections. These compilations share some aspects with thematic GIS toolboxes, e.g. for hydrology, terrain analysis, or spatial multi-criteria evaluation. If a WPS server declares support for a certain thematic collection of geoprocessing functions it is turned into a domain-specific computing service. Compared to structured profiles, compilations of processing functions are less driven by structural similarity at an algorithmic or syntactic level. They rather recognize similarities between geoprocessing functions from the perspective of a particular application domain. While the creation of thematic collections is extremely useful for specialized functions that clearly belong to a single domain, the approach provides little help for structuring and comparing more general functions which are used across multiple domains.

2.3. Geospatial catalogs and metadata

The ISO 19115-1 metadata standard (ISO, 2014) is the current baseline for cataloging geospatial data. Its extension ISO 19115-2 (ISO, 2009) provides additional lineage elements that describe the provenance of a particular data set, including the processing steps that have been applied to the original data. Although this lineage structure generally improves transparency for data-intensive research (Bernard et al., 2013), its *Process Step* element is a bad substitute for interface descriptions of geoprocessing services. A Process Step may provide documentation about a particular software package – or link to a documentation resource for the applied algorithm – but does not model input and output elements like, for instance, WPS. Hence the ISO19115 standard series is of

little use for cataloging geoprocessing services.

Based on an early draft of ISO 19115, Hill et al. (2001) attempted to develop a content standard for computational models, e.g. in the form of executable software packages or computer code. Next to written publications and archived data, these models should be archived by digital libraries where they can be found, downloaded, and hence reused. In contrast to ISO 19115-2 the proposed content standard is more interface-oriented and characterizes syntactic and semantic properties of computational models top-down at different levels of abstraction, i.e. conceptual, symbolic, algorithmic, and coding representations. It is noteworthy that this work anticipated some characteristics of WPS process descriptions which appeared several years later. The metadata schema is aware of input and output elements, including the ability to specify the type and format of input and output data.

2.4. Formal descriptions and behavioral semantics

Semantic interoperability problems in GIS and geographic Web services have been a topic of extensive study in the literature (Albrecht, 1996; Bishr, 1997; Kuhn, 2005; Lutz et al., 2003; Scheider et al., 2009; Yue et al., 2007). Two basic directions of investigation can be distinguished here: (1) Semantic descriptions of geoprocessing functions, and (2) semantic mappings that associate real-world processes and phenomena with these functions. For this paper approaches in the first category are of foremost interest.

An early attempt to formalize GIS operations for environmental modelling has been made by Albrecht (1996). His work defines the interfaces of twenty basic geoprocessing functions in the functional programming language Haskell. This early work focused on syntactic properties of geoprocessing functions, i.e. function signatures. Albrecht did not consider implementation uncertainty but rather assumed sufficient documentation in existing literature which could be derived from the functions' names and their arguments.

Implementation uncertainty in conjunction with geoinformation Web services has been investigated by Lutz et al. (2003). The authors provide a list of heterogeneities that hinder the exchange and use of geoprocessing services in spatial data infrastructures. *Cognitive heterogeneity* arises from the use of homonyms for processing functions or data items. For instance, topological relations between spatial objects are defined in different taxonomies (Clementini et al., 1993; Cohn et al., 1997; Egenhofer and Herring, 1992) and at different levels of granularity (Grigni et al., 1995), resulting in various system specific implementations that provide seemingly identical functions but produce slightly different results. The authors provide empirical examples for the topological operators "meet" and "touch" which behave differently in the investigated systems. *Naming heterogeneity* is introduced as the dual of cognitive heterogeneity and refers to a situation where identical items (processing functions or their arguments) share the same semantics and behavior but are named differently. Subsequent work by Probst and Lutz (2004) acknowledges *type heterogeneity* as a third interoperability issue. It could be paraphrased as cognitive heterogeneity for geographic data models and refers to the use of different conceptual data models for the same entities. A typical example is the object versus field debate (Couclelis, 1992; Kuhn, 2012; Liu et al., 2008) that contrasts the two opposing modelling approaches for geographic phenomena. At a technical level, type heterogeneity may also occur due to the multitude of data encodings spatial data and data schemas (e.g. in XML or GML). Formal specifications of geoprocessing functions have been suggested to resolve issues that arise from the use of heterogeneous concepts, names, and data types.

Function subtyping and Hoare Logic (HL; Hoare, 1969) have been applied by Lutz (2007) to describe input and output types as well as the behavior of a processing function. HL is a rigorous approach for verifying the correctness of imperative computer programs at the source code level. It assumes that the operation under test can be fully specified and verified in a formal system by providing explicit assertions, pre-conditions, and post-conditions. Lutz provides a proof-of-concept for a set of simple distance calculation functions in different types of coordinate systems. To assess the suitability of the approach for geoprocessing functions of higher complexity he suggests further investigations. However, progress on in-depth formalization of ordinary geoprocessing functions has not happened since then. Some early work by Clarke (1979) has shown general limitations of HL, suggesting that this is not feasible or at least most impracticable to provide and maintain complex formal representations for larger programs with complex control structures.

Fitzner et al. (2011) suggest declarative descriptions for geoprocessing functions to allow for automated workflow composition. Logic programming is applied to the problem in order to support comparison and matchmaking tasks for different geoprocessing functions. An example illustrates the matchmaking process for some common overlay operations. In contrast to Lutz's approach, Fitzner does not attempt to specify the general mechanics or the behavior of a particular processing function but rather assumes it is already documented in a "geospatial operation ontology" – which is still an open task. However, the provided examples illustrate the advantages of well-defined geoprocessing services for workflow composition and verification.

3. A profiling approach for geoprocessing services

The possible meanings of the term *profiling* in conjunction with geoprocessing services have already been discussed in the context of WPS. A more robust definition is provided by the Unified Modelling Language standard (UML; OMG, 2005), which describes "profiling" as a specialization mechanism for refining the semantics of a superior reference metamodel. The resulting profile is usually more constraining than the referenced metamodel but consistent with its semantics. Profiles that stem from the same metamodel are aligned to some degree and are substitutable at the metamodel's level of granularity. This approach has strong similarities to the Liskov substitution principle, i.e. that "… objects of a subtype should behave the same as those of the supertype as far as anyone or any program using the supertype can tell" (Liskov and Wing, 1994, p. 1811).

This kind of additive inheritance can also be used for the definition of geoprocessing services. Besides allowing representations of functionality at different levels of abstraction, a profiling process could also help to align future implementations of geoprocessing services. This section shows how geoprocessing functions can be profiled in a subtyping fashion by gradually refining their semantics. Four recurring levels of granularity are identified which can be arranged in a hierarchical fashion to obtain interoperable geoprocessing services that are no longer tied to the mechanics of a particular legacy system.

3.1. Concepts

Well-known geoprocessing functions stem from well-known concepts (Hill et al., 2001). Normative specifications such as OGC (1999) implicitly use concepts when they define the generic behavior of an operation. Formal specifications reference concepts to define the semantics of an operation and its arguments (Kuhn, 2005). Familiar examples of commonly used functions are geometric intersection and overlay operators, all kinds of buffering functions, map algebra operations (Tomlin, 1990), convolution

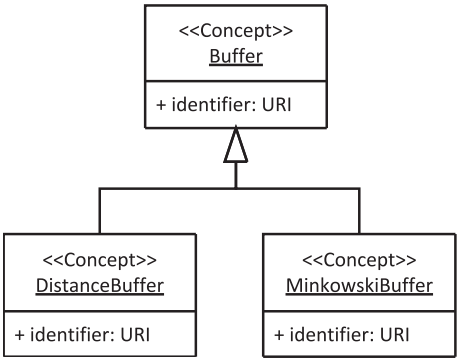


Fig. 1. Concept taxonomy for Buffers.

filters, Viewshed and watershed computations, slope operators, distance transformations, and many others.

Concepts are documentation resources with unique identity. They may describe the functionality performed by a particular process or GIS tool or describe the result of a modelling or computing activity, e.g. a "three kilometer Buffer around New York City". Buffer definitions, for instance, may be grounded in different concepts (Fig. 1). The Distance Buffer is probably the most common notion of a Buffer which represents all points whose distance from a given geometric object is less than or equal to a given distance (de Smith et al., 2015; ISO, 2004). An alternative subconcept for Buffer uses Minkowski Sums in conjunction with a reference shape instead of a distance-based definition (Zhang et al., 2010).

Another exemplary taxonomy is shown in Fig. 2, which classifies different Viewshed concepts by their outputs. The classification has been adopted from Floriani and Magillo (2003) who give an overview of Viewshed algorithms that operate on regular square grids and triangulated irregular networks (TINs).

A more complex taxonomy is required for some generalization functions. Optimal simplification algorithms for polygonal paths have been extensively studied in literature and are frequently applied in geospatial applications. A polygonal path is defined as a totally ordered set of vertices $P=(p_1, p_2,\ldots,p_n)$ and a set of path segments $S = (s_1 = p_1\bar{p}_2,\ s_2 = p_2\bar{p}_3,\ \ldots,\ s_{n-1} = p_{n-1}\bar{p}_n)$. A simpler version P' of P is obtained by removing some of the vertices v. This results in a reduced number of path segments $S' = (s'_1 = p'_1\bar{p}'_2,\ s'_2 = p'_2\bar{p}'_3,\ \ldots,\ s_{n-1} = p_{k-1}\bar{p}_k)$ with $2 \le k \le n-1$.

Two broad classes of optimal simplification algorithms can be distinguished (Bose et al., 2006; Imai and Iri, 1988). $min\#(\varepsilon)$ algorithms minimize the number of vertices according to some error criterion ε, such as displacement, total difference of the path length, or area transfers between the left and right sides of the path. $min\varepsilon(\#)$ algorithms minimize an error criterion ε for a given maximum number of vertices k.

Fig. 3 shows a taxonomy of path simplification algorithms where the class of $min\#(\varepsilon)$ algorithms is further subdivided according to different error criteria. A similar distinction can be made for the branch of $min\varepsilon(\#)$ algorithms (Bose et al., 2006; Gudmundsson et al., 2007). All simplification concepts remove unnecessary detail or noise from the original data. However, each concept applies a different set of optimization criteria. A user may take advantage of this taxonomy to get an overview of the different algorithms and decide about the correct or suitable simplification strategy for a particular task. Furthermore, a user has certainty about a software component that implements a particular sub-concept.

3.2. Generic profiles for functions and algorithms

While most authors agree that concepts precede more fine-grained definitions of processing functions, they pursue different approaches for refined specifications. Hill et al. (2001) propose that symbolic (i.e. mathematical or logical) representations precede algorithmic representations of computing models which are then turned into programmatic implementations. This is a typical approach in modeling and simulation domains, where every subcomponent of a simulation model can be expressed as a set of differential equations (Zeigler et al., 2000), but it might not fit the typical situation in GIS and environmental modelling. Many useful functions such as generalization operations, data fusion algorithms, visibility and watershed calculations are defined in an algorithmic fashion, i.e. by procedural specifications, pseudo code, or simply by stating pre- and post-conditions in written text. In other cases algorithmic and symbolic representations may be mixed, i.e. by embedding mathematical equations into pseudo code statements. For instance, the path simplification algorithms discussed in Section 3.1 are specified in free-text which provides a detailed description of the simplification procedure, states pre- and post-conditions and include explanatory images, mathematical statements, and pseudo code. This kind of documentation is typical for most geoprocessing functions and largely comprehensible for humans. Fully formalized approaches that provide abstract function interfaces with explicit typing and formal behavioral specification have been shown to work for the simplest geospatial operations in controlled environments.

Based on these findings generic profiles are scoped as semi-formal and encoding-independent specifications, which contract the most basic properties of a function's interface and document the expectable behavior. While concepts are output-centric, i.e. describe *what* is produced by a particular geoprocessing service, generic profiles detail *how* the outputs are derived from the inputs. They extend concepts with (1) abstract signatures, that provide compulsory names and cardinalities of inputs and outputs to avoid

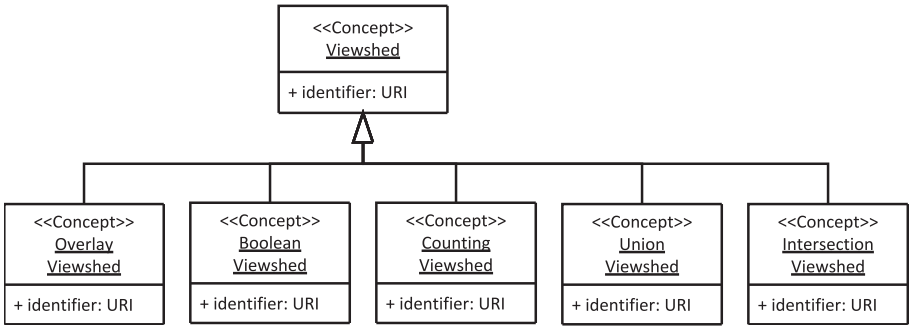


Fig. 2. Concept taxonomy for Viewsheds.
Source: Floriani and Magillo (2003).

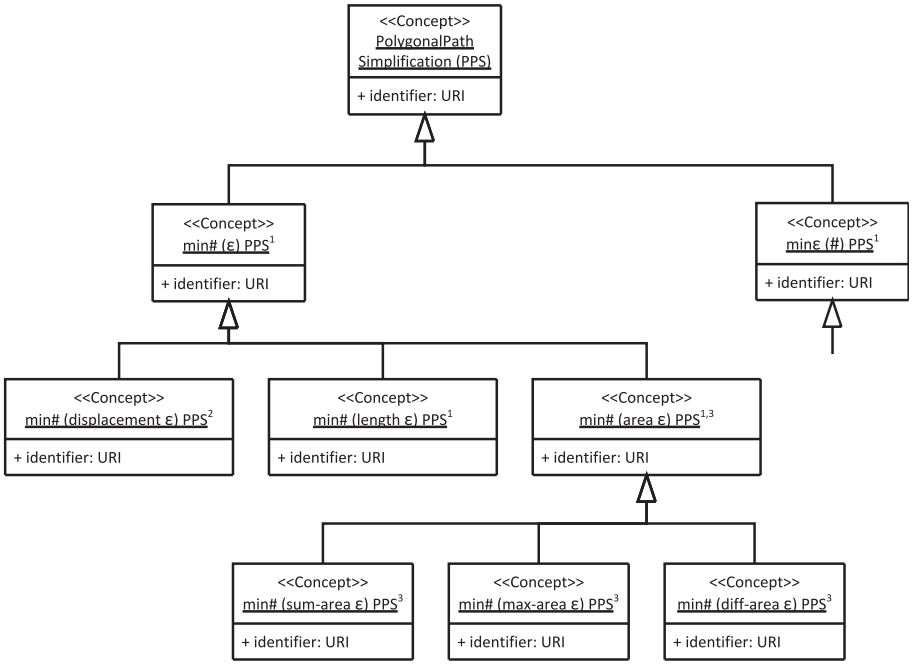


Fig. 3. Concepts taxonomy for polygonal path simplification. The listed concepts and compliant algorithms are discussed by [1]Imai and Iri (1988); [2]Gudmundsson et al. (2007); [3]Bose et al. (2006).

naming ambiguities at the implementation level. Similar to normative definitions for geoprocessing functions they provide (2) a rigorous description of the expectable behavior, the meaning of function arguments, and pre-and post-conditions for execution – but not necessarily in a fully formalized way.

The generic Buffer interfaces in Fig. 4 show two distinct realizations of a Distance Buffer concept. Euclidean Buffers are provided by most GIS and intended for use with projected length-preserving coordinate reference systems (CRS). Geodesic Buffers recently appeared in GIS and spatial databases using geodesic distance computation on an ellipsoid, i.e. the shortest point-to-point distance on an ellipsoid's surface. The latter Buffer concept is frequently used in conjunction with geographic CRSes and global data sets, in which case the use of a Euclidean Buffer is rather meaningless. While both Buffer profiles refer to the same concept and have identical signatures (except for their names), they behave quite differently. The Euclidean Buffer uses linear interpolation and performs distance calculations in the spatial reference system of the geometric object. The Geodesic Buffer uses geodesic interpolation and performs distance calculations on the spatial reference system's ellipsoid.

Computational precision is another important aspect that may be covered in a generic profile. The *Euclidean Distance Buffer* is

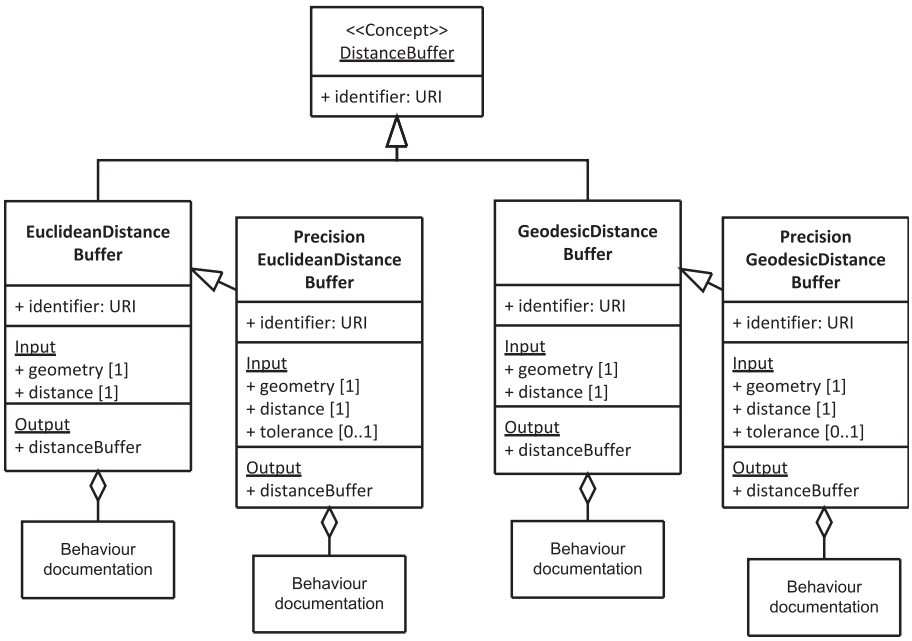


Fig. 4. Generic profiles for Buffer functions.

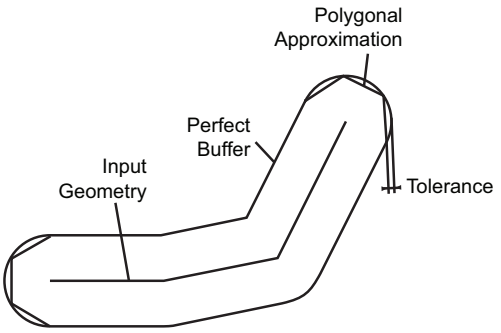


Fig. 5. Polygonal Buffer approximation.

similar to the Buffer operation defined in OGC (1999). Although the interface seems quite complete at first glance, the geometric precision of the result is virtually undefined. The precision aware Buffer functions make precision explicit by providing an additional tolerance parameter that controls the geometric precision of the result. Imprecision in Buffer computation may have two causes. First, it is be caused by approximate distance calculations. Second, imprecision may be introduced by the geometry model since most Buffer implementations do not support circular arcs but approximate curved segments by piecewise linear functions (Fig. 5). Both cases can be covered by an additional tolerance parameter that guarantees an approximation error lower or equal to the specified tolerance. Use cases that require negligible precision may stick to the general Distance Buffer variants while more demanding applications would invoke a service that implements a Buffer function with precision guarantees.

