# Flood Modeling in Spatial Data and Grid Infrastructures

Vom Promotionsausschuss der
Technischen Universität Hamburg-Harburg
zur Erlangung des akademischen Grades
Doktor der Naturwissenschaften (Dr. rer. nat.)
genehmigte Dissertation

von
Dipl.-Inf. Stefan Kurzbach

aus
Gifhorn

2014

1. Gutachter: Prof. Dr. rer. nat. habil. Ralf Möller

2. Gutachter: Prof. Dr. Matthew Smith

Datum der mündlichen Prüfung: 24. Oktober 2013

# Vorwort

Liebe Leserinnen und Leser der Hamburger Wasserbauschriften

Seit dem Beginn des Informationszeitalters werden Computer zur Modellierung hydrodynamischer Prozesse eingesetzt. Auch in der Wasserwirtschaft setzte die Nutzung entsprechender Modelle etwas später ein. Mit steigender Komplexität der mathematischen Abbildung physikalischer Prozesse, die häufig lediglich durch empirische Modelle beschrieben werden können, und nicht zuletzt mit den gesteigerten Anforderungen an die räumliche und zeitliche Auflösung der Modelle steigen insbesondere auch die Anforderungen an Speicherkapazität und Rechenleistung der Rechner. Zwar werden auch einzelne Hochleistungsrechner immer leistungsfähiger und können komplexe Vorgänge immer schneller berechnen, jedoch ist der Einsatz von Hoch- und Höchstleistungsrechnern teuer und bislang meist auf ausgewählte Projekte beschränkt. Aus diesem Grund haben eine Reihe nationaler und internationaler Vorhaben (wie die D-Grid-Initiative oder die European Grid Infrastructure EGI) sich zum Ziel gesetzt, Industriepartner und Forschungseinrichtungen in einer Grid-Infrastruktur zu vernetzen und ihnen so Rechenleistung zur Verfügung zu stellen. Hier setzt die vorliegende Arbeit an und zeigt Rahmenbedingungen und Umsetzungskonzepte für die Modellierung von Hochwasser in einer Grid-Infrastruktur auf.

In diesem Zusammenhang ist es dann erforderlich, fundamentale Probleme der Informatik zu lösen, die entstehen, wenn eine Aufgabe nebenläufig in einem verteilten System ohne zentrale Kontrolle bearbeitet wird. Dazu gehören unter anderem i) die räumliche Zerlegung eines Modells in Teilmodelle, ii) die parallele Berechnung von Teilmodellen und iii) das Zusammenfügen der Teilergebnisse zu einem Ganzen und zudem auch iv) die logische Zerlegung des Lösungsverfahrens in mehrere automatisierbare Schritte, die parallel oder sequentiell in der Grid-Infrastruktur ausgeführt werden können. Hierzu entwickelt Herr Kurzbach grundsätzliche Wege zur Umsetzung, die dann in der Arbeit konzeptionell auf der Grundlage von bestehenden Standards und einer speziellen Software, der Grid-Middleware, bis in die Ebene der Ausführung und Ausführbarkeit konkretisiert werden. Für die Umsetzung der Methodik wird das Problem der Diskretisierung eines Strömungsmodells aus digitalen Geländedaten unter der Berücksichtigung von Standards aus Geodaten-Infrastrukturen als Anwendungsfall genommen.

Mit Bezug auf die Hochwassermodellierung bedeutet die Ausführung eines numerischen Modells die Durchführung einer konkreten numerischen Berechnung, welches immer auch die Erstellung einer räumlichen Diskretisierung für das numerische Modell zeitlich voraussetzt. Eine solche Diskretisierung ist die räumlich und ggf. auch zeitlich aufgelöste Abbildung eines Gewässers inklusive der Sohle und des an ein Gewässer anschließenden Geländes. Die hierfür erforderlichen hochauflösenden Geländeinformationen werden seit einiger Zeit – auf ähnliche Weise wie hochauflösende Luftbilder – von vielen Gebieten der Erde aufgenommen und stehen somit grundsätzlich zur Verfügung, wobei die Verwaltung und Verarbeitung dieser enormen Datenmengen die verantwortlichen – zumeist Behörden – vor große Probleme stellt.

Ansätze für die Verwaltung von Gelände- und anderen Geodaten werden seit einigen Jahren entwickelt und strukturiert sowie standardisiert. Diese Entwicklungen zur standardisierten Verwaltung von Geodaten lief parallel zur der o. g. Entwicklung von Grid-Infrastrukturen. Hierbei wird das Netzwerk von Geodaten-Anbietern und -Nutzern unter dem Schlagwort "Geodaten-Infrastruktur" zusammengefasst.

Die vorliegende Arbeit von Herrn Kurzbach zeigt auf der beschriebenen Grundlage einen möglichen Weg für das zukünftige Zusammenspiel von Geodaten-Infrastrukturen und Grid-Infrastrukturen auf. Als Beispielanwendung hat Herr Kurzbach mit der Hochwassermodellierung zwar eine exemplarische Fragestellung aus dem Wasserbau gewählt, die entwickelte Methodik wird dabei aber so allgemein gehalten, dass sie sich leicht auf andere geobasierte Fragestellungen übertragen lässt.

Hamburg, 05.03.2014

Peter Fröhle

# Summary

Spatial data and grid infrastructures manage distributed geographic data, computing, and storage resources from different organizations without centralized control. However, these two infrastructures have evolved separately and use different open standards. This thesis makes a contribution to the implementation of such open standards from spatial data and grid infrastructures in the field of flood modeling. In a first step, and for the first time ever, the process of flood modeling by two-dimensional hydrodynamic simulation — flow model discretization, flood simulation, and results evaluation — is formalized as a sequence of geoprocessing tasks. These geoprocessing tasks are then implemented as geoprocessing grid services. To achieve this, two standards commonly used in spatial data and grid infrastructures, the Web Processing Service (WPS) and the Web Services Resource Framework (WSRF), are harmonized. This harmonization results in a generic procedure for the creation of geoprocessing grid services called *Grid-WPS*.

This procedure is evaluated by two prototypical implementations of flood modeling tasks. The first prototype, a *flow model discretization service*, employs a new strategy for the parallel creation of large, unstructured, two-dimensional computational meshes based on a high-resolution digital elevation model. Through the application of grid technology, it is shown how large volumes of geographic data from a spatial data infrastructure can efficiently be processed in a distributed computing environment using a geoprocessing grid workflow. The second prototype, a *flood simulation service*, further demonstrates how to execute a large-scale hydrodynamic simulation on multiple distributed clusters in a grid infrastructure with application of a two-level parallel domain decomposition approach.

Together, the flow model discretization service and flood simulation service provide an efficient technology for mapping flood hazards. By adhering to the presented Grid-WPS procedure, the developed services are enabled to coexist in both infrastructures. In this way, they can make use of the provided resources and can also easily be reused in the context of other grid applications or geoprocessing workflows.

# Contents

## Appendix

# 1. Introduction

In recent history, Europe has been overshadowed by numerous flood disasters and their devastating consequences for the environment, economy, and citizens. Climatologists anticipate even more frequent and extreme precipitation events leading to extreme floods [Bar07; BRL07]. In 2007, the European Commission acted on this issue and passed the "Floods Directive" [Eur07b]. Its scope is the evaluation of flood hazard and flood risk, the creation of flood hazard maps, and the preparation of flood risk management plans in all European countries until 2015. The large number of flood maps to be created and flood scenarios to be simulated puts enormous pressure on national authorities.

## 1.1. Motivation

According to the Floods Directive, flood hazard maps have to be created for flood events of medium statistical probability, with a water level or discharge that is expected to occur about every 100 years on average, as well as extreme floods and events with higher recurrence periods. Those flood maps are required to display the inundated areas and flow properties, i. e. water depths and flow velocities. They are ideally based on two-dimensional, time-dependent hydrodynamic models that take into account in detail the surface topography, bathymetry, roughness, and vegetation of an area [MAA09]. The main data source for both topography and bathymetry is a digital elevation model, i. e. a three-dimensional representation of the earth's surface. Data for modern digital elevation models is usually acquired by shipborne, airborne, or spaceborne remote sensing technology (e. g. sonar / laser altimetry or the TanDEM-X satellite mission). Advances in this field lead to increasing precision in the representation of terrain surfaces resulting in data volumes for digital elevation models in the order of terabytes to petabytes.

Nowadays, high-resolution digital elevation models are frequently available making them applicable to flood modeling by two-dimensional hydrodynamic simulation. Discretization of the computational mesh, which is a major step in creating a flood model, and the hydrodynamic simulation are time- and storage-consuming processes carried out by research organizations, technical and scientific federal authorities, or

engineering companies. The use of high-resolution elevation data for the creation of large-scale, two-dimensional flood models, the numerical simulation of such models, and subsequent result evaluations create the need for sophisticated hardware, processing techniques, and large-scale, inter-organizational management of digital elevation and simulation data [Vero6; Vero7].

The geographic data required for flood model creation, foremost the digital elevation data, often spans multiple administrative regions and data providers. Spatial data infrastructures strive to bring together data providers and consumers enabling the use of geographic data across organizational boundaries using well-known standards [Nebo4]. The European INSPIRE (Infrastructure for Spatial Information in Europe) directive [Euro7a] enforces that all countries of the EU provide standardized web services for geographic data simplifying access, visualization, processing, and sharing inside and beyond national boundaries for environmental and political purposes. Spatial data infrastructures also define standards for the processing of geographic data over the web using geoprocessing services. Such geoprocessing services may aid in the automation of tasks related to the processing of digital elevation data for flood model creation. Nevertheless, the lack of software support of these standards in flood modeling clearly inhibits an efficient use of geoprocessing services in this field, as they have to deal with complex problems and huge data volumes. This creates the requirement for a sophisticated processing infrastructure which can be applied for geoprocessing.

A grid is a kind of distributed system using standard protocols and interfaces for efficient, collaborative problem solving. Grid infrastructures use grid middleware for managing computing and data storage resources from different organizations without centralized control. Grid techniques are not only about submitting jobs to a computing cluster, but also provide an integrative environment for domain-specific applications. Grid technology is capable of providing standardized access to the computational power and storage capacities required for flood model creation and flood simulation at low cost and on demand. Moreover, basing an application for this purpose on a stack of grid services, grid middleware, and computing and storage resources promises improvements in one or more of the following aspects:

1. High performance and reliability
2. Data storage and availability
3. Service-orientation and reusability
4. Standardization and interoperability
5. Automation and workflows
6. Security

## 1.2. Problems and Research Objectives

Flood modeling is an example of an application that could combine the advantages of processing and data management techniques applied in both spatial data and grid infrastructures. The goal of this thesis is to support interoperability in the flood modeling process by implementing open geographic data and processing standards and to develop appropriate distributed geoprocessing services. The focus is on providing a procedure that allows the implementation of standard geoprocessing services for flood modeling in a grid infrastructure. However, the problem of enabling existing software to take advantage of grid technology ("gridification") is all but well-defined from a formal perspective. Gridification can only be accomplished considering all technical layers of a software system and following a suitable procedure that takes all relevant standards and available technologies into account. For this purpose, the flood modeling process is to be regarded as a sequence of geoprocessing tasks that can then be automated as a workflow of geoprocessing services. These geoprocessing services are to be designed as grid services. Geoprocessing grid services for flood modeling are to be developed as basic, reusable components suitable for service composition while maintaining compatibility with existing standards from spatial data and grid infrastructures. As there exist different standards for geoprocessing services and grid services, in particular the Web Processing Service (WPS) standard and the Web Services Resource Framework (WSRF), a solution to reconcile these standards is sought.

This thesis evaluates the prospects of employing spatial data and grid infrastructures for the large-scale modeling of flood events by prototypical implementation of two uses cases. This is done at the example of original data and a real model of the Elbe estuary.