For other types of functions, precision uncertainty may be an inherent property of the algorithm and cannot be controlled by a tolerance parameter. Simplification algorithms for polygonal paths (Section 3.1) are computationally demanding. Next to exact algorithms, heuristics have been developed which run faster by orders of magnitude but only approximate the exact solution. Heuristics still comply with the overall optimization strategy imposed from the conceptual level, but behave differently from their exact counterparts. In such cases, exact and heuristic variants must be expressed in distinct generic profiles.

3.3. Implementation profiles

Generic profiles have been introduced to resolve conceptual and naming heterogeneity for geoprocessing functions. Software implementations of generic interfaces must comply with their syntax and behavioral contract. Due to their abstract nature, generic interfaces cannot be immediately used for execution. The typing of interface elements is purely conceptual and lacks the specification of a technical data exchange format. Without such a format, a potential client would not know how to send or receive data to/from a concrete geoprocessing service.

Implementation profiles extend generic profiles with non-functional properties. They inherit the behavioral and syntactic properties from concepts and generic interfaces. In addition they supply encoding formats that encode the conceptual data types in common data exchange formats such as XML/GML, JSON/GeoJSON, NetCDF, GeoTiff, or other well-known data exchange formats.

In Fig. 6 the generic interface of the Simple Buffer is extended by to implementation profiles, one for GeoJson and one for GML. The distance value has to be encoded with Double precision.

Due to limited computing resources implementation profiles may further apply reasonable size restrictions to input datasets. For instance, the generic interface of a Mosaic function, which merges an arbitrary number of raster tiles into a single raster, may theoretically accept an unlimited number of input tiles of any data volume. Implementations will certainly restrict this generic Mosaic function by limiting the number of tiles and their permitted maximum size to a manageable amount.

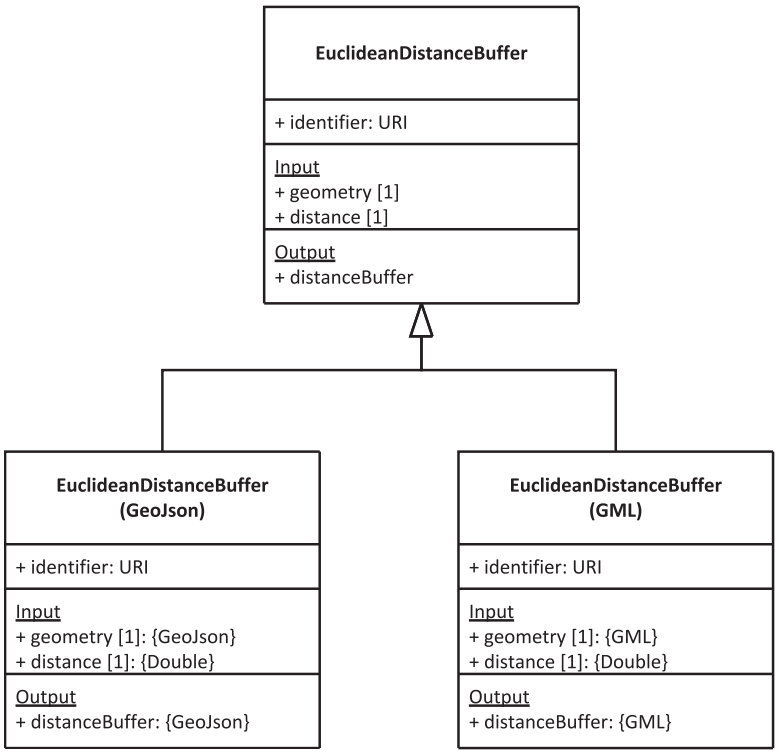


Fig. 6. Implementation profiles for the Euclidean Distance Buffer.

The process description schema of WPS, in terms of an interface blueprint (Section 2.2), is a possible encoding of an implementation profile. A function's interface is not expressed in a high-level fashion but already specifies concrete data formats and non-functional aspects, such as applicable data formats and data size limitations.

3.4. Service implementations

The development of software components that comply with the implementation profiles follows service-oriented paradigms or contract-driven service design (Erl et al., 2008; Meyer, 1992; Szyperski et al., 2002). In brief, implementations are allowed to extend an existing contract, i.e. the interface and behavior of an implementation profile, but shall not to violate it.

A straight forward implementation simply realizes a given implementation profile. However it might be desirable to provide an extended implementation for several reasons. The most obvious case is the support for multiple data exchange formats (Fig. 7A).

Another extension case is shown in Fig. 7(B) which provides an additional optional parameter for cap styles referring to similar buffer arguments in JTS and ArcGIS. Cap styles are occasionally used in conjunction with line buffering, allowing fine-grained control of the Buffer's shape at the start and end vertices. The addition of the optional parameter is fully backwards compatible. If unused or ignored by the client, it does not alter the original contract of the Euclidean Buffer for GeoJson implementation profile. If a client is aware of the optional parameter, he may use it to trigger an alternate behavior of Buffer implementation (B). Implementers would consider this kind of extension if they wish to provide enhanced capabilities, which go beyond the profiled behavior, to some clients but still need to maximize compatibility with the original contract.

3.5. Process catalogs and lineage records

Current implementations of geospatial data infrastructures have a strong focus on mapping and data access. Catalog interfaces and metadata schemas have been designed to support data and map retrieval. Inventories or catalogs for processing services are still rare and uncommon since search and retrieval tasks demand meaningful descriptions of geoprocessing functions which are not yet in place.

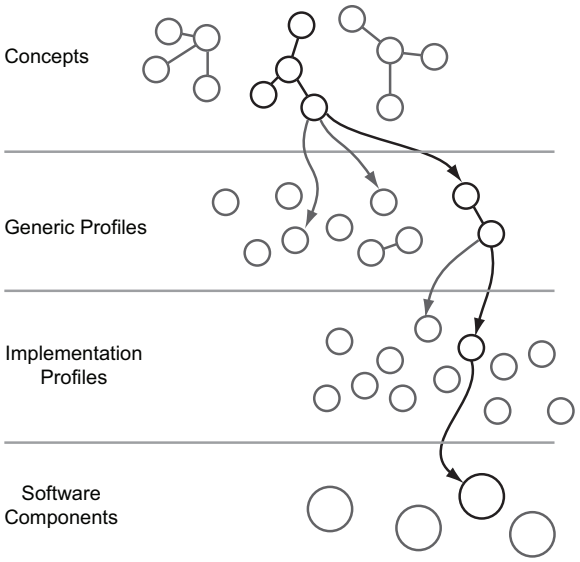


Fig. 8. Inheritance graph for profiles and software components.

The proposed profiling approach defines four granularity levels at which a processing function may be described. The distinction of input and output ports for data exchange provide compatibility with generic models for workflows, components, Web services, in particular with the process semantics of WPS (OGC, 2007). These profiles may be used to support the search for software components that implement a particular concept or profile (Fig. 8). Here, a user would start searching (or browsing) for the required functionality at the conceptual level and then follow subsequent links to refined generic profiles to finally obtain a compliant software implementation. Similarly, a given software component (i.e. a library component or online processing service) can be assessed in terms of provided functionality at different levels of abstraction, i.e. the compliance with a required generic profile or concept, in order to assess its applicability for a particular task.

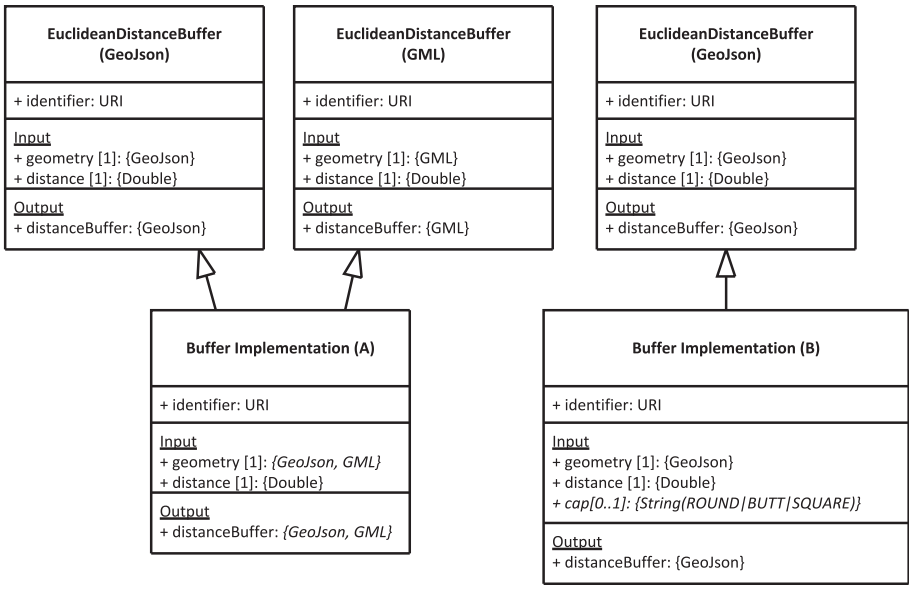


Fig. 7. Buffer implementations with (A) support for multiple data exchange formats and (B) an optional parameter for cap styles.

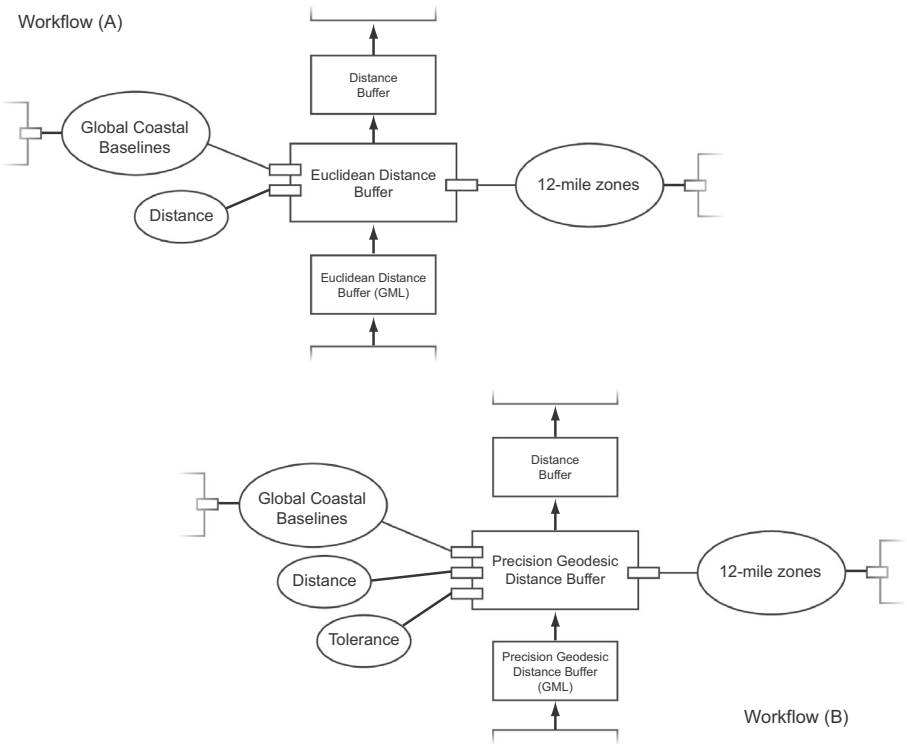


Fig. 9. Excerpt workflows and provenance graphs for territorial waters computation.

In addition, the profiling approach supports the creation of provenance records. Current metadata schemas such as ISO 19115-2 are cumbersome to use and still do not provide sufficient detail about the processing workflow (Henzen et al., 2013). With profile references embedded in provenance records, the whole workflow becomes transparent, right down to the behavior of the invoked processing functions. Fig. 9 shows an excerpt from two workflows that compute the territorial waters from global costal baselines. To assess the correctness of the result, detailed information about the employed proximity calculation is desirable. The result of workflow (A) has to be treated with caution and is probably wrong since the invoked buffering operation employs a Euclidean Buffer on a global dataset and in this case the result is probably highly inaccurate. The result of Workflow (B) can certainly be trusted since the invoked Buffer function is capable of performing geodesic computations and provides precision guarantees through a tolerance parameter. For absolute certainty, the values of *distance* and *tolerance* might be checked.

As hinted in Fig. 9, coarser representations of the workflow might be generated from the graph for general overviews, e.g. at the level of concepts. Assuming that fine-grained provenance has been recorded, a deep inspection may be performed by portraying lower levels possibly down to the particular implementation. Even if at some point in the future the invoked software implementation no longer exists, the workflow is still reproducible and verifiable based on the contracts provided by the profile definitions.

4. Conclusion and future work

This paper has presented a semi-formal approach for profiling interoperable geoprocessing services. In contrast to fully formalized specifications, the correctness of derived implementations cannot be automatically checked. A major advantage of the proposed approach is its applicability for higher-level algorithms and functions which cannot be expressed with formal logic. The reviewed literature provides sufficient evidence that this is the case for all but the simplest spatial operations.

The presented examples use existing taxonomies for geoprocessing functions and geo-spatial operators. Representations and interpretations of concepts and profiles are subject to scientific discourse and may evolve over time. There is nothing wrong with this fact; it is merely a consequence of GIS evolution and a better understanding of GIS components (Kuhn, 2005). Steady development of algorithms and methodology may lead to improved taxonomies for concepts or generic profiles. The examples provided in this paper are derived from current state-of-the-art in GIS research and application to illustrate the purpose and capabilities of the framework. There is no definite set of criteria that could indicate whether a contract, expressed by a concept or generic profile, is sufficient, conclusive, and comprehensive.

This issue is not unique to the geospatial domain and has been discussed at a broader scope for service-oriented software design in general. Szyperski et al. (2002) have drawn an analogy between semiformal contracts in software component design and real-world law texts which are regularly debated in courts. Both are subject to evolution and existing contracts and taxonomies might be rephrased, refined or even reconsidered. To organize this development process in a service-driven environment, Erl et al. (2008) have suggested versioned contracts, where subsequent revisions provide refined versions of the original contract. All elements in the profiling hierarchy are scoped as identifiable resources and would thus support this kind of versioning.

The descriptions of contracts, generic profiles, implementation profiles, and finally implementations are scoped as persistent documentation resources which require a suitable encoding for both syntax and behavior. As hinted by the figures the signatures and their elements may be encoded in a well-known modelling or markup language such as UML or XML. The upcoming revision of the Web Processing Service specification (OGC, 2015, Sections

7 and 8) will support hierarchical profiling. It provides XML encodings for generic and implementation profiles to build function taxonomies for well-known geoprocessing functions and future algorithms. Additional metadata elements can be used by service implementers to express compliance with objects inside these taxonomies, alleviating the burden of service documentation.

Sematic Web languages, such as RDF, are another option which is particularly suitable for encoding the semantic links within the sets of concepts, generic profiles and implementation interfaces (Fig. 8). In contrast, the specification of behavior requires additional documentation resources. In both industry standards and academia classical text-based documentations are primarily used for this task (Albrecht, 1996; OGC, 1999). Thus human-readable hypertext media and specification texts are viable documentation resources (e.g. ESRI, 2015). These resources might also be scoped as educative material, i.e. be supplied with examples and guidance to help non-specialists understanding domain-specific concepts and algorithms. The testing for correctly implemented behavior is still subject to classical software testing and involves human judgment and expertise.

The presented profiling approach bears strong analogies with the development process for data schemas, which are typically derived from conceptual models and then refined and extended for different application domains (Toth et al., 2012). A similar inheritance mechanism is applied to derive generic profiles, implementation profiles, and finally implementations from their ancestors. In general, pure extension-based approaches result in large numbers of subclasses which need to be managed and maintained. This might indicate a scalability issue of hierarchical profiling for large numbers of heterogeneous geoprocessing functions with diverse subtypes. Semi- or unstructured collections of processing functions, which are generally hard to profile, might be easier to handle with linked data representations as proposed by Granell et al. (2013). For geoprocessing functions, there is some evidence that they can be characterized according to common aspects, such as the definition of distance metrics and space, geometric precision, and the spatial or spatio-temporal reference system types. Future research should investigate these common aspects to possibly profile geoprocessing functions according to some globally shared properties.

Acknowledgments

The research leading to these results has received funding from the German Federal Ministry of Education and Research under Grant agreement no. 01LL0901C.

References

Albrecht, J., 1996. Universal Analytical GIS Operations [dissertation]. Universität Vechta.

Anselin, L., 2012. From SpaceStat to CyberGIS. Int. Reg. Sci. Rev. 35 (2), 131–157.

Bernard, L., Mäs, S., Müller, M., Henzen, C., Brauner, J., 2013. Scientific geodata infrastructures: challenges, approaches and directions. Int. J. Digit. Earth 7 (7), 613–633.

Bishr, Y., 1997. Semantic Aspects of Interoperable GIS [dissertation]. Landwirtschaftliche Universität, Wageningen, Enschede.

Bose, P., Cabello, S., Cheong, O., Gudmundsson, J., van Kreveld, M., Speckmann, B., 2006. Area-preserving approximations of polygonal paths. J. Discrete Algorithm. 4 (4), 554–566.

Clarke Jr., E.M., 1979. Programming language constructs for which it is impossible to obtain good hoare axiom systems. J. ACM 26 (1), 129–147.

Clementini, E., Felice, P.D., Oosterom, P.v., 1993. A small set of formal topological relationships suitable for end–user interaction. In: Proceedings of the Third International Symposium on Advances in Spatial Databases, pp. 277–295.

Cohn, A.G., Bennett, B., Gooday, J., Gotts, N.M., 1997. Qualitative spatial representation and reasoning with the region connection calculus. Geoinformatica 1 (3), 275–316.

Couclelis, H., 1992. People manipulate objects (but cultivate fields): beyond the raster-vector debate in GIS. In: Frank, A., Campari, I., Formentini, U. (Eds.), Proceedings of the International Conference GIS-From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space, pp. 65–77.

de Jesus, J., Walker, P., Grant, M., Groom, S., 2012. WPS orchestration using the Taverna workbench: The eScience approach. Comput. Geosci. 47, 75–86.

de Smith, M.J., Goodchild, M.F., Longley, P.A., 2015. Geospatial Analysis—A Comprehensive Guide to Principles, Techniques and Software Tools, 5th ed. Winchelsea Press, Winchelsea, UK.

Egenhofer, M.J., Herring, J.R., 1992. Categorizing Binary Topological Relations Between Regions, Lines, and Points in Geographic Databases. Department of Surveying Engineering, University of Maine.

Erl, T., Karmarkar, A., Walmsley, P., Haas, H., Yalcinalp, L.U., Liu, C.K., Orchard, D., Tost, A., Pasley, J., 2008. Web Service Contract Design and Versioning for SOA. Prentice Hall, Upper Saddle River, NJ.

ESRI, 2015. Geoprocessing Tool References. ⟨http://desktop.arcgis.com/en/desktop/latest/tools⟩ (accessed 01.04.15).

Fisher, P., 2006. Algorithm and implementation uncertainty: any advances?. In: Fisher, P. (Ed.), Classics from IJGIS: Twenty years of the International Journal of Geographical Information Science and Systems. CRC Press, Boca Raton, pp. 225–228.

Fitzner, D., Hoffmann, J., Klien, E., 2011. Functional description of geoprocessing services as conjunctive datalog queries. Geoinformatica 15 (1), 191–221.

Floriani, L.D., Magillo, P., 2003. Algorithms for visibility computation on terrains: a survey. Environ. Plan. B: Plan. Des. 30 (5), 709–728.

Friis-Christensen, A., Ostländer, N., Lutz, M., Bernard, L., 2007. Designing service architectures for distributed geoprocessing: challenges and future directions. Trans. GIS 11 (6), 799–818.

Granell, C., Díaz, L., Gould, M., 2010. Service-oriented applications for environmental models: reusable geospatial services. Environ. Model. Softw. 25 (2), 182–198.

Granell, C., Díaz, L., Schade, S., Ostländer, N., Huerta, J., 2013. Enhancing integrated environmental modelling by designing resource-oriented interfaces. Environ. Model. Softw. 39, 229–246.

Grigni, M., Papadias, D., Papadimitriou, C., 1995. Topological inference. In: Mellish, C.S. (Ed.), Proceedings of the 14th International Joint Conference on Artificial Intelligence, pp. 901–906.

Gudmundsson, J., Narasimhan, G., Smid, M., 2007. Distance-preserving approximations of polygonal paths. Comput. Geom. 36 (3), 183–196.

Henzen, C., Mäs, S., Bernard, L., 2013. Provenance information in geodata infrastructures. In: Vandenbroucke, D., Bucher, B., Crompvoets, J. (Eds.), Geographic Information Science at the Heart of Europe. Springer International Publishing, pp. 133–151.

Hill, L.L., Crosier, S.J., Smith, T.R., Goodchild, M., 2001. A Content Standard for Computational Models. d-Lib Mag. 7.

Hoare, C.A.R., 1969. An axiomatic basis for computer programming. Commun. ACM 12 (10), 576–580.

Imai, H., Iri, M., 1988. Polygonal approximations of a curve-formulations and algorithms. In: Toussaint, G.T. (Ed.), Machine Intelligence and Pattern Recognition. North-Holland, pp. 71–86.

ISO, 2004. Geographic Information-Simple Feature Access – Part 1: Common architecture. ISO 19125-1:2004.

ISO, 2009. Geographic Information – Metadata – Part 2: Extensions for Imagery and Gridded Data. ISO 19115-2:2009.

ISO, 2010. Geoinformation – Filter Encoding. ISO 19143:2010.

ISO, 2014. Geographic Information Metadata Part 1: Fundamentals ISO 19115-1:2004.

Kuhn, W., 2005. Geospatial Semantics: Why, of What, and How?. In: Spaccapietra, S., Zimányi, E. (Eds.), Journal on Data Semantics III. Springer, Berlin Heidelberg, pp. 1–24.

Kuhn, W., 2012. Core concepts of spatial information for transdisciplinary research. Int. J. Geogr. Inf. Sci. 26 (12), 2267–2276.

Lanig, S., Zipf, A., 2010. Proposal for a Web Processing Services (WPS) application profile for 3d processing analysis. In: Proceedings of the Second International Conference on Advanced Geographic Information Systems, Applications, and Services (GEOPROCESSING '10), Washington, DC, USA, pp. 117–122.

Liskov, B.H., Wing, J.M., 1994. A behavioral notion of subtyping. ACM Trans. Progr. Lang. Syst. 16 (6), 1811–1841.

Liu, Y., Goodchild, M.F., Guo, Q., Tian, Y., Wu, L., 2008. Towards a general field model and its order in GIS. Int. J. Geogr. Inf. Sci. 22 (6), 623–643.

Longley, P.A., Goodchild, M.F., Maguire, D.J., Rhind, D.W., 2011. Geographic Information Systems and Science. Wiley, Hoboken, NJ.

Lutz, M., 2007. Ontology-based descriptions for semantic discovery and composition of geoprocessing services. Geoinformatica 11 (1), 1–36.

Lutz, M., Riedemann, C., Probst, F., 2003. A classification framework for approaches to achieving semantic interoperability between GI web services. In: Kuhn, W., Worboys, M.F., Timpf, S. (Eds.), Spatial Information Theory. Foundations of Geographic Information Science. Springer, Berlin Heidelberg, pp. 186–203.

Meyer, B., 1992. Applying "Design by Contract". Computer 25 (10), 40–51.

Müller, M., Bernard, L., Kadner, D., 2013. Moving code – sharing geoprocessing logic on the web. ISPRS J. Photogramm. Remote Sens. 83, 193–203.

Müller, M., Bernard, L., Vogel, R., 2010. Multi-criteria evaluation for emergency management in spatial data infrastructures. In: Konecny, M., Zlatanova, S., Bandrova, T.L. (Eds.), Geographic Information and Cartography for Risk and Crisis Management. Springer, Berlin, Heidelberg, pp. 273–286.

Nash, E., 2008. WPS application profiles for generic and specialised processes. In: Pebesma, E., Bishr, M., Bartoschek, T. (Eds.), Proceedings of the 6th Geographic Information Days (GI-Days 2008), Münster.
OGC, 1999. OpenGIS Simple Feature Specification for SQL, Revision 1.1. OGC Document 99-049.
OGC, 2007. OpenGIS Web Processing Service, Version 1.0.0. OGC Document 05-007r7.
OGC, 2015. OGC WPS 2.0 Interface Standard. OGC Document 14-065.
OMG, 2005. Unified Modeling Language: Superstructure. Version 2.0. formal/05-07-04.
Ostländer, N., 2008. Creating Specific Decision Support Systems in Spatial Data Infrastructures. An Approach for Conceptualisation, Design and Implementation [dissertation]. Westfälische Wilhelms-Universität Münster.
Probst, F., Lutz, M., 2004. Giving meaning to GI web service descriptions. In: Proceedings of the Second International Workshop on Web Services: Modeling, Architecture and Infrastructure (WSMAI-2004), Porto, Portugal.
Scheider, S., Janowicz, K., Kuhn, W., 2009. Grounding geographic categories in the meaningful environment. In: Hornsby, K.S., Claramunt, C., Denis, M., Ligozat, G. (Eds.), Spatial Information Theory. Springer, Berlin, Heidelberg, pp. 69–87.
Szyperski, C., Gruntz, D., Murer, S., 2002. Component Software. Beyond Object-Oriented Programming, second edition. Addison-Wesley, London.
Tomlin, D., 1990. Geographic Information Systems and Cartographic Modelling. Prentice-Hall, Englewood Cliffs.
Toth, K., Portele, C., Illert, A., Lutz, M., Lima, M.Nd, 2012. A Conceptual Model for Developing Interoperability Specifications in Spatial Data Infrastructures. European Commission, Joint Research Centre, Institute for Environment and Sustainability, Luxembourg.
Walenciak, G., Zipf, A., 2010. Designing a web processing service application profile for spatial analysis in business marketing. In: Painho, M., Santos, M.Y., Pundt, H. (Eds.), Proceedings of the 13th AGILE International Conference on Geographic Information Science, Guimarães, Portugal.
Yuan, M., Albrecht, J., 1995. Structural analysis of geographic information and GIS operations from a user's perspective. In: Frank, A.U., Kuhn, W. (Eds.), Spatial Information Theory A Theoretical Basis for GIS. Springer, Berlin, Heidelberg, pp. 107–122.
Yue, P., Di, L., Yang, W., Yu, G., Zhao, P., 2007. Semantics-based automatic composition of geospatial web service chains. Comput. Geosci. 33 (5), 649–665.
Zeigler, B.P., Praehofer, H., Kim, T.G., 2000. Theory of Modeling and Simulation, second edition. Academic Press, San Diego.
Zhang, P., Zhou, L., Sheng, Y., Hu, Y., 2010. A buffer generation method based on Minkowski sum. In: Proceedings of the Second International Conference on Information Science and Engineering (ICISE), Syndey, Australia, pp. 3396–3399.

5. The WPS 2.0 Interface Standard

Müller, Matthias (Ed.); Pross, Benjamin (Co-Ed.): OGC WPS 2.0 Interface Standard. Published as an International Standard by the Open Geospatial Consortium, © OGC 2015, document number 14-065.

Abstract. The OGC Web Processing Service (WPS) interface specification standardises descriptions for geoprocessing functions and defines a protocol for client–server geoprocessing (see section 1.1). While the OGC WPS standard was designed with spatial processing in mind, it can also be used to readily insert non-spatial processing tasks into a web services environment.

Contributions. The author of this thesis is the editor of the WPS 2.0 specification text and has essentially developed the information models and corresponding XML schemas. One particular focus was put on the separation of the WPS process model from the service model so that WPS process descriptions can be used as an interface description language for arbitrary geoprocessing functions, no matter if they are used in a Web service or as a more general component interface, e.g. for cataloguing the contents of GIS *toolboxes* or portable *moving code* packages.

For WPS 2.0, the Standards Working Group (SWG) has requested a clear guidance profiling WPS processes, which was not properly addressed in the first version of the standard. After several presentations of the initial work on profiling approaches (Müller 2013) and subsequent refinements, the framework on hierarchical profiling (cf. chapter 4) was approved by the SWG for inclusion in the standard and is now part of the WPS process model. Corresponding XML schemas and inheritance rules have been formalised in the standard. Sections in the specification that cover the profiling mechanism are exclusive contributions of the author. The following sections are a summary of the standard.

5.1. Specification Overview

The WPS 2.0 specification (OGC 2015a) is a major revision of the WPS 1.0 standard (OGC 2007c). The standard addresses several weaknesses of its predecessor and was redesigned with improved modularity. Figure 5.1 gives an overview of the new specification structure.

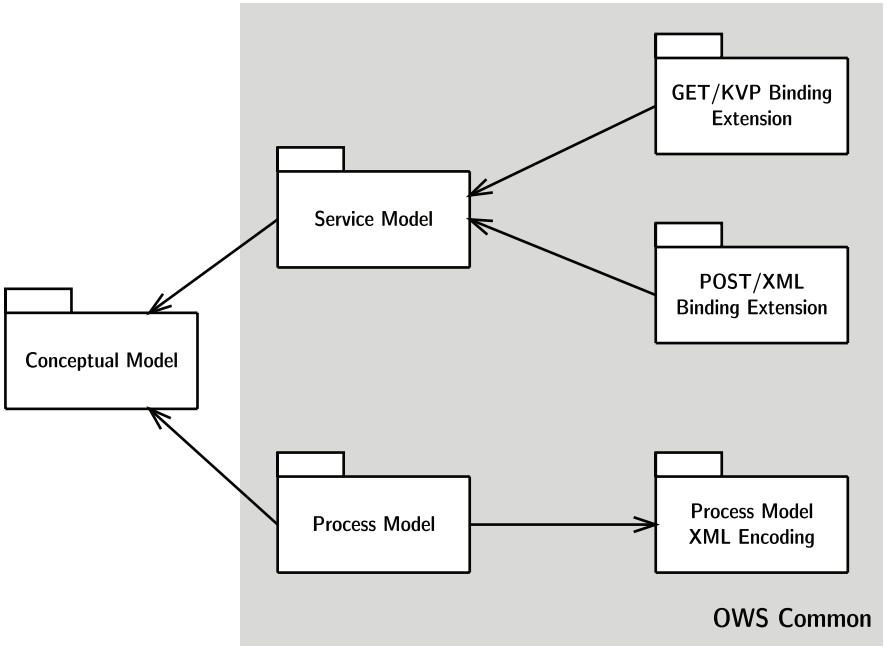


Figure 5.1.: Structure of the WPS 2.0 implementation specification (shortened)

Since the OGC abstract specifications (OGC 2015a) do not provide a common reference model or guidance for geoprocessing, the standard introduces a conceptual model for geoprocessing that is platform independent and can be implemented in various technologies and encodings. The conceptual model defines the basic properties of a process and lists basic requirements for service discovery and invocation, data transmission, and process interfaces.

Subsequent parts such as the service model and the specific process model depend on information elements defined by OWS Common 2.0 (OGC 2010). The primary encoding of the service model and process model is the Extensible Markup Language (XML). The use of the JavaScript Object Notation (JSON; ECMA 2013; IETF 2014) as an alternative encoding along with a resource-oriented service interface, based on Representational State Transfer (REST), is currently discussed and might be defined by future WPS implementation standards. The SWG on WPS 2.0 has discussed several approaches to a RESTful WPS and assumes that the current data structures can be easily adapted for that technology. It is now up to the OGC and

its members to define a common policy on JSON and REST to allow further progress towards a RESTful encoding for WPS.

In contrast to the previous version, WPS 2.0 separates the *service model* from the *process model* (outlined in section 5.3). The tight coupling between the Web service interface and the process model in WPS 1.0 prevented the use of process descriptions as a common IDL for arbitrary geoprocessing services or tools. Experiences from the work described in chapters 2 and 3, as well as a change request to support SensorML (OGC 2014c) as an alternative process interface description triggered this change and had significant impact on the new modular design. Since SensorML and WPS provide different information models for process descriptions (a comparison is provided in annex A.3), the WPS 2.0 *service model* supports both.

Other OGC standards for the Web, which have made the distinction between content encoding and service interfaces already, have served as a blueprint for this new design. For instance, the contents offered by a Web Feature Service (WFS; OGC 2014d) server depend on the ISO Feature Model (ISO 2005), and the Geography Markup Language encoding (GML; ISO 2007). Similarly, the Sensor Observation Service (SOS) relies on the information models and data encodings provided by Observations and Measurements (O&M; OGC 2011, 2013).

5.2. WPS Service Model

The service model of WPS 2.0 (cf. Figure 5.2) has been revised to provide a clearer protocol for process execution. Regarding nomenclature, the specification makes a distinction between a *process* (e.g. a geoprocessing function that can be reused in multiple executions) and a *job* representing an instance of a process with specified input and output data.

Besides the operations required for the service self-description and the description of the provided content (*GetCapabilities* and *DescribeProcess*), WPS provides an *Execute* operation for job instantiation and execution as well as *GetStatus* and *GetResult* operations for asynchronous execution. A *Dismiss* operation is provided as an optional extension which may be used by clients to signal the WPS server that they are no longer interested in a running job or its results and that the associated resources may be freed. Similarly to a cancel command in desktop GIS, this operation is useful to stop accidental execution in interactive applications or to terminate unexpectedly long running executions.

As stated in the overview section, the process model and service model in WPS 2.0 are largely decoupled. As a consequence, process descriptions can be used independently from a WPS service, i.e. to describe the interfaces of geoprocessing tools that reside in a particular GIS toolbox. On the other hand WPS servers can support other process description schemes if they comply with general requirements laid out in the conceptual model.

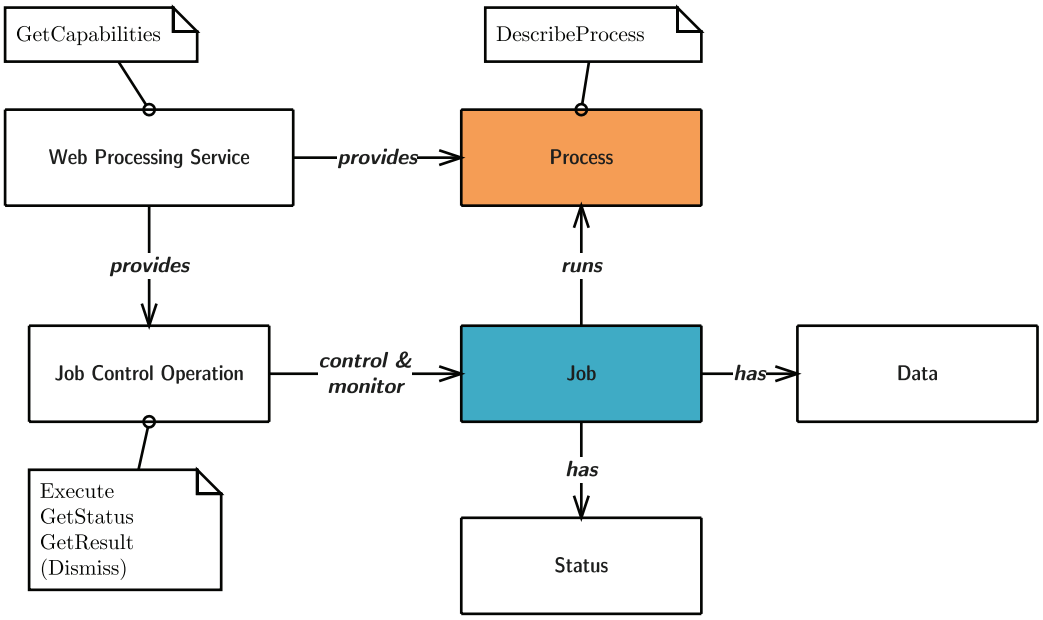


Figure 5.2.: Artefacts of the WPS service model (Source: OGC 2015a, Figure 1)

5.3. WPS Process Model and Interfaces

The WPS standard's process description has been designed to support a broad range of stateless computing functions. For historical reasons WPS uses the term *process* instead of *function*. A process' interface makes an explicit distinction between inputs and outputs. A WPS-compliant process has at least one output and zero[1] or more inputs (cf. Figure 5.3).

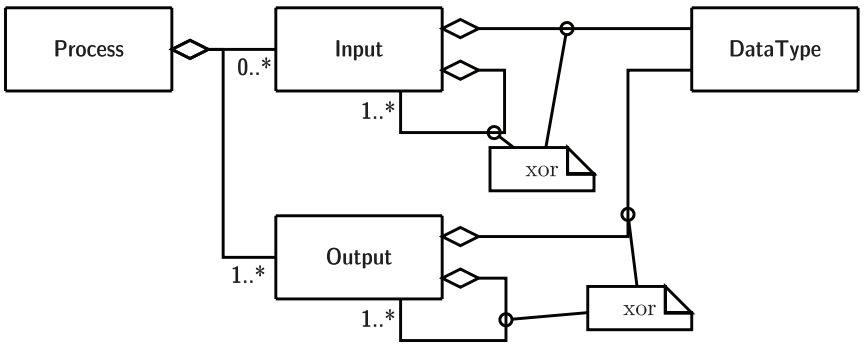


Figure 5.3.: WPS abstract process model (Source: OGC 2015a, Figure 2)

[1]This permits the description of input-free functions such as random value generators (`http://resources.arcgis.com/en/help/main/10.2/index.html/Create_Random_Raster/009z000000s6000000/`).