1. A *service for flow model creation* from digital elevation data is to be developed. The service requires a novel methodology and parallel algorithm for flow model creation in the grid. This use case is to demonstrate the application of grid workflows for automated geoprocessing of large amounts of digital elevation data.

2. An existing two-dimensional hydrodynamic model for flood simulation of the Elbe estuary is to be grid-enabled by developing a *flood simulation service*. High performance in the grid can only be achieved if the underlying numerical model can be executed in parallel. The existing model used in the flood model of the Elbe estuary does not yet have that capability, so a non-intrusive parallelization will be attempted. This use case is to show problems entailed by the gridification of tightly-coupled numerical models and presents approaches to their solution.

## 1.3. Results and Dissemination Activities

This work originates from a research project funded by the German Federal Ministry of Education and Research (Bundesministerium für Bildung und Forschung, BMBF) as part of the German D-Grid Initiative, Geodateninfrastruktur-Grid (GDI-Grid, grant number G07012F), which investigated possibilities of a "spatial data infrastructure grid" in Germany. The results from a number of peer-reviewed publications from the GDI-Grid project have provided the basis of this thesis. The following list shows all preceding related work in chronological order.

1. First experiences with a grid service for discretization of a flow network from terrain data and a parallel process have been shown in

   > S. Kurzbach and E. Pasche. "A 3d Terrain Discretization Grid Service for Hydrodynamic Modeling". In: *Proceedings of the 8th International Conference on Hydroinformatics (HEIC 2009)*. Concepción, Chile, January 12-16. Ed. by O. Parra, J. Zambrano, A. Stehr. 2009.

2. Related details about the interoperability of the Web Processing Service and the Web Services Resource Framework have also been published in the OGC community as part of

   > B. Baranski, A. Shaon, A. Woolf, S. Kurzbach. *OWS-6 WPS Grid Processing Profile Engineering Report. OGC 09-041r1*. Open Geospatial Consortium, Inc. (OGC), 2009

3. The use cases of flood modeling inside a spatial data infrastructure grid have been presented in

   > S. Kurzbach, S. Braune, C. Grimm, E. Pasche. "Hochwasser-Modellierung im Geodateninfrastruktur-Grid". In: *Angewandte Geoinformatik 2009. Beiträge zum 21. AGIT-Symposium Salzburg*. Ed. by J. Strobl, T. Blaschke, G. Griesebner. Wichmann, 2009, pp. 372–377.

4. A service-oriented perspective on flood modeling and synergies from using grid computing for spatial data infrastructures has been published in a journal article initiated at the 12th AGILE International Conference on Geographic Information Science (2009) in a workshop on *Grid Technologies for Geospatial Applications* [LK+09]:

   > S. Kurzbach, E. Pasche, S. Lanig, A. Zipf. "Benefits of Grid Computing for Flood Modeling in Service-Oriented Spatial Data Infrastructures". In: *GIS.Science* (3 2009), pp. 89–97.

5. The aspect of flood simulation in a spatial data infrastructure grid has been highlighted in

   S. Kurzbach, S. Braune, E. Pasche, M. Smith. "Operative Hochwasservorhersage-Dienste im Geodateninfrastruktur-Grid". In: *Angewandte Geoinformatik 2010. Beiträge zum 22. AGIT-Symposium Salzburg*. Ed. by J. Strobl, T. Blaschke, G. Griesebner. Wichmann, 2010, pp. 881–886.

6. Finally, the parallel meshing process and mesh generation results for the tidal river Elbe have been presented in

   S. Kurzbach and N. Schrage. "Automatic Mesh Generation for 2D Hydrodynamic Flood Models from High-Resolution Digital Elevation Data". In: *Proceedings of the 10th Conference on Hydroinformatics (HIC 2012). Understanding Changing Climate and Environment and Finding Solutions*. July 14-18, Hamburg, Germany. Ed. by R. Hinkelmann, H. Nasermoaddeli, M. H. Liong, D. Savic, P. Fröhle, K.-F. Daemrich. 2012.

## 1.4. Overview

The thesis is structured as follows: An introduction to the relevant grid technologies is given in Chapter 2. The application domain of flood modeling by two-dimensional hydrodynamic simulation is described in Chapter 3. Chapter 4 formalizes the flood modeling process as a sequence of geoprocessing tasks. Afterwards, a generic procedure to develop geoprocessing services in the grid is presented in Chapter 5. Results of the prototypical implementation of the two flood modeling use cases are shown in the following two chapters (Chapter 6 and Chapter 7). Finally, in Chapter 8, conclusions are drawn from these use cases regarding the gridification of the flood modeling process using the developed procedure and the profits promised by using grid technology.

# 2. Grid Computing

Grid computing is a shape of distributed computing originating from an analogy to the electric power grid. However, the term *grid* is only one of many buzzwords hovering in IT industry today such as *service-oriented architecture*, *cloud computing*, *utility computing*, *software-as-a-service*, and many others. The true nature of grid computing is more complicated and the result of a general trend in IT towards service-oriented systems. Foster and Tuecke [FT05, p. 29] gave an informal definition of the term that helps to understand the meaning of grid computing for this thesis:

> [...] *grid* is a big-picture term used to describe solutions relating to the flexible use of distributed resources for a variety of applications — as well as a term that emphasizes the importance of standards to interoperability.

A more formal definition will be made in Section 2.1 together with an introduction to the fundamentals of grid computing. Current methods, concepts, and standards for service-oriented grid computing are presented in Section 2.2. Finally, Section 2.3 outlines the properties of grid computing infrastructures, employed software, and related technologies with application in this thesis.

## 2.1. Fundamentals

Due to the lack of a standard definition and a multitude of different opinions, there is a common misconception of what grid computing means and how it is different from other forms of distributed computing. In this section, after giving a short overview of the history of grid computing, the term *grid* will be defined. Finally, the development of applications for a grid will be explained.

### 2.1.1. Origin of the Term

In the nineteen-twenties and nineteen-thirties the electric power grid revolutionized the economic infrastructure of the industrialized nations. The supply of electricity over long-distance transmission lines greatly increased the efficiency of national industries

and made former luxury products like electrical lighting a commodity. The prices for electricity went down rapidly with the introduction of transmission networks connecting local power plants. A new standards-based infrastructure was formed that would bring together electricity providers and consumers independent of their geographic locations [Hug83, pp. 293-295].

An electric power grid uses standard interfaces, e. g. for electricity generation and power transmission, and works even across country borders. Electricity is readily available nearly everywhere. Potential service outages for the end users are minimized by redundancies of power sources and connections in the transmission network. Most importantly, electrical appliances can just be "plugged-in" to the system. The power grid shows an exemplary high quality of service which lends itself to application in other domains. Computer science is one of those areas that could possibly adopt the image of a "plug-and-run" infrastructure for computing power and storage.

### Computational Grids

The need for computer-based problem-solving occurs in many disciplines: Scientific and engineering questions are answered using complex mathematical and numerical simulation models. Examples can be found in industry, medical diagnostics, and the military. Many sophisticated computations are extremely time-consuming and data-intensive so they can only be performed on powerful computing resources with large data storage areas. Today, most challenging problems in these domains are solved using high-performance computing (HPC) environments. This term is often attributed to applications that have been designed for tightly coupled parallel execution on a supercomputer or computing cluster [FK99a].

Foster and Kesselman [FK99a] have first applied the metaphor of the electric power grid to computing environments and postulated a similarity between computational and electric power. They expected the new paradigm to revolutionize the way people think about how computers are used and computations are performed: "A computational Grid is a hardware and software infrastructure that provides dependable, consistent, pervasive, and inexpensive access to high-end computational capabilities". This early attempt at defining grid computing did not take into account that computing resources could be provided by many geographically distributed organizations. As an increasing number of computing centers were established and scientific problems became more demanding, there was the need to extend the meaning of grid computing.

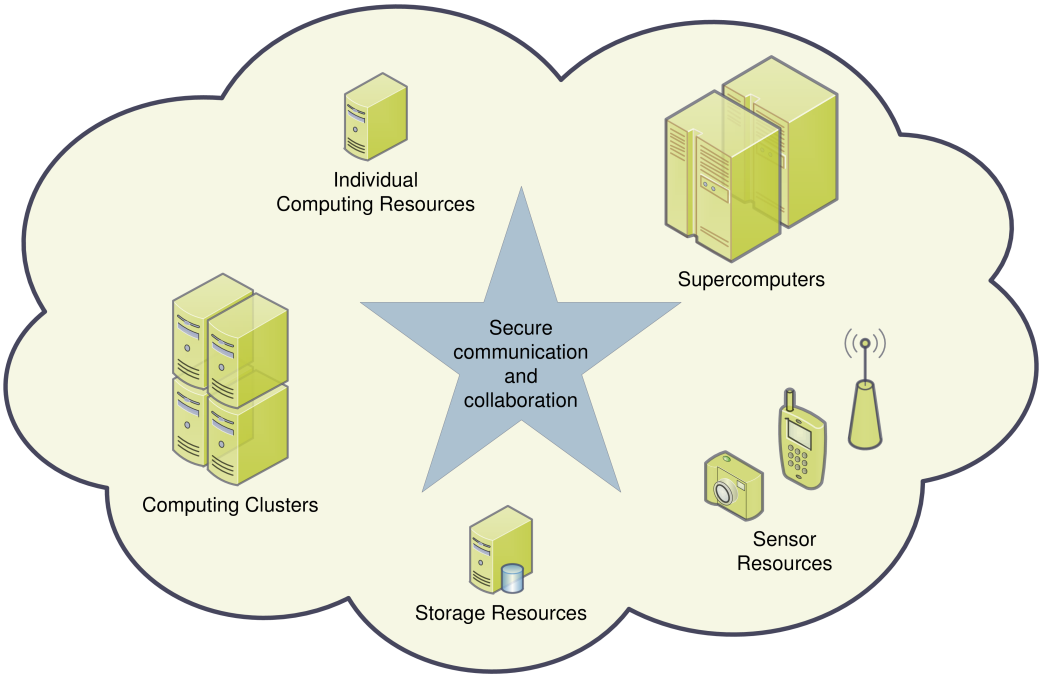



**Figure 2.1.:** A grid enables scientific collaboration using geographically distributed resources provided by different organizations in a secured way.

### 2.1.2. Definition

A widely recognized definition of the term *grid*, which will also be applied in this thesis, has been given by Foster [Fos02, pp. 2-3]:

> [...] a grid is a system that: 1) coordinates resources that are not subject to centralized control [...] 2) using standard, open, general-purpose protocols and interfaces [...] 3) to deliver non-trivial qualities of service.

Qualities of service that a grid may exhibit include, but are not limited to, on-demand access to distributed computational facilities, security, availability, reliability, high performance, and high throughput. In this sense, a grid is a special form of distributed system that uses many loosely coupled computing, storage, and other resources for scientific problem solving. Indeed, grid computing is not limited to traditional scientific problems, but that is where a grid can prove to be very powerful.

### Computing Clusters and Supercomputers

Both computing clusters and supercomputers are locally managed collections of computing resources that help to solve computationally expensive problems. Supercomputers are, actually, a monolithic resource featuring a high number of *processors* with global memory. The collection of computing resources in a cluster or, respectively, the number of processors in a supercomputer, will be called *computing nodes*. In a cluster, these nodes each have their own internal memory and disk space. Set up in close proximity to each other, they are connected by Gigabit Ethernet or, preferably, a specially designed network (*InfiniBand*).

The problem to be solved is typically split into smaller tasks that are then solved in parallel to speed up the computation. The parallel processes are distributed among the computing nodes that constitute the cluster or supercomputer. A problem is called *inherently parallel* if these tasks are independent of each other, i. e. all processes can execute their part of the algorithm independently, requiring little or no communication with other processes. Inherently parallel problems are suited for execution on a loosely coupled network of computing resources regardless of communication delays. Hydrodynamic simulation, as a counterexample, is a tightly coupled HPC application. Tightly coupled parallel programs require many synchronization points where messages have to be exchanged between the individual processes. For this reason, tightly coupled applications depend on a fast network connection with lower delays and higher bandwidth for inter-process communication. The fastest communication rates can be achieved on monolithic supercomputers.

### The Relationship Between a Grid and a Computing Cluster

Just like a computing cluster, a grid consists of computing resources. However, in contrast to a cluster, a grid is loosely coupled, transparent, decentralized, and more heterogeneous. The computing resources are usually geographically distributed and diverse regarding their hardware and software environments. A grid can be seen as a kind of "virtual supercomputer" — a single, yet massive computing resource — composed of individual computers, clusters, data bases, and remote sensors delivering streams of data (see Figure 2.1). The aforementioned definition of a grid by Foster et al. [FKT03], which is assumed in this thesis, implies that many systems that are called *grids*, although they may expose some properties of a grid, are often not more than local cluster management or load sharing systems. In contrast to the definition, they are tightly coupled and lack the attributes of a distributed system.

For HPC applications this implies that the immediate benefit of using a grid over using a single cluster is (1) being able to choose from a list of available clusters and

(2) running a number of independent HPC applications at the same time on possibly different clusters. However, as will be shown in Chapter 7, HPC applications, such as hydrodynamic simulation, can not generally be executed across multiple clusters in a grid without suffering a loss of performance. In order to make use of a grid, the tight coupling of the application will have to be relaxed.

#### Opportunistic, On-Demand, and Volunteer Computing

Many home computers run in an idle state much of their time. These computers may volunteer to provide their computing power, on-demand, for specific research projects. Such projects make use of a distributed computing infrastructure that allows the execution of small work units on the idle client computers. Volunteer computing originates from the BOINC (Berkeley Open Infrastructure for Network Computing) initiative, which became famous through the SETI@home project[1] and the World Community Grid (WCG)[2] featuring, for example, the Human Proteome Folding project. The WCG is, according to their own statement, the world's largest public computational grid. Opportunistic computing refers to the use of computer resources once they become available and is thus related to volunteer computing.

The BOINC framework sets up a client-server infrastructure for distributed computing. Because of this central control, BOINC is not a middleware for grid computing in the sense of this thesis. Nevertheless, opportunistic and volunteer computing are paradigms that may well be implemented on a grid infrastructure [KC+08]. On-demand computing was also used, for example, to create a distributed infrastructure for image processing in a campus grid [Cat09].

### 2.1.3. Developing a Grid Application

While there is a lot of literature on grid computing, little can be found on the development of applications that run efficiently on grid systems. Nevertheless this thesis proves that it requires considerable effort to develop a complex application that can benefit from all aspects of the grid. The problems of distribution, coordination, parallelization, standardization, and integration into a grid infrastructure all have to be addressed. In addition to domain-specific knowledge, a grid application developer needs insight into parallel algorithms and concurrency, hardware and software aspects of computing environments, Internet technologies, and, last but not least, service-oriented design.

[1] `https://boinc.berkeley.edu`, BOINC was actually released under the GPL in 2004 based on the experiences from SETI@home, which started in 1999.

[2] `http://www.worldcommunitygrid.org`

### Examples of Grid Applications

A number of national and international grid projects have evolved that strive to establish grid infrastructures for a great diversity of applications. There is a well-maintained list of projects on the web featuring 19 international and more than 40 national grid initiatives, as well as a few field-specific grids[1]. These projects come from application areas including astronomy, biology, climate modeling, economics, geospatial applications, high-energy physics, humanities, logistics, material sciences, medical applications, molecular simulation, multimedia, neurosciences, and seismology — to name just a few.

One can see from this list that most real grids have scientific applications as their main use case. This leads to the assumption that high performance and throughput (regarding processing, storage, and network) is the essential quality of service they expect from a grid. There are a few notable exceptions, such as the humanities, and projects focusing on inter-organizational aspects of scientific collaboration. This gives reason to believe that grid computing is indeed a technology that, at least in its fundamental principles, will prevail.

### Characterization of Grid Applications

A grid application is a complex piece of software that has to cope with problems on different layers of the grid computing stack: application, collective, resource, connective, and fabric (see Figure 2.2). Tightly coupled parallel applications are said to feature a fine-grained parallelism, whereas loosely coupled applications have a coarse-grained parallelism. Practically, however, the distinction between coarse-grained and fine-grained parallelism is gradual and has to be assessed by the developer.

The processing work of a grid application is usually done as part of one or more *grid jobs*. Clusters accept jobs for execution on their managed resources. A cluster has a batch system, or local resource management system (LRMS), in order to allocate computing nodes to jobs. A grid job can be submitted to run either on only one node, or it distributes a parallel process across several nodes. In a grid job, the selected nodes can either be local to one cluster, or span multiple clusters and even single computers. In the latter case, parallel execution in the grid becomes more powerful than batch job submission in a cluster because resources can now be geographically distributed. This approach is sometimes called Cluster-of-Clusters (CoC) or metacomputing.

The individual processes of an inherently parallel application can be executed as independent jobs in a grid, whereas an application with any kind of coupling between

[1] `http://www.gridcafe.org`

processes needs an additional coordination mechanism. A HPC application either needs a single job with multiple processes or multiple jobs with a single process each. Launching multiple jobs at the same time requires some sort of co-scheduling mechanism. Another approach, which is used in high-throughput scenarios, is to launch a single job that executes multiple processes sequentially in order to avoid the overhead of scheduling.

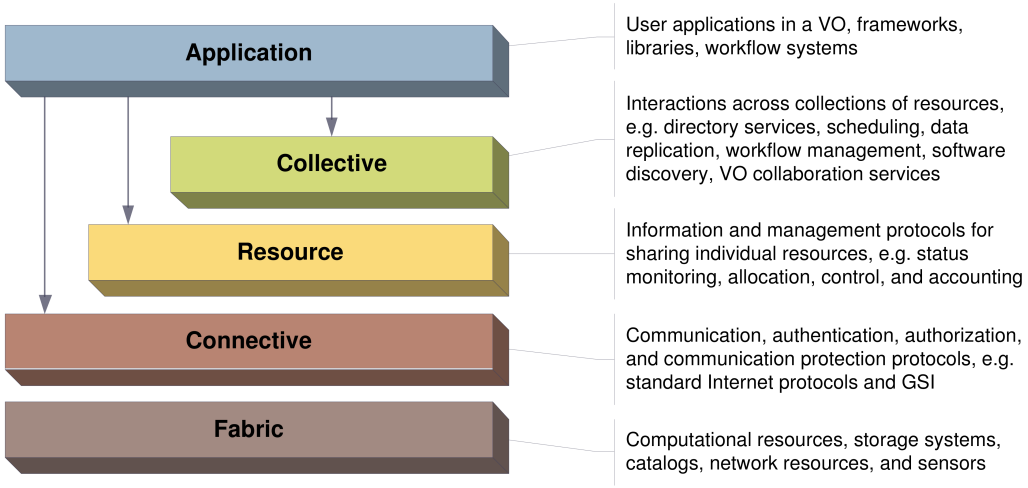



**Figure 2.2.:** Layered grid architecture and protocols provided by each layer according to Foster et al. [FKT03].

Different abstractions and programming tools are required in the development of distributed grid applications. Jha et al. [JC+10] identified that there is a significant gap between grid application programming patterns and the abstractions delivered by current programming tools[1]. A typical problem when developing a grid application is that the low-level "Fabric" and "Connective" layers actually comprise another complex, layered system of hardware, operating system, and management software. For example, in many current grid infrastructures, e. g. the German D-Grid infrastructure, the provided grid resources are often computing clusters. A computing cluster is a very different kind of computing resource than an individual computer and comes with additional HPC capabilities that a grid application developer can use, but it also entails a higher complexity, especially if the combined computing power of several computing cluster resources is to be used in an HPC application. Not all challenges coming

[1]The upcoming monograph "S. Jha and D. S. Katz. *Abstractions for Distributed Applications and Systems: A Computational Science Perspective*. Vol. 79. Wiley Series on Parallel and Distributed Computing. Wiley, 2012 (to be published)" [JK12] is going to deal with this topic in detail.

from the heterogeneity of the resources and network on the "Fabric" layer and their available communication mechanisms on the "Connective" layer are solved in current grid middleware and programming tools. In fact, this is what makes the development of distributed grid applications so difficult.

A different view on grid architecture will now be shown that helps developers to identify the gaps they might encounter when designing a HPC application for execution across clusters in a computing grid or grid-enabling an existing legacy HPC code.

### A Practical View on Grid Architecture

The view on grid architecture developed here is what developers should ideally build their grid applications on. They see the grid as a collection of many loosely coupled resources. Typical computing grids provide access to their computing resources via interfaces that are delivered through *grid middleware* functions for job submission. A grid job is broken down to a batch job submission on a cluster or the execution of a program on an individual computer. Grid middleware tries to hide the complexity from heterogeneous hardware and software of the provided computing resources. Prevalent grid middleware distributions will be presented in Section 2.3. Submitting individual grid jobs with a single process is a well-supported scenario, but it is not always clear how to submit an HPC job with multiple processes to a cluster or how to submit multiple jobs simultaneously and set up the coordination between processes.

At the lowest level, a grid application needs to be deployed and executed on the computing resource, i. e. an executable file has to be built for the local computer architecture of a cluster node, supercomputer, or desktop computer. Given the heterogeneity in hardware and software of grid resources, it is likely that a grid application will have to be tailored to each of the environments that form a part of the grid. Aloisio et al. [ACE06] presented a solution to the problem of application portability in a heterogeneous grid environment. They introduced a software design pattern called "Grid Executable Catalog". It is based on a metadata repository which serves pre-staged executable files for different platforms. In this way, the existence of heterogeneous platform architectures is made transparent to the user.

The submission of jobs in a computational grid entails knowledge about the internal mechanics of jobs in a cluster, which will be explained in the following. The executable file of the job has to reside in a shared file system accessible to all computing nodes in the cluster. A designated node, the *head node*, serves as single point of access to all other cluster nodes. Job handling inside a cluster is done by interaction with grid middleware services on the cluster head node. Designated services on the head node are used for job submission, file transfers, and security. Jobs are submitted together

with a job description. Once a job has been submitted to a cluster, the job is waiting to be scheduled in a *queue*. Eventually, some nodes matching the job description will be selected for processing this job. A batch script is then executed on a *root node* of the selected cluster nodes. This script then calls the executable in a specific way to run the application on all selected nodes and establishes the communication links between them, depending on the communication protocol implemented in the application. However, some job description parameters contain information specific to cluster resources. These depend on the type of batch system used in the cluster and are not standardized as part of the job specification. In particular, a job with multiple processes needs to specify the number of nodes and may also request a number of cores per node or an amount of memory per node. Additionally, most clusters manage differently named queues for long and short jobs or those requiring many or few nodes. When submitting a grid job, the correct queue for a specific cluster has to be known beforehand.

Parallel processing often requires sending messages between the individual processes. Application developers can rely on existing software for this purpose, e. g. a library implementing the Message Passing Interface (MPI) or the Parallel Virtual Machine (PVM) library. MPI is the de-facto standard for HPC software distribution across nodes on a computing cluster and is suited for tightly-coupled applications with fine-grained parallelism. As such it relies on a low-latency, high-throughput network connection. An MPI application is always compiled and linked against the specific MPI libraries installed on the cluster. Otherwise, portability issues may arise. In terms of an executable file, which is submitted to a cluster, this means to provide a compiled version for all architectures that it is going to be executed on. Even though the MPI standard is, in principal, suited to run programs in a heterogeneous environment, the different implementations of MPI are generally not compatible with each other, which implies that an application built on one cluster will (probably) not work on another cluster in the same computing grid. Algorithms executed across multiple clusters in a grid even need to send messages both between cluster nodes (intra-cluster communication) and between nodes in different clusters (inter-cluster communication). This requirement makes MPI a bad choice for communication across clusters as the sites may rely on different vendor-supplied, optimized MPI installations. However, different initiatives have tried to make MPI capable of connecting multiple clusters in a computing grid.