An input can be multi-valued, i.e. receive one or more input datasets for execution. More complex interfaces with cross dependencies between several inputs or outputs can be modelled with grouped inputs and outputs[2]. The functions *Weighted Sum* and *Reclassify* are typical examples from common GIS toolboxes that have nested arguments and benefit from this feature. Figure 5.4 illustrates how nested inputs may be applied to express remap tables for single values and value ranges.

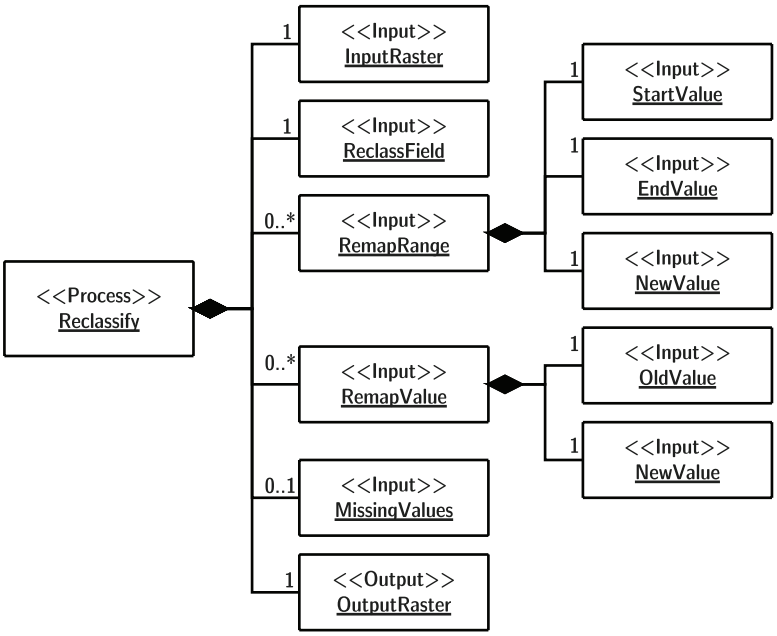


Figure 5.4.: Exemplary process interface for a *Reclassify* function[3]

In WPS 1.0, case-specific data types with distinct (XML) schemas had to be defined for geoprocessing functions with nested arguments. As pointed out by Feuerlicht (2011), such practice easily leads to a deluge of very specific data types. This can become a mjor obstacle in service chaining and eventually impedes service composition.

A UML diagram of the complete WPS process model is shown in Figure 5.5. Process descriptions, inputs, and outputs inherit basic type definitions from OWS Common and comprise a machine readable identifier, a human readable title, an optional abstract, an optional set of keywords[4], and zero or more metadata elements.

In contrast to the prior version, WPS 2.0 makes particular use of the metadata elements to provide additional documentation on the semantics of process interfaces as well as inputs and outputs. Furthermore, metadata elements are used to indicate compliance with one or more process profiles (see section 5.4).

[2]Grouped inputs and outputs were not available in WPS 1.0

[3]Interface adapted from ArcGIS Reclassify tool (`http://desktop.arcgis.com/en/desktop/latest/tools/spatial-analyst-toolbox/reclassify.htm`, accessed 2015-04-01

[4]Keywords were not available in WPS 1.0

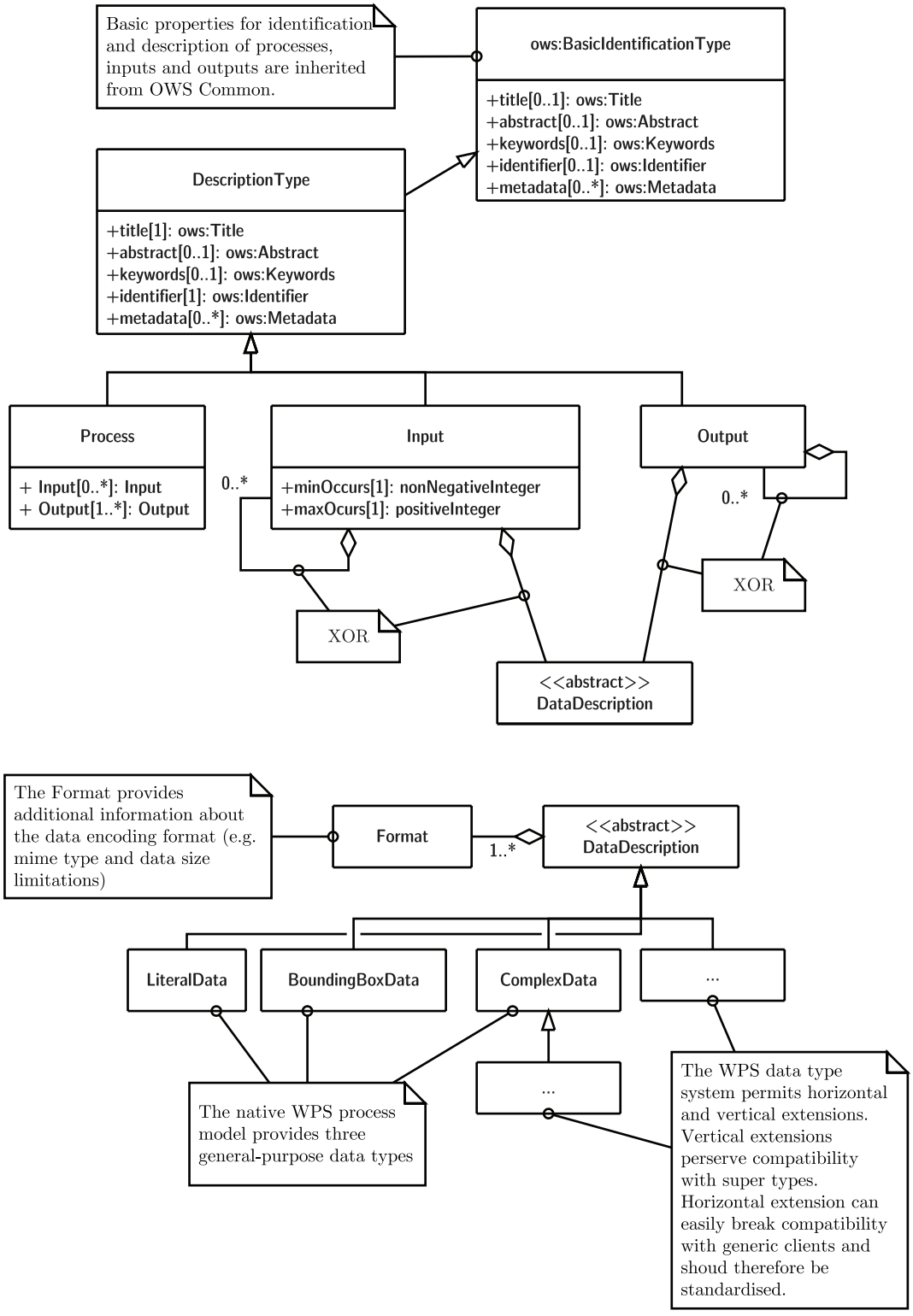


Figure 5.5.: WPS process model information elements (Source: OGC 2015a, Figures 6,7,8,12)

For data exchange, the WPS standard defines two specific and one generic type:

- *Literal Data* – a simple scalar or string value with an optional unit,
- *Bounding Box Data* – a geographic bounding box, and
- *Complex Data* – for all other data types, described by mime type, encoding and schema.

While being developed for geospatial applications, the data model is quite generic and open to use in other domains. With regard to the limited list of available data types, extensions may define additional data types – either by inheritance from *Complex Data* or by extending the abstract *Data Description* class.

5.4. Process Descriptions and Profiles

Hierarchical profiling of geoprocessing functions is another feature which was introduced by WPS 2.0 and is based on the theoretical background provided by Müller (2013) and Müller (2015, cf. chapter 4). The WPS SWG has adopted the general approach for WPS 2.0 and a corresponding XML encoding was created. Other options, such as semantic annotations, are generally possible to add useful metadata and have been discussed in conjunction with WPS (OGC 2009). Yet, semantic annotations have not received significant adoption in the OGC community or the OGC standards suite.[5]

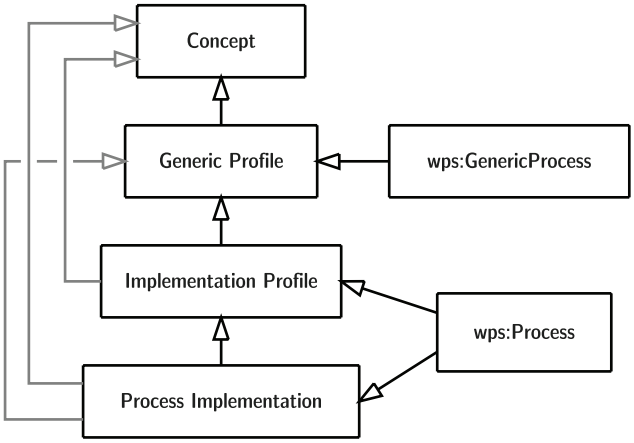


Figure 5.6.: Inheritance hierarchy for multi-level process profiles (Source: OGC 2015a, clauses 7.5 and 8)

WPS 2.0 does not mandate a full profiling hierarchy to reduce the implementation burden for early adopters of process profiles. It permits to start profiling at any

[5]The purpose and use of semantic annotations in SensorML is reviewed in annex A.3.

given level in the hierarchy and allows "gaps", such as omitted generic profiles, in the inheritance graph (cf. Figure 5.6). This allows implementers to document the *status quo* of their geoprocessing services and work towards better descriptions and more complete hierarchies in future iterations.

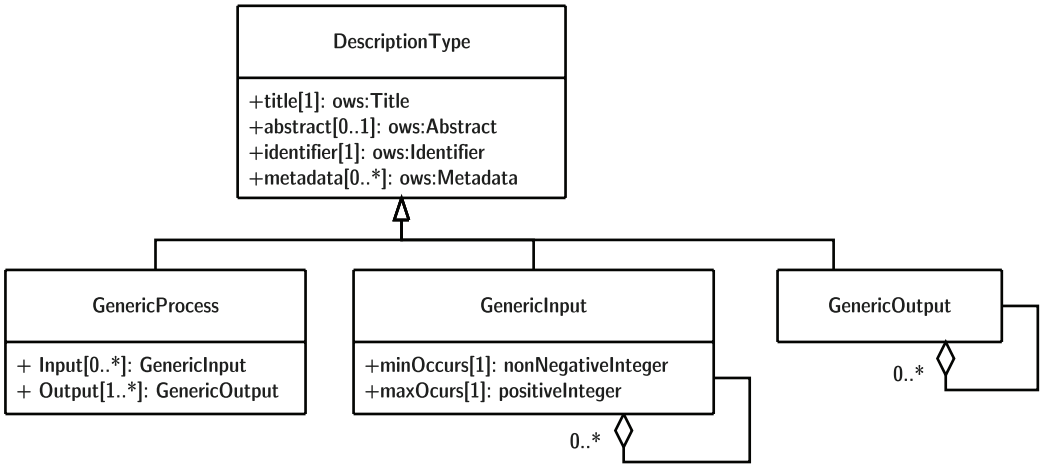


Figure 5.7.: Generic process information elements (Source: OGC 2015a, Figure 13)

A detailed discussion on profiling and inheritance is provided in chapter 4. The WPS implementation of process profiling is based on the UML classes *Process* (cf. Figure 5.5) and *Generic Process* (cf. Figure 5.7). A compilation of XML listings that encode the examples from Müller (2015) into WPS process profiles is provided in annex A.2.

6. Discussion of Results

Today's geospatial data infrastructures largely provide data-centric services. Data access services offer interfaces for data download, view services deliver cartographic visualisations of this data, and catalogue services add search and retrieval capabilities to the infrastructure.

In contrast, processing services are not very well integrated in current SDIs. Access to geoprocessing software is usually limited to experts who still face interoperability challenges if they wish to reuse 3rd party tools or products. Although Web services with geoprocessing capabilities have been around for a while now, clients are generally unable to search and invoke the processing functions that suit their needs. Open Data initiatives and SDIs have helped to establish a culture for data sharing that encourages reuse of previously collected data and avoids costly surveys for gathering data which is aleady available. Similarly, a code sharing infrastructure (cf. chapter 3, Figure 2) may ease access to geoprocessing services and encourage reuse rather than costly, time consuming, and error prone re-implementation. Geoprocessing Web services, such as WPS, are one technical option to achieve code mobility (see section 1.2). A major disadvantage of WPS is the tight coupling of computing functionality and computing resources. Developers of new data processing functions are required to operate a computing service to share their implementations with other users. In this thesis it is agued that portable computing components could be shared and exchanged independently from a computing service. This so-called *moving code* approach significantly reduces provisioning efforts and allows greater flexibility in the choice of the deployment site. Provided that both data and computing components can be transferred to any location in a world-wide network, users are free to perform geoprocessing on arbitrary – rented or owned – computing infrastructure.

6.1. Responses to the Research Challenges

As with data, the ability to share geoprocessing services can be attributed to two independent aspects: The ability to document geoprocessing functions, i.e. to describe, search, discover and compare them as well as the ability to access, enact, or otherwise invoke their implementations. This dichotomy is reflected by the research challenges laid out in section 1.4.

Chapter 2 lays the foundation for a *moving code* architecture that discusses code sharing approaches as an alternative to the provision of geoprocessing Web services. In this contribution, the focus is on improving access to software components for geoprocessing which may represent general purpose functions as well as domain–specific workflows and models. An abstract architecture is developed that connects

the user domain, where geoprocessing functions are used and applied, with the source system domain, in which the corresponding implementations are actually deployed. Furthermore, aspects of life cycle management are discussed in order to supply users with updated implementations.

This abstract architecture is refined and improved in chapter 3 which identifies a set of services and target platforms that support *moving code* approaches, such as *code on demand* or *remote evaluation* (cf. Carzaniga et al. 2007, and Table 1.1). The contribution presents an improved component model that allows a better documentation of runtime requirements for portable software components and also puts a stronger focus on component interoperability. Next to infrastructure services for code sharing, registries are proposed to store interface descriptions for geoprocessing functions of which multiple implementations exist.

Chapter 4 is devoted to improving interoperability of geoprocessing services by the definition of *profiles*, which were initially mentioned in WPS 1.0 and were repeatedly discussed and re-interpreted. Based on these reconsiderations, a hierarchical profiling approach is suggested which is generally compatible with WPS process descriptions and permits the description of geoprocessing services at different levels of detail. That same approach can also be used to harmonise geoprocessing services at different levels of granularity.

In parallel, this thesis has provided some insights and suggestions that have been integrated into the WPS 2.0 specification. Important contributions to this international standard are summarised in chapter 5.

These results are now coherently reviewed and related to the initial research challenges.

Challenge no. 1: Portable Software Components for Geoprocessing

Portable software components have been proposed as an alternative to geoprocessing Web services which may be inappropriate or underperforming in many applications. Next follows a review of the related research questions (cf. section 1.4).

How is code mobility achievable in SDI?

> Code mobility in terms of client–server geoprocessing is well-documented and practised in various cases. The WPS standard (OGC 2007c, 2015a) provides a communication protocol for this purpose. Code mobility in terms of *remote evaluation* or *code on demand* has received less attention and was primarily investigated by this thesis.
>
> Chapter 2 describes different subtypes of *moving code* in relation to data coupling and execution scheme. Some data services such as WCS with sub-setting or resampling extensions or WFS with Filter Encoding provide simple processing capabilities. These operations can be parameterised by the client to perform simple processing tasks on the data offerings. Since the client defines the computing instructions as part of

the query to the data service, these solutions qualify as code mobility in terms of *remote evaluation*. The processing capabilities of data access services are very limited and comprise only a tiny set of operations. More flexibility is offered by domain-specific languages such as WCPS which provides an SQL-like query language for raster computations.

Another option to achieve code mobility at a higher level of abstraction is the definition of algebras for geoprocessing. Instead of using generic programming languages, an algebra could provide a set of atomic spatial or spatio-temporal operations which can be interpreted and executed by multiple GIS or processing systems. Approaches in this direction are discussed by Takeyama (1997), Wesselung et al. (1996), Pullar (2001), Mennis et al. (2005), Ledoux and Gold (2006), or Camara et al. (2014), all of them largely building on Tomlin's *Map Algebra* (Tomlin 1990) or similar foundations. A commonly accepted language or algebra for geoprocessing in general is not yet in sight (Brauner 2015).

In reality, the most versatile compilations of geoprocessing functions are implemented in generic programming languages and typically encoded and shipped as either scripting code or precompiled byte code. For such implementations, the least common denominator is a common interface description language (IDL) to describe the functionality. Added logic or middleware is required to map between the particular programming interface of the implementation and the IDL representation. A copy of the IDL representation, however, can be stored in catalogues or inventories as a stand-alone document to support search and retrieval. A standardised IDL also represents a common integration layer among varois implementations. It is confined to the description of geoprocessing *functions* and hides details about particular source systems or programming languages.

What would be a suitable component model for exchanging the software representations of geoprocessing functions in an interoperable manner?

The actual deployment of code on other computers is – not surprisingly – more demanding than copying bits and bytes. To take an analogy, data sharing requires, besides harmonised schemas and metadata, well-defined data encoding formats. Similarly, code exchange requires the agreement on particular container formats that encode computer instructions. In addition, code deployment needs to consider runtime requirements such as hardware and software dependencies that must be accounted for by a suitable component model. Based on the idea of portable workspaces, which are commonly used in GIS and geoscientific applications, this thesis has outlined requirements and provided a prototypical component model for implementations of geoprocessing functions.

This component model largely builds on existing standards. It uses the well-known interface description language of the WPS standard to cap-

ture syntactic properties of the contained geoprocessing functions and relies on OCCI elements that describe the required hardware resources. Software dependencies and licenses are expressed with resolvable Uniform Resource Identifiers (URIs). Mappings between programming interfaces and interoperable WPS process descriptions are created programmatically and follow a generic command line syntax. These descriptive artefacts are encoded in an XML document.

The actual content (i.e. code and static data) is stored in structured workspaces which are widely used in GIS. Both content and descriptive artefacts are bundled together in a ZIP archive, which is widely used as a general purpose container by other encoding specifications such as Office Open XML (ISO 2008) or JAVA applications.

An implementation of this component model, including support libraries, is available in 52°North's software repository. It provides an API for reading and writing portable code packages and contains the required middleware for automatic deployment and execution. Furthermore, these libraries were integrated in 52°North's WPS framework to support rapid deployment of new functionality on geoprocessing Web services.

What would a possible code sharing architecture look like and which actors and services participate in a code-sharing environment?

Implementations of geoprocessing functions can be provided statically within particular GIS products or dynamically shared and exchanged in a distributed infrastructure. This thesis has particularly investigated distributed provisioning approaches that involve the migration of computing instructions or code in a computer network.