MPICH-G2 [KTF03][1], MPIg [MM+08], and MPICH-VMI [PJ04] are grid-enabled MPI implementations. Their advantage over other implementations is that they can run MPI jobs across multiple, geographically distributed resources. The resources of several clusters can thus be combined. MPICH-G2 has been developed using Globus Toolkit services [KTF03]. It uses a vendor-supplied MPI for intra-cluster communication and TCP/IP for inter-cluster communication, and thus introduces a compatibility layer

[1] `http://www3.niu.edu/mpi`, based on Globus Toolkit 2

between different MPI implementations. Application developers need to consider, however, that inter-cluster communication has a much higher latency (hundreds of milliseconds) than intra-cluster communication (tens of microseconds), and bandwidth can vary significantly when transferring data over the web using the TCP/IP protocol [PJ04]. MPICH-G2 was applied to perform large-scale blood flow simulations on the TeraGrid [DKK05]. Pant and Jafri [PJ04] developed a similar solution, MPICH-VMI, for communication in cluster-based grids. Another implementation for this purpose, MPIg, was given by Manos et al. [MM+08]. Another MPI-parallel hydrodynamic simulation was ported to a grid of desktop computers using ObjectWeb ProActive[1] [CC+06]. This open source platform contains components for building public and private grids and clouds. Even though the effort showed the feasibility of executing an MPI application in a grid, many problems arising in grids, like security and heterogeneity, were not addressed. Neither could the authors demonstrate the scalability of their hydrodynamic model over a slow communication link.

Another library for the development of parallel applications is Open Multi-Processing (OpenMP). The OpenMP library is a programming interface for the development of shared-memory parallel applications on multi-core processors. As such it is suited to add a layer of parallelism on the level of a single computing resource in the grid.

MPI and OpenMP are mostly used for tightly-coupled parallel applications. A multitude of other paradigms and tools enable the development of applications with coarse-grained parallelism, such as web or grid services, distributed objects, workflow systems, and multi-agent systems. At the "Connective" layer, all of them use message passing for communication, but wrapped up in higher-level programming abstractions.

#### Multilevel Parallelism

Other current efforts, e. g. [GZ10], strive to extend HPC architectures towards coupling MPI-parallelism with the even more fine-grained parallelism provided by multi-core processor systems (OpenMP) or graphics hardware (GPU clusters). OpenMP facilitates shared-memory parallelization on a single computing node or individual computer. The performance of this technology has been investigated in the context of computational fluid dynamics [Hoe01]. It is used in most current numerical model implementations (see Section 7.2).

GPU systems typically employ the stream processing paradigm, a form of single instruction multiple data (SIMD) parallel processing. Even though this type of research goes into a different direction, a number of similarities to the integration of clusters

[1] http://proactive.inria.fr/

and grids can be observed. The authors state that "a major challenge of the multi-GPU parallelization is an efficient implementation of the data exchange process." Data has to be transported from GPU to CPU memory on one computer over a network interface to another computer, there again from CPU to GPU memory. The solution given in [GZ10; Mic09] and other publications is to overlap communication and computation as much as possible.

A different kind of multilevel parallelism for numerical simulation across several clusters was shown in [DK04]. The authors designed a hierarchical, three-level (MPI / MPI / OpenMP) algorithm for a stochastic, high-order spectral/hp element CFD method and demonstrated the efficiency of their approach at the direct numerical simulation of turbulent flow past a cylinder.

In summary, a HPC application can be designed for grid computing, but this requires changes in the software design, algorithm, and communication mechanisms. As previously mentioned, there is an obvious gap between grid application programming abstractions and available programming tools. The key to HPC grid application development is the unification of different tools and paradigms into a multi-layered software architecture, which possibly uses several levels of parallelism with different granularity, thereby integrating computing resources on different layers horizontally and vertically, e. g. workflows, grid services, and message passing.

Coarse-grained parallelism does not (as much) depend on a fast network connection, based on the assumption that sufficient work is done between synchronization points. The flood simulation service in Chapter 7 shows an example of such a multilevel design using several computing clusters in a grid and attempts to provide a "big-picture" solution to the integration problem. Further focus is set to a service-oriented view on grid computing, which will be clarified in the following section.

## 2.2. Service-Oriented Grids

As a design principle for software development service-orientation provides an abstract view on else very complex systems. The advantages of service-oriented architecture (SOA) have also been deployed in most current grid infrastructures. Today's service-oriented scientific grids are de-facto based on the Open Grid Services Architecture (OGSA), which has to be considered in the development of any grid application.

After an introduction to SOA using web services the concept of a *grid service* is explained. The last subsection deals with the orchestration of grid services as *grid workflows*.

### 2.2.1. Service-Oriented Architecture

SOA proposes an abstract concept for a software architecture that focuses on offering, searching, and using services as atomic units. Some widely recognized fundamental principles of SOA are loose coupling, abstraction, reusability, statelessness, discoverability, and composability [Melo8].

The actors involved in a SOA take on the role of either service provider or service consumer. Consumers usually locate available services in a service registry to avoid a tight coupling between services. A standards-based SOA has the advantage of being platform- and implementation-independent facilitating the integration of distributed services from possibly different providers in a common software application.

#### Web Services

SOA can be realized using a range of service technologies. Web services, however, are the most commonly used technology. The XML-based specifications publicized by World Wide Web Consortium (W3C), Simple Object Access Protocol (SOAP), and Web Services Description Language (WSDL), serve as the fundamental standards of interoperable SOAs implemented using web services.

The WSDL standard describes a web service interface on both an abstract level of functionality and on the level of technical details required for calling service functions. These functions are called operations and are defined by a number of typed messages exchanged between client and service. All types used inside messages have to be defined in an XML schema document. A service endpoint specifies message structure, encoding (e. g. a style of SOAP), and physical location (i. e. Internet address) of the service. WSDL also provides a way to express service faults.

As web services always rely on the exchange of XML documents sent over Internet protocols, they introduce a noticeable overhead in communication and are generally slower than other implementations of a software architecture. Cooper and Huang [CHo8] evaluated an alternative to MPI based on web services. Their results confirm that SOAP-style web services are not suited to sending many small messages. This has to be accounted for when making design decisions in the development of SOA, in particular when designing a grid service architecture aiming at high performance. The next subsection deals with the implementation of a service-oriented grid using web services.

### 2.2.2. Grid Services

Grid computing as it has been shaped by Foster et al. [FK+02] is built on SOA. The Open Grid Forum (OGF) has issued the definition of the OGSA as the conceptional basis of many service-oriented grids and formulated a concrete specification of OGSA using web service standards in form of the Web Services Resource Framework (WSRF).

#### Open Grid Services Architecture

In OGSA the OGF defines a common, standard, and open architecture for grid-based applications. It aims at identifying the most important services that are commonly found in a grid system and standardizing their interfaces: execution management, resource management, workflow, security, and data management, among others. Key requirements of OGSA are interoperability, the management of potentially transient services and service state, dynamic lifetime, and service discoverability [FK+03]. Foster et al. also state that "all components of the environment are virtualized. [...] It is the virtualization of grid services that underpins the ability for seamlessly mapping common service semantic behavior onto native platform facilities." The virtualization aspect is dealt with in Section 2.3.

#### The Web Services Resource Framework

WSRF is based on the W3C web service standards to enhance a grid computing environment with the advantages of SOA and stateful web services. It is a concrete set of standards by the Organization for the Advancement of Structured Information Standards (OASIS) enabling the implementation of OGSA. WSRF-based stateful web services integrated in a grid computing infrastructure are called grid services. However, the WSRF specification does not make any reference to grid computing, rendering it a generic framework for web service development.

WSRF introduces new specifications in the form WS-*. Most importantly, the concept of a web service resource (WS-Resource) represents the combination of a web service with a stateful resource. In this context state refers to some information associated with the invocation of a web service similar to object instances in object-oriented programming. Resources are used in a web service request according to the WS-Addressing and WS-ResourceProperties specifications so they can be shared among any number of services. WS-ResourceLifetime defines means of destroying a WS-Resource and monitoring its lifetime. WS-Notification allows state changes in a WS-Resource to be pushed to interested consumers in either a peer-to-peer or a brokered communication style.

Java implementations of the WSRF are provided as part of the Globus Toolkit 4 and UNICORE 6 grid middlewares (see Subsection 2.3.2), and the IBM WebSphere Application Server (starting from version 6.1). Other projects implementing WSRF specifications are Apache Muse (Java), WSRF::Lite (Perl), and WSRF.NET (Microsoft .NET).

### 2.2.3. Grid Workflows

A possible use case for SOA is the implementation of workflows. Workflows can be used as a framework for the development of distributed applications by service orchestration. A workflow consists of a process dependence graph of individual activities with data and control flowing from one activity to another. This graph has exactly one start and one end point. Independent activities may be executed concurrently.

#### Scientific Workflows

Scientific workflows are the application of workflows for scientific endeavors. The automated or semi-automated computational solution to scientific problems results in the need for interconnected tools and large data quantities. Scientific and business workflows evolved in parallel resulting in different workflow tools for each purpose. Two examples of scientific workflow systems are Kepler[1] and Taverna Workbench[2].

#### BPEL4Grid Workflow Engine

The orchestration of grid services needs to face the challenges resulting from the stateful nature of a grid service. An enterprise workflow system capable of handling grid services is the BPEL4Grid Engine. It has been applied and improved in several grid computing research projects [SS+09; DSF08; FM08]. This workflow engine has originally been developed at Marburg University, Germany. It is based on the open source software ActiveBPEL 5.0 by ActiveEndpoints[3] and has been extended to support grid-related features such as WSRF and grid security.

It turns out that using an enterprise workflow system and language for scientific process automation complicates the workflow design process. This problem has been addressed in two supplementary tools for graphical workflow design assisting the definition

[1] http://kepler-project.org
[2] http://www.taverna.org.uk
[3] ActiveEndoints does not provide the download link of this version anymore

of a domain workflow, the Visual Grid Orchestrator (ViGO) and the SimpleBPEL Composer.[1]

Unfortunately, the extension of the BPEL standard for WSRF-based grid services and the adaptations of the ActiveBPEL workflow engine render some of the advantages of using BPEL obsolete. Grid workflows, although specified in BPEL, can now only be executed in this specific implementation of a workflow engine. Another approach using appropriate BPEL design patterns has been shown in Hasselbring [Has10]. Their grid services can be orchestrated by any BPEL-compliant workflow engine. Advanced WSRF service state and security mechanisms are provided by a *Workflow Management Service*, which is a grid service itself. The implementation has been made as part of the *BIS-Grid engine*[2]. Yet, the workflow engine is currently incompatible with many grid services because the implementation is based on UNICORE 6, which does not support GSI.

## 2.3. Grid Infrastructure

An infrastructure constitutes the underlaying foundation on which to develop and support inter-organizational structures and processes. Providers and consumers coming from different organizations need a working infrastructure for supply and use of services. In software architecture, a grid infrastructure is a means of creating a horizontal integration layer in an application built on resources governed by distributed responsibilities.

A generic software called *grid middleware* is installed in a grid to provide standardized and secure access to resources in a grid infrastructure. Virtualization and virtual organizations help to decouple components and to integrate a diversity of heterogeneous resources and organizations. Foster et al. [FK+03] define virtualization as a key component in grid infrastructures.

Forms of virtualization in grid infrastructures and their advantages are discussed in Subsection 2.3.1. Subsequently, grid middleware components used in this thesis are described (Subsection 2.3.2). Finally, light is shed on the security aspect and how it affects gridification (Subsection 2.3.3).

[1] All software tools mentioned can be downloaded at `http://mage.uni-marburg.de`.

[2] BIS-Grid (Business Information Systems in Grid) is a project in the context of the German D-Grid Initiative. The worflow engine is available for download at `http://bis-grid.sourceforge.net`.

### 2.3.1. Virtualization

The requirement that computer systems have to be able to dynamically adapt to changes in the environment or changing demands is becoming more and more important. Systems have to be easily assembled, extended, or reconfigured to an organization's needs. Virtualization is an attempt to provide an abstract layer that logically isolates resources of a system, possibly hiding details about their nature, with the goals of dynamic adaptability and easy management.

#### Virtual Organizations

A number of real organizations or individuals pursuing a common goal may pool in a Virtual Organization (VO) in order to more efficiently share competencies, resources, and services. Grid computing enforces the concept of a Virtual Organization for flexible, secure, coordinated sharing of human and computer resources according to a set of rules. Existing grid middleware solutions such as Globus Toolkit incorporate Virtual Organizations as an integral part into their security infrastructure (see Subsection 2.3.3 on page 24). Membership in a Virtual Organization may be required before grid access can be authorized (e. g. in the German D-Grid infrastructure).

#### Virtual Machines

A virtual machine (VM) is a virtual computer behaving like a real machine. A VM may be able to run a single program or even a full operating system depending on the level of virtualization (hardware or software). It either runs in a native execution environment or, more commonly, is hosted on top of a real computer's operating system. One physical computer has the capability to host any number of VMs, only limited by its hardware capabilities.

The main advantages of using VMs are the support of heterogeneous physical hosts and virtual (guest) computers, the pooling of physical resources, dynamic and on-demand scaling of an infrastructure by adding and removal of VMs, as well as easy administration. Its disadvantage is that the virtualization layer generally incurs a performance loss.

#### Cloud Computing

Cloud computing is a business model that makes use of virtualization to provide one of three kinds of services: Software-as-a-Service (SaaS), Platform-as-a-Service (PaaS),

or Infrastructure-as-a-Service (IaaS). These services are either commercially available to the public (e. g. Amazon's Elastic Compute Cloud EC2 or the Google App Engine) or for internal organizational use. The difference to grid computing lies mostly in the fact that clouds feature a central resource management and do not support resource sharing between organizations.

Grids and clouds are no contradictory technologies [FZ+08]. In fact, the combination of both is an intriguing idea: A grid could offer cloud services to utilize free resources in a Virtual Organization or cloud resources could be integrated into a grid infrastructure to serve peak demands. Nevertheless, the two fields are emerging separately, and there is still confusion about the terms and how they are different. The open source Nimbus project[1] is an example of a framework that allows to build an IaaS-cloud based on grid middleware services. Another project, Globus Provision[2], can automatically deploy and configure an instant grid backed by a computing cluster on Amazon EC2.

### 2.3.2. Grid Middleware

A grid middleware is the server-side "glue" to form a grid infrastructure from hardware and software resources of different providers. Grid middleware also contains a client-side interface to standard grid services, e. g. to submit computing jobs into the grid, to distribute the required data, and to perform monitoring tasks. Additional domain- or application-specific services can be installed in the grid middleware and is accessed via grid standards to provide high-level integration of software tools into the grid. A grid middleware also provides security mechanisms to protect VO resources from unauthorized use as well as for confidential communication.

There are essentially two different kinds of grid middleware: computing middleware and storage middleware. With the emergence of WSRF, grid services have found their way into the middleware frameworks, in particular Globus Toolkit and UNICORE [FC+05]. The German D-Grid Initiative integrates both middleware distributions into its reference architecture (see Figure 2.3).

#### Globus Toolkit

Globus Toolkit is a service-oriented grid middleware that has had a lot of influence on existing grid standards like Grid Security Infrastructure (GSI) and OGSA. As of version 4, Globus Toolkit is the reference implementation of OGSA and WSRF [Fos05]. A number of grid projects, such as the German D-Grid initiative, use Globus Toolkit 4

[1] http://www.nimbusproject.org
[2] http://www.globus.org/provision

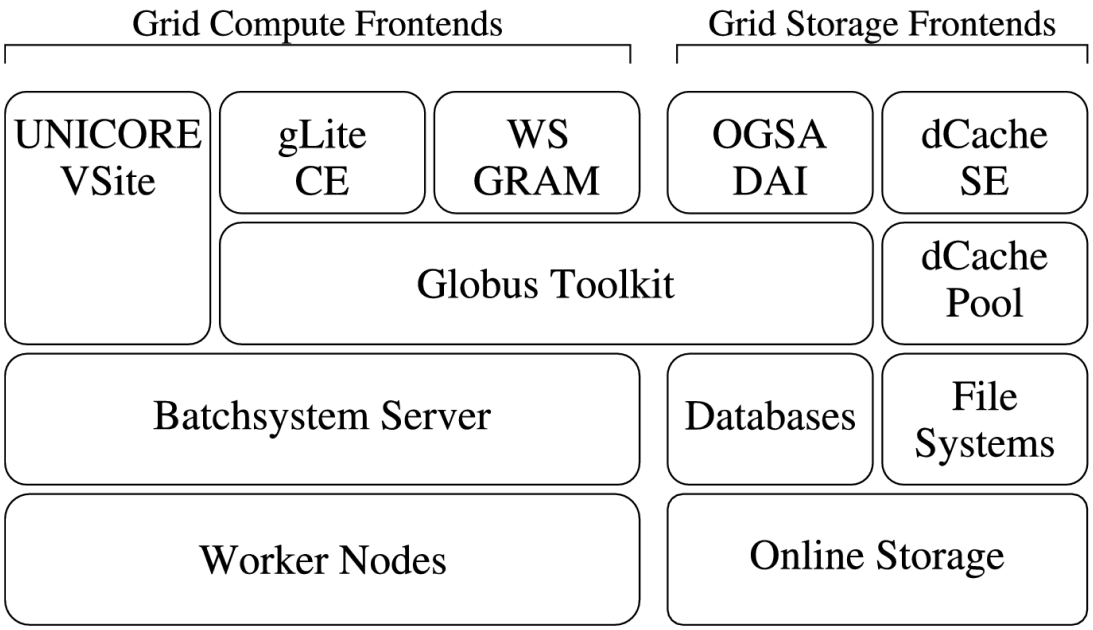



**Figure 2.3.:** Software stack of the D-Grid reference architecture [FW11].

as their middleware. Other middlewares with functionality similar to Globus Toolkit include LCG/gLite and UNICORE. The latter, in version 6, also implements the WSRF specification. Data management in Globus Toolkit is characterized by techniques such as the Replication and Location Service (RLS), Reliable File Transfer (RFT), and GridFTP, a file transfer protocol tailored to high-performance networks and grid security.

The most recent version of Globus Toolkit, version 5, does not yet include the WSRF components from version 4, but instead provides its own interfaces, mainly for performance reasons. A rewrite of the web service framework was scheduled for first release in 2009/2010 under the name *CRUX*, but the current status of the project is unclear (as of 2012). This development could endanger the further establishment of WSRF as a standard for grid computing and has thus raised the attention of many international grid projects building on Globus Toolkit.

### 2.3.3. Grid Security

Security is a major requirement in many grid computing environments [FK+98]. The requirements include authentication, authorization, confidential communication, message integrity, and delegation of credentials across institutional boundaries. These requirements are met by the GSI.

### GSI

It is common practice to use digitally signed certificates for the purpose of identifying a grid user. GSI is a specification originating from the Globus Toolkit project using certificates based on public key cryptography [Gloo5].

#### Authentication

Authentication ensures that the user's identity can be verified. This may require the integration with a local security system (e. g. Kerberos). An authentication solution should have a single sign-on characteristic, i. e. users only need to authenticate once ("log on") when working with multiple grid resources. The owner's grid certificate, containing his identity, and a root certificate create a chain of trust from the user to the Certificate Authority (CA) that initially issued the user's certificate. This chain can be checked by mutual authentication for all secure actions in the grid.

#### Authorization

Authorization is the decision if a user (or other grid entity) is allowed to perform an operation. It typically involves checking a user's credentials for membership in a Virtual Organization. A mapping from grid certificate names to local Virtual Organization user accounts is the default authorization mechanism. Such a mapping can either be global or per-service. Other authorization policies may be used. For example, access could be restricted to a list of specific users or to the creator of a WS-Resource. In general, a client should also authorize the server when requesting a service, i. e. the client makes sure the server certificate really belongs to the contacted host.

#### Confidential Communication and Message Integrity

Confidentiality means that transmitted information cannot be read by unauthorized parties. Access is restricted by encryption of the message content. Message integrity ensures that messages, which have been sent and received, cannot be changed by a third party. Any modifications of the message in transit can be detected by the receiver.

### Delegation of Credentials

With the intent to delegate his rights to another grid resource, a user can create a so-called proxy certificate based on his grid certificate that has a limited lifetime and is either stored locally for simple access or transmitted to a service designated to this purpose (e. g. a Globus Toolkit Delegation Service). This service receives a proxy certificate and may be contacted by other grid services, e. g. if they are part of a workflow, to authenticate the user that originally started the request. All that is needed is the endpoint reference of the WS-Resource containing the delegated credential. The support of proxy certificates in UNICORE 6 is experimental.

The open source software MyProxy maintains a repository of security credentials. Certificates can be stored in a MyProxy repository protected by user name and password. Proxy certificates are obtained by providing this passphrase information or via an external authorization mechanism including LDAP, Kerberos, or Virtual Organization membership and roles (using Virtual Organization Membership Service, VOMS). In addition, MyProxy supports automatic or manual credential renewal, as to prevent failure of a long-running activity in the grid due to an expired proxy credential, for example. Finally, the MyProxy server can act as a CA issuing user credentials based on pre-configured CA certificates.

## Conclusion

Grid computing is a paradigm in line with terms such as cloud computing, virtualization, and *everything-as-a-service*. Originating from the desire to provide ad-hoc computing power for resource-intensive applications, the grid has grown to be a metaphor of an integration layer in dynamic software architectures for multi-organizational resource sharing. These can be hardware resources such as computers, computing clusters, or storage systems, but also grid applications exist that integrate a diversity of sensors and software applications to enable collaboration between research institutes spanning different areas of science. Nevertheless, the typical grid user today needs to know many details about the underlying computing environments. As a consequence, the developer of a grid application has to face many problems if he is to design a system that conforms to standards and fulfills the *run-everywhere* character of a grid, hiding the complexity of application distribution from the user.

# 3. Flood Modeling by Two-Dimensional Hydrodynamic Simulation

In the last years, severe flood events with high economic damage and a high loss of lives are becoming ever more frequent. The August 2002 floods along the Elbe and its tributaries caused a total damage of more than 11 billion USD, the flash floods in Pakistan in July 2010 more than 9 billion USD, and the May 2010 China floods of many major rivers even more than 18 billion USD. The total economic damage per year has been increasing noticeably (compare Figure 3.1). Governments have reacted and started national flood management programs. These require a large number of flood models to be created that help to assess the risk of flooding and the effects of mitigation measures. In this context, the simulation of flood flow patterns (free surface hydrodynamics) plays a major role.

The European Floods Directive [Euro7b] defines the term *flood* as "the temporary covering by water of land not normally covered by water". A *flood model* has the purpose of demonstrating the spatial or temporal course of a specific flood event. Numerical or computational flood models use computers to simulate the spatial or temporal flood-related phenomena by discrete approximation of a mathematical model. The process of *flood modeling* means setting up, applying, and analyzing the results of a flood model.