Since the provided code packages may be developed, deployed, and used by independent parties (cf. Figure 1.1; Kadner et al. 2012; Henzen et al. 2015), the following actors must be considered:

- Users which require access to geoprocessing services to perform particular geoprocessing tasks,
- Developers or providers of functionality who want to publish their work and deliver new geoprocessing services or updated implementations,
- Providers of computing environments, i.e. infrastructure and software platforms, in which geoprocessing services can be deployed and executed,
- Users of desktop GIS who want to extend their toolboxes with new geoprocessing functions from Web based repositories, and
- Operators of geoprocessing Web services that provide general purpose or domain-specific collections of geoprocessing functions.

These actors and their requirements have been translated into a conceptual architecture which is presented in chapter 3, Figure 2. In the provisioning layer this architecture provides the following services:

- Discovery services that perform as catalogues to search for available geoprocessing functions,
- Download services that give access to implementations of geoprocessing functions,
- Platform registries that associate software components or compilations of software components with uniform identifiers, and
- A registry for process definitions that store commonly defined profiles for geoprocessing functions.

The application layer depends on these services. It adresses operators of geoprocessing Web services, operators of cloud infrastructures, and users of Desktop GIS which need to search, access, and deploy portable code packages for geoprocessing.

Challenge no. 2: Interoperable Interface Descriptions for Geoprocessing Services

The availability of interoperable interface descriptions for geoprocessing services is vital to search and retrieval tasks as well as for documentation of geoprocessing services in a distributed SDI. Challenges that relate to the documentation and communication of functionality have been raised in section 1.4. These are now reviewed.

How can functional descriptions be implemented in conjunction with existing standards for geoprocessing services?

WPS process descriptions have been suggested as a common IDL for geoprocessing services. These descriptions may be applied to toolboxes of legacy GIS products, portable software components, or geoprocessing Web services. Their I/O-oriented structure is an ideal skeleton to define arguments and results of geoprocessing functions. Findings from this thesis have led to several improvements of process descriptions in WPS 2.0 (OGC 2015a):

- The separation of WPS process descriptions from the Web service protocol permits their application to arbitrary implementations.
- The introduction of documentation links next to syntactic elements in the process description enables improved documentation of functionality, behavioural semantics as well as pre- and post-conditions for inputs and outputs.

- Geoprocessing services may be aligned and compared at different levels of abstraction by the use of hierarchical profiles.

SensorML (OGC 2014c) is another OGC standard that provides a process model for computational functions. The review of SensorML in annex A.3, however, shows no real benefits over WPS process descriptions. Moreover, the SensorML process model has some restrictions that complicate and in some cases prevent its application to geoprocessing services. The notion of aggregate processes is an improvement over WPS' process model and might be adopted to describe composite geoprocessing functions. This future goal is elaborated in section 6.3.

What are meaningful granularities at which geoprocessing services can be described and compared?

This question was addressed in chapter 4 which identifies recurring levels of detail – *concept*, *generic profile*, *implementation profile*, and *implementation* – that provide meaningful descriptions of geoprocessing services. These levels are arranged in a hierarchical fashion and specify behaviour, semantics, and syntax. The WPS 2.0 standard has largely adopted this framework to encourage better descriptions of geoprocessing services and support an alignment of geoprocessing services at different levels of detail.

Descriptive granularity can also be discussed with regard to compositional granularity, e.g. to define a complex geoprocessing function as a composition of simpler and more genric functions. The possibility to include compositional constructs in interface descriptions was not investigated by this thesis but is briefly discussed as future work in section 6.3.

How can SDI catalogues be enhanced to support the search and retrieval of geoprocessing functions and services?

In WPS 2.0, process descriptions have been isolated in a separate conformance class. They can be used as an IDL for arbirary geoprocessing services, i.e. geoprocessing Web services or portable software components. Since they are encoded as stand-alone XML documents, they are technically easy to integrate with existing registries and catalogues.

A proper link between catalogue standards and geoprocessing service descriptions, however, has not yet been established. The standards for search and retrieval of datasets, download services, and visualisation services on one hand and geoprocessing services on the other hand are hardly integrated. For geoprocessing (Web) services, catalogues of the the provided functionality are still uncommon.

Next to a general consideration of geoprocessing services by the OpenGIS Catalogue Services standard (OGC 2007a), e.g. by supporting WPS process descriptions in the metadata records, a more explicit link with existing metadata standards, such as ISO 19115, would be a major step

towards better integration. Some early work on ISO 19115 metadata schemas for geoprocessing functions shows conceptual parallels with WPS process interfaces (Hill et al. 2001). Since then the ISO standard became largely data-centric and thus less suitable to describe and document geoprocessing functions.

ISO 19115-2, however, allows the use of unique identifiers for a "processing package" that has been used to compute derived data. WPS process descriptions as well as process profiles can be identified with HTTP-URIs. These URIs may be embedded into existing metadata records and can replace other (free-text) descriptions of the processing steps. Due to the very limited capabilities in ISO 19115-2, it is almost impossible to document the particular workflow, i.e. the topology of the processing steps, that generated a particular dataset. The mechanism only works in the most simple cases, i.e. if the participating geoprocessing functions produce one single output and are chained sequentially.

6.2. Conclusions

The developed code sharing architecture and the component model for portable code responds to basic requirements encountered in real-world applications. The proposed component model reflects the fact that canned geoprocessing functions are usually available in the form of scripts, executable files and coarse-grained library functions which have a defined signature and can be executed via command line calls.

There are, of course, other component models that might be suitable for sharing implementations of geoprocessing functions. Most of them require a particular runtime infrastructure and may not be easily to port between different operating systems. In an early stage, the use of OSGi (a component model and runtime infrastructure for the JAVA platform) was tested (Schubert 2011) but then abandoned since it is hardly supported outside the JAVA world and is thus hard to adapt to other software platforms and programming languages. Finally, it was decided to develop a custom component model that satisfies the following properties:

- Independence from a particular programming language,
- Support for the expression of software dependencies,
- Support for the expression of hardware requirements,
- Support for comprehensive workspaces, including code and data,
- Exclusive focus on stateless computing functions,
- License documentation, and
- Ability to "wrap" existing implementations to avoid re-implementations just for the purpose of code sharing.

Its main purpose was to transfer requirements into testable and workable technology for prototyping. It is primarily intended to provide an abstraction layer to legacy implementations and to document their capabilities and runtime constraints.

The major downside of a custom packaging format is, of course, the lack of tool support. To mitigate this issue, the component model is divided into manageable parts which adhere as far as possible to open and widely supported standards (cf. chapter 4, section 6).

The workspace concept was found to be a suitable unit of shipment for code and related data. Workspaces may contain the primary programming code, tightly coupled data sets as well as 3rd party libraries that are statically linked to the program code. The workspace concept is widely applied in GIS and other software such as the statistics environment *R*. When ESRI introduced its geoprocessing package with ArcGIS 10.1 in June 2012, it followed a similar approach to bundle the associated tools, scripts, and datasets. In the context of data science, Chirigati et al. (2013) propose a *reproducible package* which contains similar information on computational experiments for computational reproducibility.

Due to the generic nature of the workspace contents, general correctness and deployment tests are hardly possible. Additional quality management mechanisms are required to ensure correctness and portability of the workspaces. Automated tests supplied with each package or manual reviews from users are two obvious possibilities; further approaches are discussed by Kadner et al. (2012). Despite these first suggestions, reliable testing remains an open issue and requires future investigation.

For the prototypical implementation of the *moving code* concept a software library has been developed that supports creating, reading, writing, and out-of-the-box execution of code packages. This library is maintained in the software repository of 52°North GmbH[1] and also integrated into the company's WPS framework to allow dynamic content updates in WPS servers.

Since geoprocessing services are still not well integrated in SDI the proof-of-concept implementation substituted the discovery services with Web feeds (Figure 6.1). These Web feeds are offered by the component providers themselves or intermediates to inform potential consumers about new or updated content. Potentially interested clients may subscribe to this feed and occasionally receive update notifications. This feature may also be used to distribute new functionality or updated implementations among a large number of processing nodes. In contrast to catalogues, Web feeds do not provide a native search protocol that allows filtering for particular functional and non-functional properties. Protocol extensions, such as the OpenSearch API (A9 2015), are reuqired to provide search capabilities in conjunction with Web feeds.

In parallel, the creation of a virtual marketplace or *Geoprocessing Appstore* has been discussed by Kadner et al. (2012) that acts as a hub for providers and consumers of geoprocessing implementations. The most recent developments are reported by Henzen et al. (2015). The Appstore supports the creation of portable packages that contain implementations of new or well-known geoprocessing functions.

[1] `https://github.com/52North/movingcode`

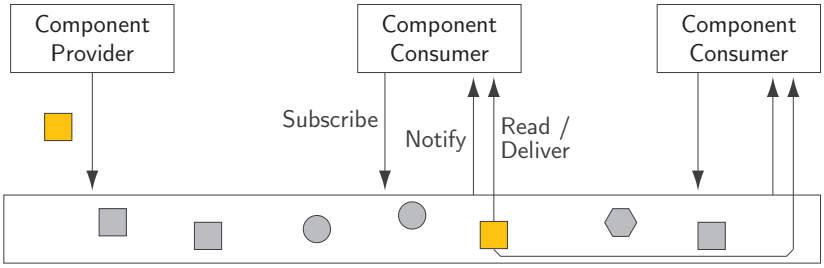


Figure 6.1.: Web feeds for component sharing (Representation adopted from Tanenbaum and Steen 2007)

These packages may be searched in a Web portal and, if the runtime dependencies are supported, executed in a sandbox environment. Compared to the conceptual architecture, the *Geoprocessing Appstore* consolidates several of the described services in a single application and is thus less distributed.

Due to the prominent position of the WPS standard in service-oriented geoprocessing, process descriptions defined in the WPS standard have been suggested as an interoperable, platform independent description language for the interfaces of geoprocessing functions. While being generally suitable to accomplish this task, the earlier version 1.0 of the WPS standard was not created in a modular fashion, and process descriptions were not scoped to be used independently from a WPS server. Besides functional information, process descriptions in WPS 1.0 included Web service specific information such as the support for synchronous and asynchronous execution or the link to alternative WSDL service endpoints. Consequently, the XML encodings for process descriptions had to be redefined to serve the purpose of stand-alone descriptions. Another deficit in WPS 1.0 was the inability to express the process semantics. The available free-text elements in the process description were generally insufficient to capture complex human-readable documentation of processing functions. Furthermore, the role, definition, and scope of WPS profiles, which were intended to provide extended semantics, proved way too unspecific for implementers.

A new modular information model for process descriptions is given in WPS 2.0 (OGC 2015a). Enhancements on process descriptions and process profiles have been developed in the course of this thesis (Müller 2013, 2015) and have shaped the corresponding parts of WPS 2.0. The new conceptual model for process descriptions is specified independently from a Web service protocol and allows interoperable definition and profiling of arbitrary geoprocessing services. The documentation abilities of process descriptions have also been extended to include keywords and comprehensive hypertext documentation.

The *moving code* component model has served as a means to look holistically at implementations of geoprocessing functions and to identify the necessary services of the code sharing architecture. The work on improved documentation elements for WPS process descriptions and the separation of process descriptions form the WPS service model improves the ability to store interoperable and meaningful process

definitions in discovery services, registries, and download services for code packages. These revised description schemas may also be used to document the contents of GIS toolboxes in a more structured way which, if applied to various GIS products, also ease the comparison between different software implementations.

With regard to interoperable interface descriptions for geoprocessing functions this thesis has put an exclusive focus on the discussion of stateless functionality, i.e. *geoprocessing functions* as defined in section 1.1. This definition was largely motivated by a survey of literature which suggested that the strengths of GIS lie in component composition and management and not in fine-grained simulation and modelling (Longley et al. 2011). Most GIS practitioners probably use a workflow approach where models take the form of complex chains of primitive operations. The *Geographic Science and Information Technology Body of Knowledge* (DiBiase et al. 2006) lists a broad range of primitive measures, operations, and methods that are commonly found in GIS and used for a plethora of analysis workflows.

Of course, the restriction to stateless functionality might be challenged, and some authors argue that dynamic simulation is an important capability of GIS software (Schmitz et al. 2013). Dynamic and thus stateful components, however, introduce a lot more complexity due to time-dependent behaviour. This includes the agreement on a common *time base* that is used by all dynamic compartments as well as the management and communication of *state* between these components (cf. Zeigler et al. 2000). Common GIS software has little support for dynamic modelling. With an apt use of workflow environments, dynamic behaviour can be mimicked in many cases, thus avoiding the use of stateful components. Most workflow languages support a set of generic control structures for iterations, conditional branching etc. that can be used to create simple forms of feedback (vander Aalst et al. 2003).

A large part of this thesis has discussed improvements on functional descriptions for geoprocessing functions. The suggested profiling mechanism defines functionality in a top–down fashion. If properly applied, this approach ensures harmonised implementations of concepts and generic profiles, and finally leads to exchangeable geoprocessing services. A downside of this approach is the management overhead with a growing hierarchical structure.

Another option is to tag existing implementations according to well-defined aspects or common properties. This approach was used, for instance, by Brauner (2015) who defined an extensible set of aspects that characterise existing implementations. Similarly, Granell et al. (2013) has proposed a linked data approach that captures properties of implementations through well-defined link relations. The descriptions of implementations might be easier to create and manage in one of these bottom-up approaches. Exchangeability, however, may only be guaranteed with regard to particular properties, i.e. the defined aspects (cf. Brauner 2015) or link relations (cf. Granell et al. 2013). The challenge for both approaches is to provide a *complete* feature space that fully captures the behaviour of a geoprocessing service, which may be hard to achieve within a loosely defined tagging system. In contrast, a human-readable description (or *service contract*, cf. Meyer 1992; Erl 2007) at sufficient detail is achievable for any service interface or implementation.

6.3. Outlook

For some of the addressed research challenges this thesis delivered only partial answers and some issues remain. At the same time, some new challenges and research topics could be identified. Hence this section discusses future development perspectives for service-oriented geoprocessing.

Coexistence of code mobility paradigms

Today, the solutions for distributed geoprocessing are manifold, and several solutions will probably coexist in the near future. Toolboxes in Desktop GIS are used by the majority of GIS experts. In parallel, larger institutions such as the United States Geological Survey (USGS) and Deutsches Klimarechenzentrum (DKRZ) have centralised some of their geoprocessing resources (i.e. functionality and computing resources) in Web services.[2][3] Similarly, ESRI ships lean geoprocessing services for geometric computations with ArcGIS server or provides public access to these services through its ArcGIS Online platform. And geospatial cyberinfrastructures (Wang et al. 2013), that heavily borrow concepts from cloud computing, provide both private Web services for geoprocessing and portable components that can be invoked by their users.

A future challenge is the management of these different provisioning options for geoprocessing services. The *moving code* concept may be an enabler in many of these scenarios. It may be used to supply new or updated functionality to public or private Web services and thus facilitate continuous deployment which is primarily convenient for developers and service administrators. It also allows the addition of new and updated geoprocessing functions in local desktop environments, at best with automated toolbox updates for the most popular Desktop GIS products. This last scenario has not yet been prototyped and is thus left as future work.

If Web services are used as an integration layer between different computing platforms, a combined use of geoprocessing Web services and *moving code* can deliver scalable systems. *Moving code* contributes to a granular and portable software stack from which new service instances can be created on demand, thus allowing for dynamic resource allocation that matches current or expected load conditions in a distributed environment.

Since geoprocessing Web services provide access to self-contained stateless functionality, they can easily support ad-hoc integration. The provision of RESTful Web service interfaces is usually preferred in such scenarios while WS-* or SOAP services rather suit complex enterprise application integration scenarios (Pautasso et al. 2008). With the new abstract service model in WPS 2.0 RESTful WPS service interfaces may be specified in compliance with OGC guidelines.

[2] `http://cida.usgs.gov/gdp`, accessed 2015-08-18
[3] `https://mouflon.dkrz.de`, accessed 2015-08-18

Composite functions, workflows, and lineage

Another field for future research is the better integration of functional descriptions with metadata standards and workflow languages as well as the development of a notation for composite (or aggregate) geoprocessing functions. These tasks are mutually dependent and require coherent actions (Figure 6.2).

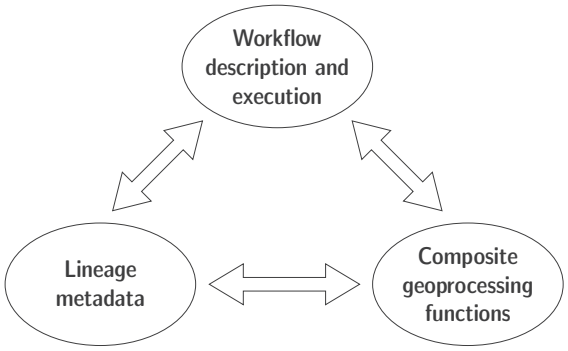


Figure 6.2.: Mutual dependencies between workflow descriptions, lineage records, and composite geoprocessing functions

As stated in section 6.2, catalogue standards do not provide adequate support for lineage documentation. For optimal transparency, a lineage record should be at a level of detail that permits a reproduction of the original workflow which generated the data in the first place. This raises questions about shared foundations: Is it acceptable to have a set of standards for workflow creation and execution on one hand and then another set of standards for the documentation of lineage on the other? In such cases, the use of separate incoherent standards and schemas creates a risk of loss of information.

A similar overlap appears beween workflow languages and interface descriptions for geoprocessing functions. As the review of SensorML (cf. annex A.3) shows, it could be beneficial to express composite functions as a workflow of atomic (or other composite) geoprocessing functions. A common example is a *Buffer* function combined with a *Dissolve* operation that merges overlapping buffer zones of individual features into a single geometry. Since composite geoprocessing functions are basically no different from workflow fragments, they can be described in existing workflow languages and notations.

Finally, a distributed inventory of well-defined and well-documented processing components can have very positive effects on existing data infrastructures. Transparent descriptions of data processing workflows are considered good practice in science and industry. Metadata standards such as ISO 19115 (ISO 2014, 2009) provide lineage information that comprises detailed information about the processing steps that were applied to the data. Currently, the documentation largely consists of free-text elements that are individually appended to each data record. In the future, lineage records and workflows might directly link to the definitions of geoprocessing

functions thus providing a simpler and more consistent approach to interoperable service chaining and data documentation.

Integrated approaches for interface descriptions, service discovery, and service access

Maintaining related interface specifications in registries and catalogues is one possible approach to assist in search and retrieval tasks. It is largely compatible with catalogue concepts in a SOA (cf. Figure 1.2) which requires that copies of the interface descriptions of catalogued services are stored by the catalogue operator to support clients in searching particular functionality. Interface descriptions are usually encoded in some standardised format and aggregate syntactic and semantic information about an implementation or serve as common specifications for multiple implementations. The suggested profiling approach extends this concept by defining several levels of granularity and declaring a set of transition rules that coordinate inheritance between the different levels.