Floods can be distinguished by their type as *fluvial floods* (arising from a watercourse), *pluvial floods* (arising from surface runoff before reaching a drainage system), *coastal floods* (arising from the sea), and *groundwater floods* (arising from subsurface flow). All have different causes, that may occur jointly and combine their effects, such as prolonged rainfall, cloudbursts (leading to flash floods), snow melt, failure of flood protection infrastructure (e. g. dike breaks), unfavorable wind and tidal conditions (e. g. storm surges), tsunamis, or insufficient urban drainage capacities. Existing flood models typically cannot take into account all these factors at a time, so the most appropriate model needs to be selected for a given scenario.

This chapter begins with an overview of the process of flood hazard map creation (Section 3.1). The mathematical foundation, applicability, and technical details of hydrodynamic flood models are then explained in Section 3.2.

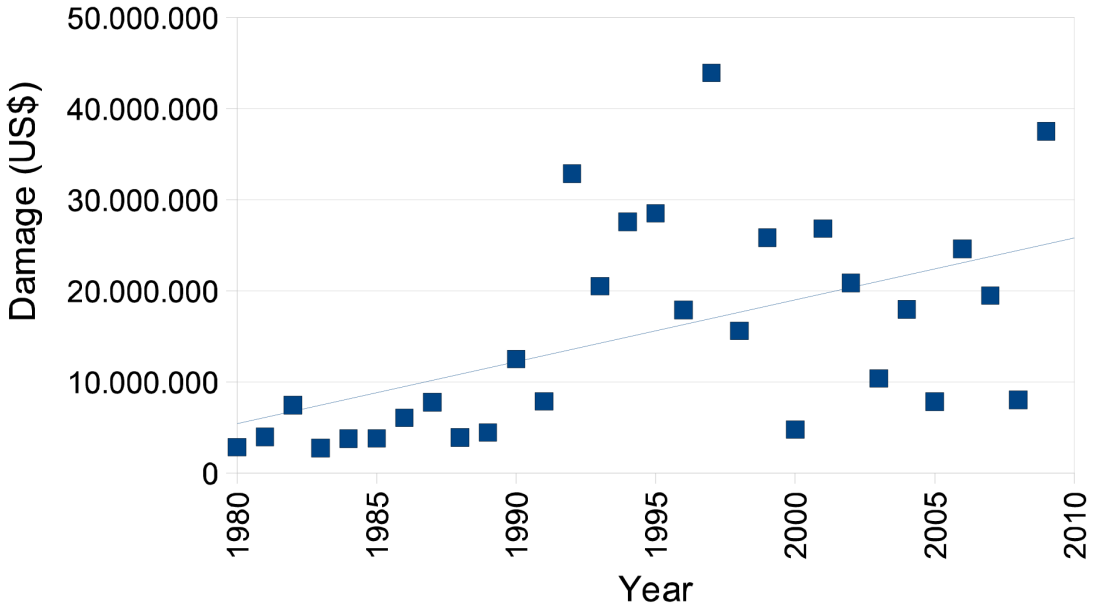



**Figure 3.1.:** Linear regression analysis of the total economic damage caused by floods from 1980 to 2011 (source of data: EM-DAT – The OFDA/CRED International Disaster Database. Université Catholique de Louvain. Brussels, Belgium. `http://www.emdat.be`).

## 3.1. Flood Mapping

*Flood mapping* is an application of numerical flood models, which has the primary goal of creating a flood hazard map (here also termed *flood map*, in short). Flood map creation is a complex engineering process that requires geographic information system (GIS) expertise as well as in-depth knowledge of the causes of flooding and appropriate flood models [MAA09].

### 3.1.1. Flood Hazard Maps

Flood hazard maps serve as a decision support tool in flood risk management and help to spatially identify and illustrate the objective hazard of flooding [KN+10]. Recent legal efforts in European countries and the EU strive to standardize flood maps and the flood map creation process in order to ensure high-quality results.

### Definition of Flood Hazard

Merz and Thieken [MT04] define *flood hazard* as the *exceedance probability* of potentially damaging flood situations in a given area and within a specific period of time combined with the *intensity* (magnitude) of the flood. A flood event can be categorized by its average recurrence interval (or return period) over a longer period of time, i. e. an estimate of the time interval between events of a certain intensity or, equally, the inverse of the probability that the event will be exceeded in any one year. The intensity of a flood is measured by maximum (peak) river discharge, or water stage. For example, there is a $\frac{1}{100}$ chance every year that the discharge of the Rhine river at Lobith, the Netherlands, exceeds 12000 $\frac{m^3}{s}$, i. e. this flood event has a return period of 100 years [Gel99].

### European Law

European legislative has recently issued the "Directive on the assessment and management of flood risks" [Eur07b]. It dictates that all member states of the European Union have to create flood risk management plans in three consecutive steps. As a *first step*, the areas with significant flood risks shall be identified (by the end of 2011). The *second step* consists in the preparation of detailed flood hazard maps and flood risk maps for these areas (by the end of 2013). The flood and risk maps serve as the basis of the *third step*, which is the establishment of appropriate flood risk management plans (by the end of 2015).

The Floods Directive requires maps to be created for flood events of low, medium, and high probability (corresponding to return periods of $\gg 100, \geq 100,$ and $\approx 10$ years) as well as extreme events. Extreme events may include scenarios like dike breaks or the joint occurrence of a low probability river flood with a storm surge in a coastal region. Generally, current climate change impacts have to be represented in the flood and risk maps while future trends only have to be considered in the flood risk management plans. Nevertheless, this outdates existing flood maps in many cases creating the need for additional calculations.

### Map Components

The basic information that a flood map needs to provide are the return period of the flood shown, the location of the map, legend, north arrow, scale, the responsible editors, and the publication date. As a minimum requirement, a flood hazard map must display the inundated areas, i. e. the flood extent and water depths. Flow velocities need to be

shown only "where appropriate" [Euro7b, Article 6(4)]. The European Exchange Circle on Flood Mapping (EXCIMAP) [Exco7] proposes a possible application of flow velocity maps the technical planning of flood defense measures or structures in the area.

A representative flood hazard map conforming to the Floods Directive has been published by the German Working Group on Water Issues (Bund/Länder-Arbeitsgemeinschaft Wasser, LAWA). It can be seen in Figure 3.2 and shows the delineation of the flood extent as well as a classification of the water depth in shades of blue. In flooded areas, it overlays the local flow velocity and direction in the form of green, orange, and red arrows. The LAWA suggests the additional presentation of flood defense structures, e. g. dikes or flood protection walls, in coastal areas [Bun10].

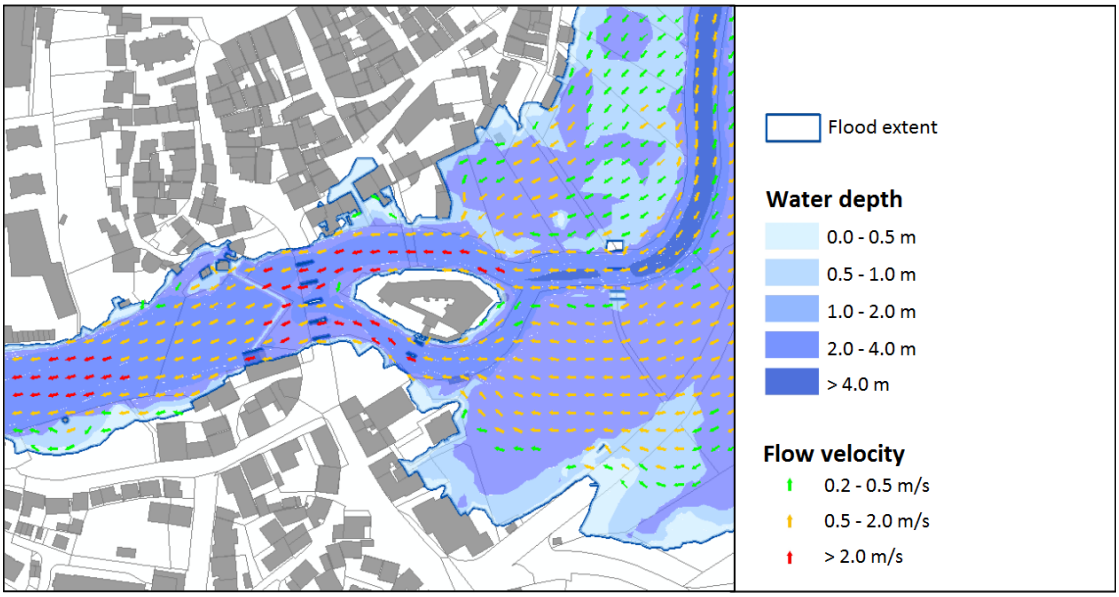



**Figure 3.2.:** Recommendation for the layout of a flood hazard map including flow velocities [Bun10, p. 18, the legend has been translated by the author.]

### 3.1.2. The Process of Flood Map Creation

This subsection aims at describing a flood mapping methodology in line with the EU Floods Directive. Apart from the regulations on the content of flood maps, unfortunately, the process of flood map creation is rather unspecified. However, more important than the map layout is the methodology and data used for production of the map. There are many ways in which flood maps can be derived. The most commonly applied models are hydrologic and hydrodynamic models, but specific causes of a flood event require

other kinds of models (e. g. groundwater). A good overview of the process can also be found in [MAA09].

### Basic Geographic Data

The flood mapping process starts with the compilation of input data. This data forms the basis of the numerical models and the final flood map. It involves surface elevation, land use data, meteorological data, as well as information about the course of historical flood events.

A digital elevation model (DEM) represents the surface *topography* of a model area. A flood map can only be as accurate and reliable as the DEM. Current elevation data is often obtained using remote sensing technology, such as airborne laser scanning, with a horizontal resolution $< 1$ m and an error $< 20$ cm. Additionally, for a high-quality hydrodynamic calculation, the *bathymetry* of watercourses including embankments and engineering structures (dikes, weirs, bridges) have to be represented correctly in the DEM. The bathymetry is often surveyed separately, e. g. in the form of cross-sectional profiles, which has to be integrated into the topography data. The LAWA recommends that the final DEM provides a resolution of $< 2$ m and that hydraulically relevant terrain features, such as breaklines, are surveyed independently.

*Land use* is an indication of how humans use a certain land cover type, whereas *land cover* is the actual consistence of the surface of the earth, e. g. asphalt, trees, or water. Land cover data is gathered either by field survey or by classification of remote sensing imagery (with a resolution similar to the topography) and serves as an estimate of terrain roughness. Land use data may include e. g. population, economical, and environmental statistics. Examples are industrial areas, pastures, or natural reserves. In urban areas these classes are often coarse-grained zones, but they can also be as detailed as single land parcels in a cadastral map.

*Meteorological data*, i. e. weather and climate data, is relevant to hydrologic modeling (see below). It includes, among others, wind speed and direction, air temperature, and precipitation (rainfall, snowfall). Historical rainfall data is important as input for hydrologic models, finding statistical return periods, and designing hypothetical rainfall and flood events of a certain intensity by flood frequency analysis. Meteorological simulation models may be used to create near-term precipitation forecasts and long-term projections of climate change.

Historical flood extents, water stage, and discharge data are used for the calibration and verification of flood models.

### Hydrologic Models

*Hydrologic models* conceptually represent the hydrologic cycle, i. e. the circulation of water on, above, and below the surface of the earth. The two major types of hydrologic models are deterministic and stochastic models. Stochastic (or probabilistic) hydrologic models are mathematical black box systems employing techniques such as regression, transfer functions, or neural networks. They have in common the attempt to create a relationship between precipitation and runoff, for which reason they are also named rainfall-runoff models. The aim of hydrologic models in flood mapping is to predict the hydrograph, i. e. runoff over time, at selected river locations caused by a specific rainfall event. This information is subsequently used as input to a hydrodynamic river model.