In parallel, resource-oriented approaches have been proposed recently, e.g. by Brauner (2015) and Granell et al. (2013), which use flexible linked data structures instead of static schemas. These approaches treat geoprocessing services, executable environmental models, or *geooperators* as resources that can be related to other resources by *role links*. Those other resources might be related processing services, documentation resources, or terms and definitions from formally specified ontologies and controlled vocabularies. Resource-oriented linked data approaches do not necessarily require catalogues; the "meaning" or functionality of a particular resource is implicitly defined by the links to neighbouring resources.

Profiling and resource linking can be considered complementary. Profiling permits detailed human-readable descriptions and subtyping. Resource linking is ideal to document relationships and cross-cutting properties that are hard to isolate in a hierarchy. Distance metrics (Albrecht 1996), geometric precision, abstract data models (e.g. fields and objects Couclelis 1992), or Kuhn's 10 high-level concepts of spatial information (Kuhn 2012) are some examples. Such recurring properties should be identified and collected in controlled vocabularies to assist the comparison and browsing of different geoprocessing services.

Provision of attractive geoprocessing services and best practice implementations

While the scientific contributions have theoretically shown the benefits of distributed architectures for geoprocessing, their successful implementation depends on the availability of valuable functionality, i.e. useful, robust, and interoperable geoprocessing services. Approaches that lower the technical barrier to sharing geoprocessing services were discussed in this thesis. The development of a comprehensive Web based repository that contains a large set of attractive geoprocessing services is a pending practical task that is probably required to promote, assess, and evolve the *moving*

code approach. Starting with one or more domain-specific repositories that complement existing collections of geoprocessing tools might also be a good starting point. For instance, Müller et al. (2012) discuss a possible use case that could create an added value on top of the the Open Street Map data base by providing dedicated generalisation and filtering tools. Ongoing efforts of GIS vendors and open source developer communities to replace static and tightly integrated toolsets in monolithic GIS with open Web based tool repositories for geoprocessing could also bring about a paradigm shift.

In the end, SDI do not just comprise services and technology but also include people, organisations, and institutions that have an interest in sharing and effectively using geographic information (Bernard et al. 2005). This thesis has discussed the benefits of interoperable, service-oriented geoprocessing and the potential value of reusable implementations. It will be interesting to see whether community activities or the initiatives of particular companies can promote the use of interoperable geoprocessing services at larger scale. Institutions may also realise that they have an interest in advancing these technologies to avoid a data deluge (Economist 2010). Resource-efficient handling of data, computing resources, and analysis software is a key issue for creating information and added value of the enormous data volumes collected by remote sensing programmes, in-situ sensor networks, mapping companies and communities, and mobile devices in the emerging internet of things.

7. Summary

Many of the ongoing activities towards establishing SDI on a regional, national, and international level are focused on data sharing and dissemination. Next to data access and visualisation, data processing is a third important pillar of GIS which can generate new insights by creating derived data or conducting computational analyses on original data sets.

Within today's largely data-centric SDI, two major challenges were identified that stand in the way of ubiquitous geoprocessing: The ability to use and exchange implementations of geoprocessing functions as freely as geographic data and the ability to describe, communicate, and catalogue existing functionality on the Web.

The *moving code* approach was evaluated as an alternative to client–server processing in a distributed SDI. While classical client–server configurations move pieces of data between remote processing services, data services, and clients over the network, *moving code* setups consider the transfer of portable software components to the location of data or the location of computing resources. This approach is beneficial in the following cases:

- It is easier, cheaper, and faster to ship the code instead of sending volumes of data over the network,
- The network connectivity is too unstable for reliable client–server processing,
- Geoprocessing services need to be versioned and archived to ensure exact reproducibility,
- Identical geoprocessing services need to be distributed among a large number of processing nodes (consumers), especially if these implementations receive a lot of maintenance, i.e. are constantly enhanced and updated,
- The provision of processing functionality, computing resources, and data are to be treated as separate concerns, i.e. for scheduling processing tasks efficiently in a distributed computing environment and process the data in arbitrary network locations, and
- Sustainable, resource-efficient provision and reuse of implementations are a foremost concern.

Classical client–server processing over the web may not be feasible at all if:

- The data to be processed falls under privacy obligations and is not to be exchanged with third parties or to be transported over the web,

- A network connection is unavailable or can only be established during short temporary periods, and
- Data transport times over the network are prohibitively long (e.g. due to large data volumes).

To exchange implementations of geoprocessing functions at a larger scale, a Web based code sharing architecture was proposed. Besides the description of functionality, which is a general prerequisite for cataloguing mobile code (section 1.3), consumers of portable software components need further information about the required software environment, the processing hardware, and the terms of use. Furthermore, this architecture considers a range of deployment targets in a distributed infrastructure, such as computing nodes that offer Web services for geoprocessing, cloud computing environments, or individual workstations.

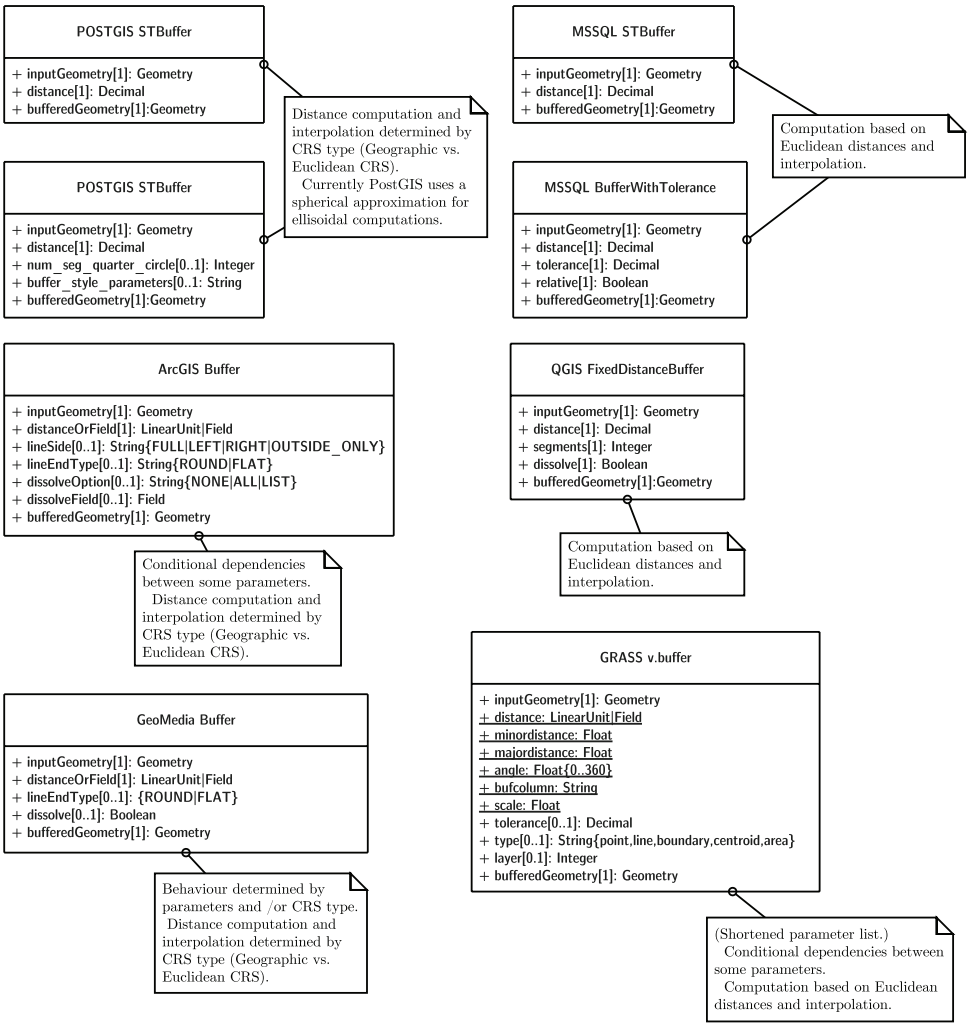
For improved description and documentation of geoprocessing functions, this thesis has contributed to the revised modular information model of WPS 2.0 which facilitates the use of WPS process descriptions as an interface description language for geoprocessing services. Furthermore, it has provided conceptual foundations for WPS process profiles and contributed a suitable encoding for WPS 2.0.

Future research is required to coordinate approaches towards a better integration of (interoperable) process descriptions, workflow descriptions, and lineage metadata. These aspects have a significant overlap but are treated rather independently in recent standards. Similarly, consolidated efforts are also required to establish commonly accepted cataloguing schemes for geoprocessing services. Further investigations of an alignment of interface descriptions for geoprocessing functions and *linked data* approaches for discovery and documentation are suggested.

A. Annex

A.1. Comparison of Interfaces for Buffer Functions

The figure below shows an overview of Buffer interfaces in different Desktop GIS and spatial databases. It was prepared for a report to the WPS standards working group at the Technical Committee Meeting in Mumbai, 2013. Some interfaces may have changed slightly in recent software releases but their diversity is just the same.



A.2. Process Description Examples for WPS

Listing A.1: Generic Profile for a *Precision Geodesic Distance Buffer* function

```
<wps:GenericProcess xmlns:wps= ... >

  <!-- Title, abstract, and unique identifier of the process interface -->
  <ows:Title>Precision Geodesic Distance Buffer</ows:Title>
  <ows:Abstract>
    Returns a geometry that represents all points whose distance from this
    Geometry is less than or equal to distance. Buffer distance calculations are
    performed on the ellipsoid defined by the Geometry's spatial reference system.
  </ows:Abstract>
  <ows:Identifier>http://.../PrecisionGeodesicDistanceBuffer</ows:Identifier>

  <!-- Metadata related to superior process profiles -->
  <!-- Metadata links to superior concepts -->
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/concept"
    xlink:href="http://.../Buffer"/>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/concept"
    xlink:href="http://.../DistanceBuffer"/>

  <!-- Metadata links to superior generic profiles -->
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/generic"
    xlink:href="http://.../GeodesicDistanceBuffer"/>
  <!-- Documentation link describing the process behaviour and semantics -->
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
    xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html"/>

  <!-- Title, abstract, and identifier of the 1st input -->
  <wps:Input>
    <ows:Title>Input Geometry</ows:Title>
    <ows:Abstract>Geometry to be buffered</ows:Abstract>
    <ows:Identifier>geometry</ows:Identifier>

    <!-- Documentation link for the 1st input. The link points to an anchor
    element within process documentation page. -->
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
      xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html#geometry"/>
  </wps:Input>

  <!-- Title, abstract, and identifier of the 2nd input -->
  <wps:Input>
    <ows:Title>Distance</ows:Title>
    <ows:Abstract>Buffering distance</ows:Abstract>
    <ows:Identifier>distance</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
      xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html#distance"/>
  </wps:Input>

  <!-- Title, abstract, and identifier of the 3rd input -->
  <wps:Input>
  <ows:Title>Distance Tolerance</ows:Title>
    <ows:Abstract>
      Permissible distance tolerance of the buffer. Buffer computation is guaranteed to
```

```
      be precise with in these error margins.
    </ows:Abstract>
    <ows:Identifier>tolerance</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
      xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html#tolerance"/>
  </wps:Input>

  <!-- Title, abstract, and identifier of the (single) output -->
  <wps:Output>
    <ows:Title>Buffered Geometry</ows:Title>
    <ows:Abstract>
      Geometry representing a Geodesic Distance Buffer of the input geometry.
    </ows:Abstract>
    <ows:Identifier>distanceBuffer</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
      xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html#distance_buffer"/>
  </wps:Output>

</wps:GenericProcess>
```

Listing A.2: Implementation Profile for a *Precision Geodesic Distance Buffer* for GML

```
<wps:Process xmlns:wps= ... >
  <!-- Title, abstract, and unique identifier of the process interface -->
  <ows:Title>Precision Geodesic Distance Buffer</ows:Title>
  <ows:Abstract>
    Returns a geometry that represents all points whose distance from this
    Geometry is less than or equal to distance. Buffer distance calculations are
    performed on the ellipsoid defined by the Geometry's spatial reference system.
  </ows:Abstract>
  <ows:Identifier>
    http://.../PrecisionGeodesicDistanceBufferGML
  </ows:Identifier>

  <!-- Metadata related to superior process profiles -->
  <!-- Metadata links to superior concepts -->
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/concept"
    xlink:href="http://.../Buffer"/>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/concept"
    xlink:href="http://.../DistanceBuffer"/>

  <!-- Metadata links to superior generic profiles -->
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/generic"
    xlink:href="http://.../GeodesicDistanceBuffer"/>

  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/generic"
    xlink:href="http://.../PrecisionGeodesicDistanceBuffer"/>

  <!-- Documentation link describing the process behaviour and semantics -->
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
    xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html"/>

  <!-- Title, abstract, and identifier of the 1st input -->
```

```
<wps:Input>
  <ows:Title>Input Geometry</ows:Title>
  <ows:Abstract>Geometry to be buffered</ows:Abstract>
  <ows:Identifier>geometry</ows:Identifier>

  <!-- Documentation link for the 1st input. The link points to an
    anchor element within process documentation page. -->
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
    xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html#geometry"/>
  <wps:ComplexData>
    <wps:Format mimeType="application/gml+xml" encoding="UTF-8"
      schema="http://schemas.opengis.net/gml/3.2.1/gml.xsd#AbstractFeature"
      default="true"/>
  </wps:ComplexData>
</wps:Input>

<!-- Title, abstract, and identifier of the 2nd input -->
<wps:Input>
  <ows:Title>Distance</ows:Title>
  <ows:Abstract>Buffering distance</ows:Abstract>
  <ows:Identifier>distance</ows:Identifier>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
    xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html#distance"/>
  <wps:LiteralData>
    <wps:Format mimeType="text/plain" default="true"/>
    <wps:Format mimeType="text/xml"/>
    <LiteralDataDomain default="true">
      <ows:AllowedValues>
        <ows:Range>
          <ows:MinimumValue>-INF</ows:MinimumValue>
          <ows:MaximumValue>INF</ows:MaximumValue>
        </ows:Range>
      </ows:AllowedValues>
      <ows:DataType
        ows:reference="http://www.w3.org/2001/XMLSchema#double">Double
      </ows:DataType>
    </LiteralDataDomain>
  </wps:LiteralData>
</wps:Input>

<!-- Title, abstract, and identifier of the 3rd input -->
<wps:Input>
<ows:Title>Distance Tolerance</ows:Title>
  <ows:Abstract>
    Permissible distance tolerance of the buffer. Buffer computation is guaranteed to
    be precise with in these error margins.
  </ows:Abstract>
  <ows:Identifier>tolerance</ows:Identifier>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
    xlink:href="http://.../generic/PrecisionGeodesicDistanceBuffer.html#tolerance"/>
  <wps:LiteralData>
    <wps:Format mimeType="text/plain" default="true"/>
    <wps:Format mimeType="text/xml"/>
    <LiteralDataDomain default="true">
      <ows:AllowedValues>
        <ows:Range>
          <ows:MinimumValue>-INF</ows:MinimumValue>
          <ows:MaximumValue>INF</ows:MaximumValue>
        </ows:Range>
```

```
        </ows:AllowedValues>
        <ows:DataType
          ows:reference="http://www.w3.org/2001/XMLSchema#double">Double
        </ows:DataType>
      </LiteralDataDomain>
    </wps:LiteralData>
  </wps:Input>

  <!-- Title, abstract, and identifier of the (single) output -->
  <wps:Output>
    <ows:Title>Buffered Geometry</ows:Title>
    <ows:Abstract>
      Geometry representing a Geodesic Distance Buffer of the input geometry.
    </ows:Abstract>
    <ows:Identifier>distanceBuffer</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
      xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html#distance_buffer"/>
    <wps:ComplexData>
      <wps:Format mimeType="application/gml+xml" encoding="UTF-8"
        schema="http://schemas.opengis.net/gml/3.2.1/gml.xsd#AbstractFeature"
        default="true"/>
    </wps:ComplexData>
  </wps:Output>

</wps:Process>
```

Listing A.3: Process implementation that realises a *Precision Geodesic Distance Buffer* for GML and GeoJson

```
<wps:Process xmlns:wps= ... >
  <!-- Title, abstract, and unique identifier of the process interface -->
  <ows:Title>Precision Geodesic Distance Buffer</ows:Title>
  <ows:Abstract>
    Returns a geometry that represents all points whose distance from this
    Geometry is less than or equal to distance. Buffer distance calculations are
    performed on the ellipsoid defined by the Geometry's spatial reference system.
  </ows:Abstract>
  <ows:Identifier>
    http://.../my-implementation-of-a-PrecisionGeodesicDistanceBuffer
  </ows:Identifier>

  <!-- Metadata related to superior process profiles -->
  <!-- Metadata links to superior concepts -->
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/concept"
    xlink:href="http://.../Buffer"/>
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/concept"
    xlink:href="http://.../DistanceBuffer"/>

  <!-- Metadata links to superior generic profiles -->
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/generic"
    xlink:href="http://.../GeodesicDistanceBuffer"/>

  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/generic"
    xlink:href="http://.../PrecisionGeodesicDistanceBuffer"/>

  <ows:Metadata
```

```
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/implementation"
    xlink:href="http://.../PrecisionGeodesicDistanceBufferGML"/>

  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process-profile/implementation"
    xlink:href="http://.../PrecisionGeodesicDistanceBufferGeoJson"/>

  <!-- Documentation link describing the process behaviour and semantics -->
  <ows:Metadata
    xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
    xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html"/>

  <!-- Title, abstract, and identifier of the 1st input -->
  <wps:Input>
    <ows:Title>Input Geometry</ows:Title>
    <ows:Abstract>Geometry to be buffered</ows:Abstract>
    <ows:Identifier>geometry</ows:Identifier>

    <!-- Documentation link for the 1st input. The link points to an
      anchor element within process documentation page. -->
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
      xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html#geometry"/>
    <wps:ComplexData>
      <wps:Format mimeType="application/gml+xml" encoding="UTF-8"
        schema="http://schemas.opengis.net/gml/3.2.1/gml.xsd#AbstractFeature"
        default="true"/>
      <wps:Format mimeType="application/json" encoding="UTF-8"
        schema="http://geojson.org/geojson-spec.html"/>
    </wps:ComplexData>
  </wps:Input>

  <!-- Title, abstract, and identifier of the 2nd input -->
  <wps:Input>
    <ows:Title>Distance</ows:Title>
    <ows:Abstract>Buffering distance</ows:Abstract>
    <ows:Identifier>distance</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
      xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html#distance"/>
    <wps:LiteralData>
      <wps:Format mimeType="text/plain" default="true"/>
      <wps:Format mimeType="text/xml"/>
      <LiteralDataDomain default="true">
        <ows:AllowedValues>
          <ows:Range>
            <ows:MinimumValue>-INF</ows:MinimumValue>
            <ows:MaximumValue>INF</ows:MaximumValue>
          </ows:Range>
        </ows:AllowedValues>
        <ows:DataType
          ows:reference="http://www.w3.org/2001/XMLSchema#double">Double
        </ows:DataType>
      </LiteralDataDomain>
    </wps:LiteralData>
  </wps:Input>

  <!-- Title, abstract, and identifier of the 3rd input -->
  <wps:Input>
  <ows:Title>Distance Tolerance</ows:Title>
    <ows:Abstract>
      Permissible distance tolerance of the buffer. Buffer computation is guaranteed to
```

```
      be precise with in these error margins.
    </ows:Abstract>
    <ows:Identifier>tolerance</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
      xlink:href="http://.../generic/PrecisionGeodesicDistanceBuffer.html#tolerance"/>
    <wps:LiteralData>
      <wps:Format mimeType="text/plain" default="true"/>
      <wps:Format mimeType="text/xml"/>
      <LiteralDataDomain default="true">
        <ows:AllowedValues>
          <ows:Range>
            <ows:MinimumValue>-INF</ows:MinimumValue>
            <ows:MaximumValue>INF</ows:MaximumValue>
          </ows:Range>
        </ows:AllowedValues>
        <ows:DataType
          ows:reference="http://www.w3.org/2001/XMLSchema#double">Double
        </ows:DataType>
      </LiteralDataDomain>
    </wps:LiteralData>
  </wps:Input>

  <!-- Title, abstract, and identifier of the (single) output -->
  <wps:Output>
    <ows:Title>Buffered Geometry</ows:Title>
    <ows:Abstract>
      Geometry representing a Geodesic Distance Buffer of the input geometry.
    </ows:Abstract>
    <ows:Identifier>distanceBuffer</ows:Identifier>
    <ows:Metadata
      xlink:role="http://www.opengis.net/spec/wps/2.0/def/process/description/documentation"
      xlink:href="http://.../PrecisionGeodesicDistanceBuffer.html#distance_buffer"/>
    <wps:ComplexData>
      <wps:Format mimeType="application/gml+xml" encoding="UTF-8"
        schema="http://schemas.opengis.net/gml/3.2.1/gml.xsd#AbstractFeature"
        default="true"/>
      <wps:Format mimeType="application/json" encoding="UTF-8"
        schema="http://geojson.org/geojson-spec.html"/>
    </wps:ComplexData>
  </wps:Output>

</wps:Process>
```

A.3. SensorML Process Interfaces

One of the challenges laid out in section 1.4 is the provision of interoperable interfaces for geoprocessing functions. The obtained interface descriptions may then be applied to any kind of implementation of a geoprocessing function, i.e. a built-in geoprocessing tool that ships with a particular GIS product, a portable software component or a geoprocessing Web service. The contributions in chapters 2 and 3 use and evolve WPS process descriptions to model interfaces for geoprocessing services. For WPS 2.0 this description model has been revised to be used independently from a Web service, to provide additional metadata on the process semantics, and to support hierarchical profiling of processing functions (cf. chapter 5).

With the Sensor Model Language (SensorML; OGC 2014c) OGC provides another specifications that models process interfaces. SensorML is part of OGC's standards suite for Sensor Web Enablement (SWE; OGC 2015b) and provides a process model which is focused on the pre- and post-processing stages for sensor measurements. This section reviews the SensorML process model and assesses its capabilities toward the WPS process model.

The properties of any SensorML process are summarised by the specification of an *Abstract Process* which has three types of arguments: *InputList*, *Outputs* and *Parameters* (cf. Figure A.1). These arguments are organised in lists and support the same three data types:

- *Abstract Data* – an abstract type that comprises basic data types defined in SWE Common such as Quantity, Count, Category, Boolean, Text etc.,
- *Data Interface* – provides access to a *Data Stream* defined by SWE Common (OGC 2011), and
- *Observable Property* – a physical observable or otherwise measurable property which has a value and an ambiguous identifier; usually used to describe a physical stimulus in a larger sequence of processes as an input for a detector.

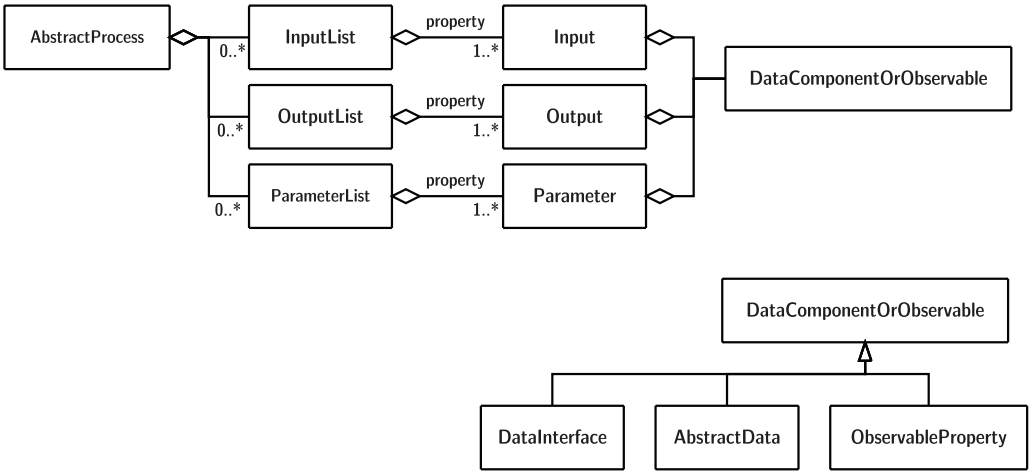


Figure A.1.: Elements of the SensorML Process specification (simplified)

A *Parameter* is defined as special kind of input which has the same semantics as a regular *Input* but changes less frequently. Multi-valued inputs, which are common in WPS, are not supported by SensorML.

Based on the definition of an *Abstract Process*, SensorML defines different subtypes of processes which are shown in Figure A.2. *Simple Processes* are typically used to describe computational processes that are either atomic or require no further decomposition. *Physical Processes* and *Physical Components* represent abstractions of physical devices or observation equipment), such as observation procedures of

physical phenomena or physical devices that interact with the observation equipment (e.g. a gimbal for a camera).

Both simple and physical processes share a *Process Method* element that are used to provide information aout the methodology by which a process generates outputs based on inputs and parameter values. In the case of a *Simple Process* (see next paragraphs), the *Process Method* element should contain a description that is precise enough to allow software developers creating compliant software implementations. SensorML currently mandates a free text description but envisages that future profiles of SensorML use more formal specifications, e.g. in MathML (OGC 2014c, clause 8.2.2). The possible use of semantic annotations to communicate the meaning of processes or data items is briefly mentioned but not further elaborated. That said, at the current state SensorML considers mainly syntactic properties of processes.

Besides atomic process types SensorML provides specifications for composite processes. An *Aggregate Process* represents a collection of interconnected *Abstract Processes* expressed as a component list, including connections between them.[1] Based on this structure, a graph can be computed that describes the data flow between all participating components. Assuming that the contained processes are well-defined, the graph structure sufficiently describes the data flow and thus provides a transparent description of the *Aggregate Process*.

The *Physical System* is structurally similar to an *Aggregate Process* but is intended to describe a real-world system that consists of physical sub-processes. A *Physical System* must comply with the following restrictions:

- The participating components must be instances of the *Physical Component* and
- The specification of the Physical System shall be a subtype of an *Abstract Physical Process*.

As shown in Figure A.2 there are some redundancies in the specification of the *Physical Component* and the *Simple Process* as well as between the *Aggregate Process* and the *Physical System*. Ideally the common properties should be deferred to abstract super classes and propagated to the implementing subclasses. Such a structure would require multiple inheritance, which is easily achievable in UML but not available in XML (XML does not permit inheritance from multiple base types). So here the intended encoding language has probably impacted the design of the conceptual model.

The SensorML process model also supports inheritance and considers processes at different levels of abstraction (*Abstract Process*, *Configurable Process*). Two basic types of inheritance are considered. Simple inheritance only permits the addition of descriptive elements or metadata. Inheritance with configuration extends simple inheritance and allows setting or restricting properties of the supertype such as the

[1] There is no formal syntax for specifying inputs and outputs in SensorML. Their intended use, however, is somewhat apparent from the provided examples.

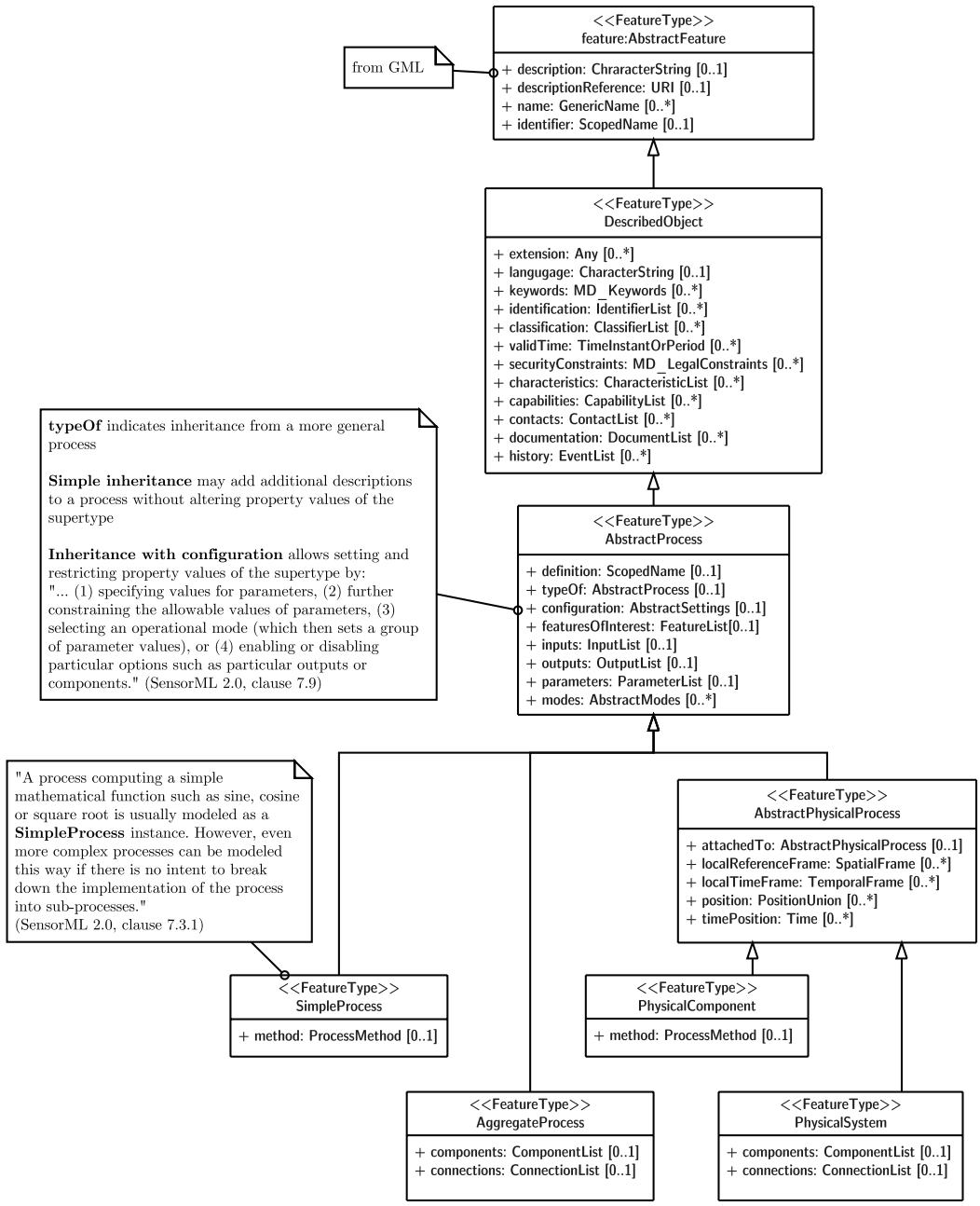


Figure A.2.: SensorML Process types overview (Source: OGC 2014c, condensed and commented)

value ranges of process parameters and outputs (OGC 2014c, clause 7.2.3.3). Dependence on a supertype process is indicated by the *type Of* property of the *Abstract Process*.

Next to the shared properties of SensorML processes and WPS processes there are significant differences. The scope of SensorML's process model is broader than the WPS process model. The *Simple Process* comes closest to the notion of a process in WPS since both process models shall be used for computational processes. The main differences between the SensorML *Simple Process* and the WPS process model concern the available data models and the meta-structure for the process interface. All sensor ML process interfaces receive (and deliver) values that comply with the types defined in the SWE data model. In contrast, WPS processes are open to arbitrary data formats which are specified by their mime type, encoding, and schema (cf. section 5.3).

The interface meta-structure for SensorML processes is rather flat. Each input may receive values from exactly one data source and each output sends result data to exactly one data sink. Since data items in SensorML support streaming, multiple values are supplied sequentially to a SensorML process on the same stream. The number of arguments is defined at design time, i.e. linking multiple streams to the same input is not possible. To pass multiple input sources to the same port of a SensorML process, they need to be aggregated first and then passed as a single stream to the process.

WPS, in contrast, supports multi-valued arguments, so the definite number of data sources passed to a process can be defined at runtime. For instance, a mosaic process may receive an arbitrary number of tiled georeferenced input images which will be merged into a single image by a mosaic function. The inability to use multi-valued arguments is the greatest obstacle that prevents SensorML's application in formalising geoprocessing functions. Nested inputs, which have been introduced by WPS 2.0 are also not available in SensorML, which might lead to inconvenient interfaces for some functions (e.g. the *Reclassify* function in section 5.3). Instead SensorML uses argument lists of a fixed size that are obviously designed towards stream processing where the number of input and output streams or items is known at design time.

The provision of machine readable process specifications is also an issue in SensorML. It largely specifies the syntactic properties of processes; the challenge of machine-readable process is deferred to subsequent profiling activities. SensorML supplies a broad collection of descriptive elements that might hold semantic descriptions. Furthermore it even suggests the use of semantic annotations if these elements should prove insufficient. The strong overlap in the responsibility of descriptive elements and metadata encodings is likely to create a deluge of metadata and governance practices without achieving true metadata interoperability.

An interesting feature of SensorML's process model is the ability to specify *Aggregate Processes*. The wiring of atomic components has strong similarities with simple workflows. Besides orchestration and execution such workflows may also be used to generate process representations for both humans and machines. Providing

such structures for more complex processes might be more valuable than traditional metadata. Since WPS 2.0 does not yet consider aggregate processes an adaptation of SensorML's wiring scheme to the WPS process model might prove useful.

Inheritance among processes is available in both WPS 2.0 and SensorML. The inheritance mechanism in SensorML is quite limited and might be somewhat comparable to WPS implementation profiles, which already define specific types for data exchange. Generic and conceptual levels are not provided. Inheritance in SensorML shall provide more detailed process specifications at lower levels in the hierarchy. Since it allows subtypes to restrict value ranges of inputs and parameters, it breaks with the Liskov substitution principle, i.e. that

> "... objects of a subtype should behave the same as those of the supertype as far as anyone or any program using the supertype can tell" (Liskov and Wing 1994, p. 1811).

As a final remark it should be noted that SensorML mandates that all processes should be GML feature types. At least for purely computational processes this requirement is questionable. The few useful elements that are inherited from GML's *Abstract Feature* (unique identifier, description etc.) could be easily replicated without any reference to GML. Besides the syntactic properties, SensorML processes also inherit the definition of a feature which is supposed to be an *abstraction of real world phenomena*, as declared by ISO (2002, clause 4.11) and OGC (2007b, clause 4.1.26). Computational processes (SensorML examples refer to simple mathematical functions such as Cosine or square root), however, are no abstractions of real world objects but purely theoretical constructs. The property of a feature may only be granted to instances and subtypes of *Abstract Physical Process* since these do refer to real world entities.

The comparison between WPS and SensorML has revealed a significant overlap in the fundamental concepts (basically the input/output-centric view on a process interface and the topology of process chains and workflows) which could be used to frame a joint metadata model that is applicable to all these technologies. A harmonisation of the different existing process models has not yet happened. The abstract process model of WPS already defines some general requirements and covers a broad range of applications. The process model of SensorML can be considered compliant with this abstract process model and WPS 2.0 servers can basically be used to offer SensorML processes. An OGC abstract specification that provides basic terminology and concepts would fill this gap and could aim at a harmonised taxonomy for process models.

Both SensorML's and WPS's process models assume that processes are (or can be subdivided into a set of) stateless I/O-specified functions. The consideration of state could lead to an extended formalisation of processes which is then also applicable to geographic modelling in general. The work of (Zeigler et al. 2000) on modelling and simulation formalisms may provide a conceptual input towards a more general process model which may be used for more complex applications where the consideration

stateful processing is inevitable. The Open Modelling Interface Standard (OpenMI OATC 2010), which has also been adopted as an OGC standard (OGC 2014b), largely deals with this kind of processes but is yet hardly integrated with OGC baseline standards. A governing standard that can be profiled towards stateful and stateless component interfaces may potentially align terminology, technology, and finally improve interoperability in future standards.

Bibliography

A9 (2015). OpenSearch 1.1, Draft 5. Technical specification, A9 and the OpenSearch community. `http://www.opensearch.org/Specifications/OpenSearch/1.1`.

Albrecht, J. (1996). *Universal Analytical GIS Operations*. Dissertation, Universität Vechta.

Almeida, N., Catarino, N., Gutierrez, A., Martinho, F., Rosado, H., Andrade, J., Caumont, H., Gonçalves, P., and Brito, F. (2014). SENSYF: Processing Framework for Sentinel Data. In Soille, P. and Marchetti, P. G., editors, *Proceedings of the 2014 Conference on Big Data from Space (BiDS'14)*, pages 275–278.

Alonso, G., Casati, F., Kuno, H., and Machiraju, V. (2004). *Web Services - Concepts, Architectures and Applications*. Springer, Berlin Heidelberg, Germany.

Bernard, L., Craglia, M., Gould, M., and Kuhn, W. (2005). Towards an sdi research agenda. In Fullerton, K., editor, *The 11th EC GIS & GIS Workshop - ESDI: Setting the Framework - Abstracts Handbook*, pages 147–151.

Bernard, L., Mäs, S., Müller, M., Henzen, C., and Brauner, J. (2013). Scientific geodata infrastructures: challenges, approaches and directions. *International Journal of Digital Earth*, 7(7):613–633.

BMVJ (2009). Gesetz über den Zugang zu digitalen Geodaten (Geodatenzugangsgesetz - GeoZG).

Brauner, J. (2015). *Formalizations for Geooperators - Geoprocessing in Spatial Data Infrastructures*. Dissertation, Technische Universität Dresden.

Brauner, J., Foerster, T., Schaeffer, B., and Baranski, B. (2009). Towards a Research Agenda for Geoprocessing Services. In Seester, M., Bernard, L., and Paehlke, V., editors, *Proceedings of the 12th AGILE International Conference on Geographic Information Science (AGILE 2009)*, Hannover, Germany.

Camara, G., Egenhofer, M., Ferreira, K., Andrade, P., Queiroz, G., Sanchez, A., Jones, J., and Vinhas, L. (2014). Fields as a Generic Data Type for Big Spatial Data. In Duckham, M., Pebesma, E., Stewart, K., and Frank, A., editors, *Geographic Information Science*, volume 8728 of *Lecture Notes in Computer Science*, pages 159–172. Springer International Publishing, Switzerland.