Current deterministic hydrologic models are either fully spatially distributed, i. e. based on the topography, or (semi-) distributed but conceptually lumped, i. e. based on the spatial concept of a hydrotope. A hydrotope is a modeling unit that delineates an area with common hydrologic properties. They simulate the horizontal and vertical physical processes in the hydrologic cycle. Vertical processes are e. g. snow storage, evapotranspiration, soil water storage, and groundwater recharge. Horizontal processes include surface runoff (overland flow), subsurface runoff (interflow), and groundwater flow (baseflow). A main aspect of surface runoff is the propagation of a flood wave in water courses (flood routing). The processes in a water course are obviously highly hydrodynamic and are best modeled as such. Lumped rainfall-runoff models, however, typically treat river strands as linear reservoirs, or quasi-steady cascades of reservoirs, in order to determine translation and retention of water. Hydrologic models need to be calibrated prior to use [Bro05; AR96].

### Hydrodynamic Models

The class of *hydrodynamic models* (also commonly called *hydraulic models* or *flow models*), as applied in surface hydrology, examine the flow of water on the surface in detail, i. e. water level, flow velocity, and flow direction are represented directly. Generally, meaningful flow velocity information in flooded areas can only be derived from two-dimensional hydrodynamic models. The simulation of pluvial flooding requires a special kind of hydrodynamic model that can quickly flood an initially dry model using inflow over a model boundary or distributed rainfall. Like hydrologic models, hydrodynamic models need to be calibrated.

For detailed information regarding hydrodynamic models see Section 3.2.

#### Map Evolution

Flood hazard maps are subject to regular changes in the underlying data model, rendering them obsolete. Additionally, decision makers might adapt the preconditions of map creation, i. e. the choice and definition of flood scenarios to be mapped.

Furthermore, changes in the data model have to be expected due to urban development. In addition, erosion and deposition of sediment material also leads to a natural evolution of river beds and coastal areas. These changes commonly concern the topographic and land cover data. In this case, the map needs to be recreated with new parameters, typically requiring hydrologic and hydrodynamic models to be adapted and more simulations to be performed.

### 3.1.3. Possible Derivative Products

Based on flood hazard maps, other flood map products can be derived. Of major importance are the creation of flood damage and risk maps as well as flood risk management and emergency plans. At the same time, strategies to reduce the risk of flooding need to be developed and their effectiveness must be assessed.

#### Flood Risk Maps

Flood hazard is a component of flood risk (see Figure 3.3). In order to estimate the flood risk inside an area, one has to account for the consequences of a flood by vulnerability assessment, i. e. determining the exposure of buildings and people as well as their susceptibility relative to the previously derived flood intensity. Flood incidents and the resulting damages should be recorded so that the impact of a future event — and the total expected annual flood damage — can be estimated with a higher confidence.

#### Emergency Plans

An emergency, or disaster, is a major incident with a high (expected) damage or an increased risk of casualties. Emergency plans have the purpose of raising the people's awareness of living in a flood-prone area. They give guidance regarding precautions to take before a flood (preparedness), what to do or where to go during a flood (response, e. g. following an evacuation plan or trying to avoid health risks), and how to recover after a flood. The creation of evacuation plans is a complex task supported by flood maps and, additionally, knowledge of the course of flooding. This knowledge can only be gained through previous experiences and records or numerical simulation.

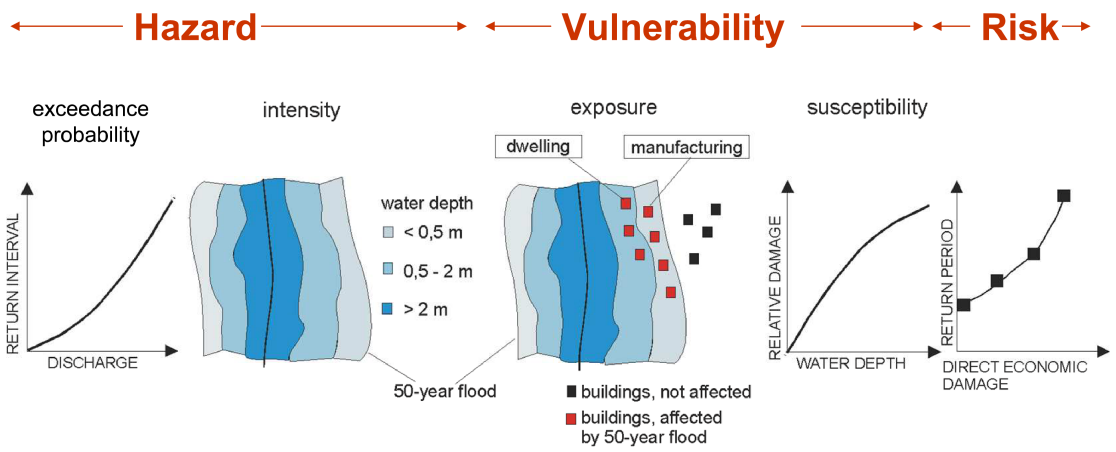



**Figure 3.3.:** Three components of flood risk [MT04]

#### Mitigation of Flood Risk

Flood mitigation efforts try to prevent or reduce the effect of disasters. Strategies that reduce the flood risk, according to the source-pathway-receptor model [Fle02], act on one of the two conceptual areas related to the process of flooding: the flood plains and defenses (pathway) or the people and properties at risk (receptor). Measures on the pathway are typically of a structural nature and aim at the prevention of floods, such as building a dike or retention basin. Non-structural measures, on the other hand, involve redirecting part of the flood to where it causes less damage or protecting people and properties, so they are not as vulnerable to a flood. Non-structural measures are highly political, i. e. involving regulations and legislation, and depend on careful urban planning, public information and education, and flood warning systems.

## 3.2. Hydrodynamic Numerical Models

Whereas hydrologic models predict the amount of discharge during a flood at designated points in a water course, the detailed, time-dependent flow pattern of the flood can only be determined by hydrodynamic models. They relate the course of a flood wave at the boundary of the model to the time-dependent flow state at discrete points or volumetric cells inside the model.

This section gives the theoretical background on the numerical modeling of shallow water hydrodynamics. Subsection 3.2.1 presents variants and the governing equations of depth-averaged hydrodynamic models. Next, Subsection 3.2.2 describes approaches

to the discretization and numerical solution of the resulting mathematical model. For further information regarding hydrodynamic models see, for example, [Nov10; Din08; Pas07; Cha04; Jul02; Mal01; Mal10].

### 3.2.1. The Shallow Water Equations

#### The Navier-Stokes Equations

The velocity field in a fluid at a certain point in time can be described by the so-called Navier-Stokes equations. These vector-valued equations consist of a momentum equation (conservation of momentum) and a continuity equation (conservation of mass).

In order to have shorthand expressions for the gradient of a scalar field as well as the directional derivative $\nabla$ (del or nabla) is defined as the vector operator of partial derivatives $\left(\frac{\partial}{\partial x_1}, \frac{\partial}{\partial x_2}, \ldots, \frac{\partial}{\partial x_n}\right)$.

Assuming that the flow is incompressible, i. e. no shock waves may occur, the continuity equation of the Navier-Stokes equations has the simple form

$$\nabla \cdot \mathbf{v} = 0 \tag{3.1}$$

where $\mathbf{v}$ denotes the flow velocity vector. The Navier-Stokes momentum equation can be written in vector notation as

$$\underbrace{\frac{\partial \mathbf{v}}{\partial t}}_{\substack{\text{Local}\\ \text{acceleration}}} + \underbrace{\mathbf{v} \cdot \nabla \mathbf{v}}_{\substack{\text{Convective}\\ \text{acceleration}}} = \underbrace{-\frac{\nabla P}{\rho}}_{\substack{\text{Pressure}\\ \text{gradient}}} + \underbrace{\nabla \boldsymbol{\tau}_{visc}}_{\substack{\text{Viscous normal}\\ \text{and shear stress}}} + \underbrace{\mathbf{f}}_{\substack{\text{Outer body}\\ \text{forces}}} . \tag{3.2}$$

Here, $P$ is the pressure and $\rho$ is the fluid density. The viscous and normal shear stress term $\boldsymbol{\tau}_{visc} = \nu \nabla \mathbf{v}$.

depends on the kinematic viscosity $\nu$.

### The Reynolds-Averaged Navier-Stokes Equations

The flow of water in nature is always three-dimensional, unsteady, and turbulent. Because turbulence takes place at extremely low spatial and temporal scales, time-averaging the equations helps in the practical numerical solution of flow problems. The velocity is split into a mean part ($\overline{\mathbf{v}}$) and a fluctuating part ($\mathbf{v}'$), which yields the so-called Reynolds-averaged Navier-Stokes equations. The continuity equation becomes

$$\nabla \cdot \overline{\mathbf{v}} = 0 \tag{3.3}$$

and the Reynolds-averaged momentum equation is

$$\frac{\partial \overline{\mathbf{v}}}{\partial t} + \overline{\mathbf{v}} \cdot \nabla \overline{\mathbf{v}} = -\frac{\nabla \overline{P}}{\rho} + \nabla \boldsymbol{\tau}_{visc} + \underbrace{\frac{\nabla \boldsymbol{\tau}_{turb}}{\rho}}_{\substack{\text{Reynolds} \\ \text{shear stress}}} + \mathbf{f} \tag{3.4}$$

with $\boldsymbol{\tau}_{turb}$ being the Reynolds (turbulent) shear stress tensor $\boldsymbol{\tau}_{turb} = -\rho\, \overline{v_i' v_j'}$.

In effect, it is assumed that small-scale eddies can be averaged out and replaced by an eddy viscosity (Boussinesq hypothesis). A turbulence model must be applied to give a physical meaning to $\boldsymbol{\tau}_{turb}$.

### The Two-Dimensional Shallow Water Equations

The solution of the three-dimensional Reynolds-averaged Navier-Stokes equations in large domains remains to be challenging even with the power of current computers. In order to reduce computation times, simplifications are made in the mathematical formulation. A suitable approximation for the hydrodynamic modeling of floods in rivers and coastal areas is given by the so-called depth-averaged shallow water equations (SWE). They are based on the assumption that the horizontal spatial scale is much larger than the vertical scale regarding water depth and flow velocity. Essentially this means that the water depth is much smaller than the wave length and that vertical acceleration is negligibly small compared to gravity and horizontal velocity components.

The SWE can be derived from the Reynolds-averaged Navier-Stokes equations by depth-averaging the velocities and integrating all terms over the water depth, reducing the three-component velocity to $v_x$ and $v_y$ in a Cartesian coordinate system. The equations may then be expressed with a dependence on the water depth $h$ measured from the

water surface. A discharge vector $\overline{\mathbf{q}}$ is introduced as the depth-integration of the depth-averaged flow velocities from a bottom elevation $z_0$ to the water surface elevation $H = z_0 + h$:

$$\overline{\mathbf{q}} = \int_{z_0}^{H} \overline{\mathbf{v}} \, \mathrm{d}z = h\,\overline{\mathbf{v}}. \tag{3.5}$$

From now on, the horizontal line indicating the time- and depth-average of a term will be omitted. The continuity equation of the SWE can then be stated as

$$\frac{\partial h}{\partial t} + \nabla \cdot \mathbf{q} = 0. \tag{3.6}$$

Finally, by taking on a hydrostatic pressure distribution, i. e. $P = \rho g h$, where $g$ is the gravitational acceleration, the pressure term can be replaced by a water depth term and the momentum equation of the SWE becomes

$$\frac{\partial \mathbf{q}}{\partial t} + \mathbf{q} \cdot \nabla \mathbf{v} = \underbrace{-gh\nabla H}_{\substack{\text{Free surface}\\ \text{pressure}\\ \text{gradient}}} + h\,\nabla \boldsymbol{\tau_{visc}} + \frac{h\,\nabla \boldsymbol{\tau_{turb}}}{\rho} + \underbrace{\nabla \boldsymbol{\tau_{disp}}}_{\substack{\text{Dispersive}\\ \text{term}}} + \underbrace{\nabla \boldsymbol{\tau_{fric}}}_{\substack{\text{Bottom and}\\ \text{surface}\\ \text{friction}}} + \mathbf{f}. \tag{3.7}$$

The depth-integration leads to friction shear stresses $\boldsymbol{\tau_{fric}}$ at the bottom $z_0$ and surface $H$ and further introduces a dispersion error similar to the Reynolds shear stress that has to be accounted for in the dispersive term $\boldsymbol{\tau_{disp}}$. Because the effect of the viscous shear stress $\boldsymbol{\tau_{visc}}$ is much smaller than the other shear stresses, it is commonly ignored.