Carzaniga, A., Picco, G. P., and Vigna, G. (1997). Designing Distributed Applications with Mobile Code Paradigms. In *Proceedings of the 1997 International Conference on Software Engineering*, pages 22–32.

Carzaniga, A., Picco, G. P., and Vigna, G. (2007). Is Code Still Moving Around? Looking Back at a Decade of Code Mobility. In *Proceedings of the 29th International Conference on Software Engineering – Companion, 2007*, pages 9–20.

Cheng, T., Haworth, J., and Manley, E. (2012). Advances in geocomputation (1996–2011). *Computers, Environment and Urban Systems*, 36(6):481–487.

Chirigati, F., Shasha, D., and Freire, J. (2013). Reprozip: Using provenance to support computational reproducibility. In *Proceedings of the 5th USENIX Workshop on the Theory and Practice of Provenance, 2013*, pages 1:1–1:4.

Couclelis, H. (1992). People manipulate objects (but cultivate fields): beyond the raster-vector debate in GIS. In Frank, A., Campari, I., and Formentini, U., editors, *International Conference GIS - From Space to Territory: Theories and Methods of Spatio-Temporal Reasoning on Theories and Methods of Spatio-Temporal Reasoning in Geographic Space*, pages 65–77.

Couclelis, H. (1998). Geocomputation – a primer. In Longley, P. A., Brooks, S. M., McDonnell, R., and MacMillan, B., editors, *Geocomputation in Context*, pages 17–29. Wiley, Chichester.

Crnković, I., Sentilles, S., Vulgarakis, A., and Chaudron, M. R. V. (2011). A Classification Framework for Software Component Models. *IEEE Transactions on Software Engineering*, 37(5):593–615.

Davis, P. K. and Anderson, R. H. (2004). Improving the Composability of DoD Models and Simulations. *Journal of Defense Modeling and Simulation*, 1(1):5–17.

de Smith, M. J., Goodchild, M. F., and Longley, P. A. (2007). *Geospatial Analysis. A Comprehensive Guide to Principles, Techniques and Software Tools. Second Edition.* Matador, Winchelsea, UK, 2nd edition.

DiBiase, D., DeMers, M., Johnson, A., Kemp, K., Taylor Luck, A., Plewe, B., and Wentz, E. (2006). *Geographic Science and Information Technology Body of Knowledge*. University Consortium for Geographic Information Science.

Docan, C., Parashar, M., Cummings, J., and Klasky, S. (2011). Moving the code to the data - dynamic code deployment using activespaces. In *Proceedings of the 26th Parallel Distributed Processing Symposium (IPDPS 2011)*, pages 758–769.

EC (2007). 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).

EC (2010a). Commission Regulation (EC) No 976/2009 of 19 October 2009 implementing Directive 2007/2/EC of the European Parliament and of the Council as regards the Network Services.

EC (2010b). Draft Technical Guidance for INSPIRE Coordinate Transformation Services. Version 2.0. INSPIRE Technical Guideline.

ECMA (2013). The JSON Data Interchange Format. Standard, Geneva, Switzerland. Document number ECMA-404.

Economist (2010). The data deluge. *The Ecomonomist – Special report: Managing information*, (8671):13.

Egenhofer, M. and Frank, A. (1992). Object-Oriented Modeling for GIS. *Journal of the Urban and Regional Information Systems Association*, 4(2):3–19.

Egenhofer, M. J. and Franzosa, R. D. (1991). Point-set topological spatial relations. *International Journal of Geographical Information Systems*, 5(2):161–174.

Erl, T. (2007). *SOA – Principles of Service Design*. Prentice Hall, Boston, USA.

ESRI (2015). What is geoprocessing? `http://resources.arcgis.com/en/help/main/10.1/index.html#//002s00000001000000`. Accessed: 2015-04-01.

Farnaghi, M. and Mansourian, A. (2013). Disaster planning using automated composition of semantic OGC web services: A case study in sheltering. *Computers, Environment and Urban Systems*, 41:204–218.

Feuerlicht, G. (2011). Simple metric for assessing quality of service design. In Maximilien, E. M., Rossi, G., Yuan, S.-T., Ludwig, H., and Fantinato, M., editors, *Service-Oriented Computing*, Lecture Notes in Computer Science, pages 133–143. Springer, Berlin Heidelberg, Germany.

Fisher, P. (2006). Algorithm and implementation uncertainty: Any advances? In Fisher, P., editor, *Classics from IJGIS: Twenty years of the International Journal of Geographical Information Science and Systems*, pages 225–228. CRC Press, Boca Raton, FL, USA.

Fitzner, D., Hoffmann, J., and Klien, E. (2011). Functional description of geoprocessing services as conjunctive datalog queries. *Geoinformatica*, 15(1):191–221. Geoinformatica.

Friis-Christensen, A., Ostländer, N., Lutz, M., and Bernard, L. (2007). Designing service architectures for distributed geoprocessing: Challenges and future directions. *Transactions in GIS*, 11(6):799–818.

Gahegan, M. (1999). Guest editorial: What is geocomputation? *Transactions in GIS*, 3(3):203–206.

Gahegan, M. (2015). What is GeoComputation? A history and outline. `http://www.geocomputation.org/what.html`. Accessed: 2015-04-02.

Goodchild, M. F. (2002). Spatial analysis and modeling. In Bossler, J., Jensen, J., McMaster, R., and Rizos, C., editors, *Manual of Geospatial Science and Technology*, pages 482–499. Taylor and Francis, London, UK.

Granell, C., Diaz, L., and Gould, M. (2010). Service-oriented applications for environmental models: Reusable geospatial services. *Environmental Modelling & Software*, 25(2):182–198.

Granell, C., Diaz, L., Schade, S., Ostländer, N., and Huerta, J. (2013). Enhancing integrated environmental modelling by designing resource-oriented interfaces. *Environmental Modelling & Software*, 39:229–246.

Granell, C., Diaz, L., Tamayo, A., and Huerta, J. (2014). Assessment of OGC web processing services for REST principles. *International Journal of Data Mining, Modelling and Management*, 6(4):391–412.

Gray, J. (2009). escience: a transformed scientific method. In Hey, A. J. G., Tansley, S., and Tolle, K. M., editors, *The fourth paradigm: data-intensive scientific discovery*, pages xvii–xxxi. Microsoft Research, Redmond, WA.

Henzen, C., Brauner, J., Müller, M., Henzen, D., and Bernard, L. (2015). Geoprocessing appstore. In Bação, F., Santos, M. Y., and Painho, M., editors, *Proceedings of the 18th AGILE International Conference on Geographic Information Science (AGILE 2015)*, Lisbon, Portugal.

Hill, L. L., Crosier, S. J., Smith, T. R., and Goodchild, M. (2001). A content standard for computational models. *D-Lib Magazine*, 7(6).

Hofer, B. (2015). Uses of online geoprocessing technology in analyses and case studies: a systematic analysis of literature. *International Journal of Digital Earth*, 8(11):901–917.

Howard, M., Payne, S., and Sund, R. (2010). Technical Guidance for the INSPIRE Schema Transformation Network Service. Version 3.0. INSPIRE Technical Guideline.

IETF (2014). The JavaScript Object Notation (JSON) Data Interchange Format. Standard. RFC 7159.

ISO (2002). Geographic information – Reference model. International Standard ISO 19101:2002.

ISO (2004). Geographic information - Simple feature access – Part 1: Common architecture. International Standard ISO 19125-1:2004.

ISO (2005). Geographic information – Rules for application schema. International Standard ISO 19109:2005.

ISO (2006). Geographic information – Services. International Standard ISO 19119:2006.

ISO (2007). Geoinformation – Geography Markup Language (GML). International Standard ISO 19136:2007.

ISO (2008). Information technology – Office Open XML formats. International Standard ISO/IEC 29500.

ISO (2009). Geographic information – Metadata – Part 2: Extensions for imagery and gridded data. International Standard ISO 19115-2:2009.

ISO (2014). Geographic information Metadata Part 1: Fundamentals. International Standard ISO 19115-1:2004.

Kadner, D., Müller, M., Brauner, J., and Bernard, L. (2012). Konzeption eines marktplatzes für den austausch von geoprozessierungsimplementierungen. *gis.SCIENCE*, 25(3):118–124.

Kiehle, C., Greve, K., and Heier, C. (2007). Requirements for next generation spatial data infrastructures-standardized web based geoprocessing and web service orchestration. *Transactions in GIS*, 11(6):819–834.

Kuhn, W. (2012). Core concepts of spatial information for transdisciplinary research. *International Journal of Geographical Information Science*, 26(12):2267–2276.

Ledoux, H. and Gold, C. (2006). A Voronoi-Based Map Algebra. In Riedl, A., Kainz, W., and Elmes, G., editors, *Progress in Spatial Data Handling*, pages 117–131. Springer, Berlin Heidelberg, Germany.

Liskov, B. H. and Wing, J. M. (1994). A behavioral notion of subtyping. *ACM Transactions on Programming Languages and Systems*, 16(6):1811–1841.

Longley, P. A. (1998). Foundations. In Longley, P. A., Brooks, S. M., McDonnell, R., and BillMacMillan, editors, *Geocomputation - A Primer*, pages 3–15. Chichester, Wiley.

Longley, P. A., Goodchild, M. F., Maguire, D. J., and Rhind, D. W. (2011). *Geographic Information Systems and Science*. Wiley, Hoboken, NJ, USA.

Lopez-Pellicer, F. J., Renteria-Agualimpia, W., Béjar, R., Muro-Medrano, P. R., and Zarazaga-Soria, F. J. (2012). Availability of the OGC geoprocessing standard: March 2011 reality check. *Computers & Geosciences*, 47:13–19.

Lutz, M. (2007). Ontology-Based Descriptions for Semantic Discovery and Composition of Geoprocessing Services. *Geoinformatica*, 11(1):1–36. Geoinformatica.

Lutz, M., Riedemann, C., and Probst, F. (2003). A classification framework for approaches to achieving semantic interoperability between gi web services. In Kuhn, W., Worboys, M. F., and Timpf, S., editors, *Spatial Information Theory. Foundations of Geographic Information Science*, Lecture Notes in Computer Science, pages 186–203. Springer, Berlin Heidelberg, Germany.

Marshall, J., Downs, R., and Samadi, S. (2010). Relevance of software reuse in building advanced scientific data processing systems. *Earth Science Informatics*, 3(1):95–100.

Maué, P., Michels, H., and Roth, M. (2012). Injecting semantic annotations into geospatial web service descriptions. *Semantic Web*, 3(4):385–395.

Mennis, J., Viger, R., and Tomlin, C. D. (2005). Cubic Map Algebra Functions for Spatio-Temporal Analysis. *Cartography and Geographic Information Science*, 32(1):17–32.

Meyer, B. (1992). Applying "Design by Contract". *Computer*, 25(10):40–51.

Mineter, M. J., Jarvis, C. H., and Dowers, S. (2003). From stand-alone programs towards grid-aware services and components: a case study in agricultural modelling with interpolated climate data. *Environmental Modelling & Software*, 18(4):379–391.

Müller, M. (2013). Hierarchical process profiles for interoperable geoprocessing functions. In Vandenbroucke, D., Boucher, B., and Crompvoets, J., editors, *Proceedings of the 16th AGILE International Conference on Geographic Information Science (AGILE 2013)*, Leuven, Belgium.

Müller, M. (2015). Hierarchical profiling of geoprocessing services. *Computers and Geosciences*, 82:68–77.

Müller, M., Bernard, L., and Brauner, J. (2010). Moving code in spatial data infrastructures - web service based deployment of geoprocessing algorithms. *Transactions in GIS*, 14(S1):101–118.

Müller, M., Bernard, L., and Kadner, D. (2013). Moving code – sharing geoprocessing logic on the web. *ISPRS Journal of Photogrammetry and Remote Sensing*, 83:193–203.

Müller, M., Wiemann, S., and Grafe, B. (2012). A framework for building multi-representation layers from openstreetmap data. In *Proceedings of the 15th ICA Workshop on Generalisation and Multiple Representation*, Istanbul, Turkey.

Nebert, D. D. (2004). *Developing Spatial Data Infrastructures: The SDI Cookbook*.

NIST (2014). *Draft NIST Big Data Interoperability Framework: Volume 6, Reference Architecture*. National Institure of Standards and Technology, US Department of Commerce.

OATC (2010). OpenMI Document Series: OpenMI Standard 2 Specification for the OpenMI (Version 2.0). Standard. The OpenMI Association Technical Committee (OATC).

OGC (1999). OpenGIS Simple Feature Specification for SQL, Revision 1.1. OGC Standard. OGC document 99-049.

OGC (2007a). OpenGIS Catalogue Services Specification. OGC Standard. OGC document 07-006r1.

OGC (2007b). OpenGIS Geography Markup Language (GML) Encoding Standard. OGC Standard. OGC document 07-036.

OGC (2007c). OpenGIS Web Processing Service, Version 1.0.0. OGC Standard. OGC document 05-007r7.

OGC (2009). Semantic annotations in OGC standards. OGC Discussion Paper. OGC document 08-167r1.

OGC (2010). OGC Web Services Common Standard. OGC Standard. OGC document 06-121r9.

OGC (2011). OGC SWE Common Data Model Encoding Standard. OGC Standard. OGC document 08-094r1.

OGC (2012). OGC WCS 2.0 Interface Standard – Core: Corrigendum. OGC Standard. OGC document 09-110r4.

OGC (2013). OGC Abstract Specification Geographic information – Observations and measurements. OGC Standard. OGC document 10-004r3.

OGC (2014a). Filter Encoding 2.0 Encoding Standard – With Corrigendum. OGC Standard. OGC document 09-026r2.

OGC (2014b). OGC Open Modelling Interface Standard. OGC document 11-014r3.

OGC (2014c). OGC SensorML: Model and XML Encoding Standard. OGC Standard. OGC document 12-000.

OGC (2014d). OGC Web Feature Service 2.0 Interface Standard – With Corrigendum. OGC Standard. OGC document 09-025r2.

OGC (2015a). OGC WPS 2.0 Interface Standard. OGC Standard. OGC document 14-065.

OGC (2015b). Sensor Web Enablement (SWE). Web page. Accessed 2015-04-09.

OGC (2015c). Web Coverage Service. OGC Standard. Accessed 2015-04-09.

OMG (2005). Unified Modeling Language: Superstructure. Version 2.0.

Openshaw, S. (2000). Geocomputation. In Openshaw, S. and Abrahart, R. J., editors, *GeoComputation*, pages 1–31. Taylor and Francis, London, UK.

Pautasso, C., Zimmermann, O., and Leymann, F. (2008). Restful Web Services vs. "Big" Web Services: Making the Right Architectural Decision. In *Proceedings of the 17th International Conference on World Wide Web (WWW '08)*, pages 805–814. ACM.

Pierce, M. E., Fox, G. C., Ma, Y., and Wang, J. (2010). Cloud computing and spatial cyberinfrastructure. Report, Indiana University.

Pullar, D. (2001). MapScript: A Map Algebra Programming Language Incorporating Neighborhood Analysis. *GeoInformatica*, 5(2):145–163.

Schaeffer, B. (2008). Towards a transactional web processing service (wps-t). In Pebesma, E., Bishr, M., and Bartoschek, T., editors, *Sixth Geographic Information Days (GI-Days 2008)*, volume 32. IfGIPrints.

Schmitz, O., Karssenberg, D., de Jong, K., de Kok, J.-L., and de Jong, S. M. (2013). Map algebra and model algebra for integrated model building. *Environmental Modelling & Software*, 48:113–128.

Schubert, C. (2011). Bereitstellung dynamischer Web Processing Services zur Verarbeitung von Geodaten mit OSGi.

Soille, P. and Marchetti, P. G. (2014). Preface. In *Proceedings of the 2014 Conference on Big Data from Space (BiDS'14) in Frascati, Italy*. Publications Office of the European Union, Luxembourg.

Szyperski, C. and Pfister, C. (1997). Component-oriented programming: Wcop'96 workshop report. In Mühlhäuser, M., editor, *Special Issues in Object-Oriented Programming - Workshop Reader of the lOth European Conference on Object-Oriented Programming*, pages 127–130. Dpunkt Verlag.

Takeyama, M. (1997). Building spatial models within GIS through Geo-Algebra. *Transactions in GIS*, 2(3):245–256.

Tanenbaum, A. S. and Steen, M. v. (2007). *Distributed Systems - Principles and Paradigms (Second Edition)*. Pearson Prentice Hall, Upper Saddle River, NJ, USA.

Tomlin, D. (1990). *Geographic Information Systems and Cartographic Modelling*. Prentice-Hall, Englewood Cliffs.

van der Aalst, W., ter Hofstede, A., Kiepuszewski, B., and Barros, A. (2003). Workflow patterns. *Distributed and Parallel Databases*, 14(1):5–51.

Vanhellemont, Q. and Ruddick, K. (2014). Landsat-8 as a precursor to Sentinel-2: Observations of human impacts in coastal waters. In *Proceedings of the Sentinel 2 for Science Workshop*, Frascati, Italy.

W3C (2006). Reference Model for Service Oriented Architecture 1.0. W3C Standard.

W3C (2007). Web Services Description Language (WSDL) Version 2.0. W3C Standard.

Wade, T. and Sommer, S. (2006). *A to Z GIS: An Illustrated Dictionary of Geographic Information Systems*. ESRI Press, Redlands, CA, USA.

Wang, S., Anselin, L., Bhaduri, B., Crosby, C., Goodchild, M. F., Liu, Y., and Nyerges, T. L. (2013). CyberGIS Software: A Synthetic Review and Integration Roadmap. *International Journal of Geographical Information Science*, 27(11):2122–2145.

Wesselung, C. G., Karssenberg, D.-J., Burrough, P. A., and Van Deursen, W. P. A. (1996). Integrating dynamic environmental models in GIS: The development of a dynamic modelling language. *Transactions in GIS*, 1(1):40–48.

Worboys, M. and Duckham, M. (2004). *GIS: A Computing Perspective. Second Edition.* CRC Press, Boca Raton, FL, USA.

Yang, C., Raskin, R., Goodchild, M., and Gahegan, M. (2010). Geospatial cyberinfrastructure: Past, present and future. *Geospatial Cyberinfrastructure*, 34(4):264–277. Computers, Environment and Urban Systems.

Yue, P., Di, L., Yang, W., Yu, G., and Zhao, P. (2007). Semantics-based automatic composition of geospatial web service chains. *Computers and Geosciences*, 33(5):649–665.

Zaharia, R., Vasiliu, L., Hoffman, J., and Klien, E. (2008). Semantic execution meets geospatial web services: A pilot application. *Transactions in GIS*, 12:59–73.

Zeigler, B. P., Praehofer, H., and Kim, T. G. (2000). *Theory of Modeling and Simulation, Second Edition.* Academic Press, San Diego, CA, USA.