### 3.2.2. Numerical Solution of the Shallow Water Equations

Flood hydrodynamics as described by the two-dimensional shallow water equations leads to a system of nonlinear partial differential equations. It is a continuous problem that can be solved numerically by a discretization method. There is a variety of these methods available, but most of them entail some approximate solution of the problem on a finite set of interconnected structural elements, which is called mesh or grid[1].

[1]The term *grid* for a computational mesh will be avoided to reduce the chance of confusion with the term *grid computing*.

Meshes are either structured (e. g. raster-like) or unstructured (e. g. consisting of triangles). Unstructured meshed are more flexible because they allow for local refinements in the mesh without increasing the global resolution and coupling of different geometrical shapes of structural elements. Discretization methods are classified into finite element, finite volume and finite difference methods. A few methods do not rely on a mesh, but use a particle-based or hybrid approach (meshfree methods).

This thesis builds on an unstructured finite element discretization. However, many of the practical considerations regarding flood modeling are not specific to the finite element method and can be generalized to other unstructured discretizations of the SWE or variations thereof.

#### Finite Element Discretization

Finite element methods (FEM) approximate the continuous fields $\mathbf{q}$ and $h$ of the SWE at a set of discrete points in a two-dimensional, unstructured mesh. A finite element mesh is an unstructured division of the model area into a set of regular polygons, typically triangles or quadrilaterals. This renders FEM a very flexible and powerful class of methods for regions with complex geometries [Ter10]. In the context of geographic information systems, unstructured meshes with only triangles are also often called Triangulated Irregular Networks (TINs).

In order to derive a functional representation of the SWE for the elements, an integral form of the equations is developed. It uses a linear combination of a finite number of piecewise polynomial basis functions to approximate the original function. The result is a set of algebraic equations for the unknowns. The approximation error is called residual.

#### Solving Sparse Nonlinear Systems

A nonlinear system of equations can be transformed into a series of linear systems, e. g. by an iterative Newton-Raphson, line search, or trust region method [Kel87]. As a result, the Jacobian matrix of first partial derivatives, or parts of it, has to be assembled in each iteration step, which involves calculating the element equations (stiffness matrix) for each element. All element contributions are either inserted into a global sparse matrix or calculated when needed.

A sparse linear system $Ax = f$ has to be solved in each nonlinear step. One distinguishes between direct and iterative solvers. The direct solution depends on a matrix decomposition, in particular the LU factorization $A = LU$ where $L$ and $U$ are lower

and upper triangular matrices, whereas iterative methods seek an approximate solution and can further be classified into stationary and Krylov subspace methods [Saa03]. Stationary methods are rather slow and restricted to certain matrix types, so the majority of iterative solvers implement a Krylov subspace method, e. g. conjugate gradient (CG) [HS52], biconjugate gradient stabilized (BiCGstab) [SF93; Sv95], or generalized minimal residual (GMRES) [SS86].

#### Preconditioning Iterative Methods

The success and rate of convergence of an iterative method depends largely on the conditioning of the problem. An ill-conditioned system leads to bad convergence behavior. The goal of preconditioning is to reduce the condition number of the matrix $A$. Rather than solving the original system, the left preconditioned system $P^{-1}(Ax - f) = 0$ is solved, where $P$ is the preconditioning matrix. Since $P^{-1}$ has to be applied in every iteration, a trade-off has to be made between the effort to calculate $P^{-1}A$ and the acceleration of the iterative solution. Examples of preconditioning are the Jacobi (or diagonal) preconditioner $P = diag(A)$ or incomplete LU factorization (ILU) $P \approx LU$.

Two other methods to accelerate the solution process shall be mentioned here: geometric or algebraic multigrid[1] and domain decomposition methods. The idea of geometric multigrid is to approximate the original problem on a hierarchy of coarser meshes and use their solutions to correct the initial guess. Algebraic multigrid works directly on the system matrix and is thus independent of any geometric representation of the problem on a mesh. Domain decomposition methods split the original problem into possibly overlapping, independent sub-problems of smaller size that are easier to solve. An iterative procedure must be applied to match the values on the sub-domain interfaces in the global solution. Again there are the two possibilities of geometric and algebraic domain decompositions.

## Conclusion

The flood mapping process requires two-dimensional, numerical flow models in order to characterize both the dynamics of a flood event and a reliable representation of the inundated areas. Two-dimensional hydrodynamic model creation and simulation are computationally demanding, time- and data-intensive tasks conducted by modeling experts and engineers. The simulation results can be used to create flood hazard

[1] Please note: The term *multigrid* does not imply that *grid computing* is used.

maps or derive further informative products, such as flood risk maps and evacuation plans. Moreover, legal requirements regarding flood risk management entail additional flow simulations so that the performance of different mitigation strategies can be compared.

In the near future, all basic geographic data used in the process will be provided by national spatial data infrastructures. When that happens, the question will occur how to efficiently make use of this data in flood mapping and other applications. Simultaneously, as flood models become larger and more detailed data becomes available, a computational infrastructure is needed for the processing of these data. Flood simulations, in particular, can be regarded as long-running processes that produce data of common interest. Exactly for this purpose, the following chapter will show how the hydrodynamic modeling process can be split into modular processing tasks that can be *integrated* into a spatial data infrastructure, i. e. they make use of existing geodata and processing standards to access data and feed back their results. The logical next step (in Chapter 5) is to establish a framework that allows these processes to be executed in a computational infrastructure.

# 4. The Hydrodynamic Modeling Process

This chapter delineates the hydrodynamic modeling process by taking a detailed look at the different phases of flood modeling (see Figure 4.1). The individual work steps of these phases emerge from flood mapping practice using a two-dimensional shallow water flow model based on an unstructured finite element discretization. The purpose of this chapter is to enable the reader to understand the rationale for the proposed gridification method described in Chapter 5. At the same time, it introduces the methodology on which the prototypes are founded (Chapters 6 and 7).

The three phases of the flood modeling process are mesh generation (pre-processing), flood simulation (processing), and results evaluation (post-processing). Mesh generation is the most important task when setting up a hydrodynamic model for flood simulation. Equally important is the calibration of the model in the processing phase in order to improve the quality of the simulation results. The necessary model calibration also interrupts the process at this point. In the final phase, the results form the basis of flood maps to be created. Two-dimensional hydrodynamic modeling comprises tasks that, to a large extent, require processing of geospatial data (geoprocessing)[1].

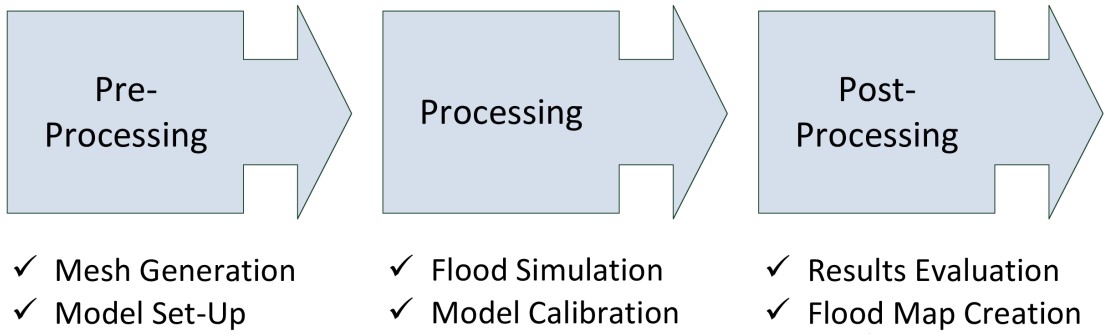



**Figure 4.1.:** The three phases of the hydrodynamic modeling process.

[1]Here and in the following, the term *task* is to denote an abstract unit of work, while *operation* stands for the concrete, technical realization of a task as part of a software solution. The Web Processing Service specification also uses the term *process* for an individual geoprocessing operation.

Geoprocessing is also vital to many other fields in earth science, e. g. geology, geography, geodesy, oceanography, meteorology, and hydrology. The term *geoprocessing* refers to the repeatable execution of any operation on geographic data, such as data conversion or spatial analyses. A geoprocessing operation has a defined number of typed inputs and outputs. An example of a typical basic geoprocessing operation is "buffering", i. e. finding the set of points at a specified distance from the geometry of an input feature. However, a geoprocessing operation can also be complex, long-running, and resource-intensive, or a sequence of operations.

The hydrodynamic modeling process is complex and difficult because different types and quantities of data have to be analyzed, manipulated, and transformed to derivative products in each of the three phases. All hydrodynamic modeling tasks require extensive knowledge and experience. Most are computationally intensive or process large amounts of data (see Chapter 3). Some need parameter studies to be conducted, and sometimes it is necessary to go back and improve the results of earlier steps. In the following sections, modular geoprocessing operations for these tasks will be identified, which can be integrated into a spatial data infrastructure. Furthermore, the large potential of geoprocessing in a grid infrastructure will be shown.

## 4.1. Processing Geographic Data

In the last years, considerable progress has been made in open standards for modeling and processing of geographic data. The Open Geospatial Consortium, Inc.® (OGC), founded under the name *Open GIS Consortium* in 1994, is now the leading international voluntary consensus organization for standardization of geospatial information and services. The OGC has issued a collection of standards for serving geographic information over the web. This includes services for map-like products, for vector-based data, for raster data or other coverages, as well as for observations, such as sensor time series. The leading spatial data infrastructure initiative in European (INSPIRE) has adopted many of the OGC's standards [Neb04].

It is current practice that geographic data is primarily processed in a GIS. Standards-based *geoprocessing services*, as opposed to local geoprocessing, enable users to conduct geospatial analyses remotely over a network. This is beneficial for three reasons:

- Resource-intensive computations are moved to a geoprocessing server, which has the potential of creating more efficient applications and reducing costs.
- Complex analysis tasks can be made available to a wider audience on the Internet using custom-tailored front-ends.

- Comprehensive geoprocessing operations can be composed from basic building blocks, improving software modularity, interoperability, and reuse.

Moreover, the integration of geoprocessing services into a grid infrastructure generates additional synergies:

- Grid resource providers gain a generic, standards-based interface to processing services, hiding the inherent complexity of the grid.
- Geoprocessing operations can be distributed to many computing resources and may use the grid as a storage environment.
- Geoprocessing services can rely on grid infrastructure for security and state management.

As of 2005 there also exists a standard for remote geoprocessing, which has found considerable attention in the open GIS community. It will serve as the foundation on which a framework for geoprocessing grid services will be built in Chapter 5.

### 4.1.1. The OGC Web Services Architecture

All OGC web service (OWS) standards follow an architectural style similar to one coined by Fielding [Fie00], which he called representational state transfer (REST). Key to REST architectures are the two concepts of a *resource*, identified by a URI, and its *representation*, e. g. a document or image.

An OWS accesses resources and provides operations over the transport protocol HTTP. However, it does not use HTTP as an application protocol. Instead, the semantics of an operation are either encoded in the URI of a HTTP GET request (key-value argument pairs) or inside an XML document sent via HTTP POST. This inconsequential design has frequently been criticized [GD+12]. Due to the prevalence of SOAP bindings of web services, an alternative option has been proposed in recent versions of OWS standards. Operations may also be called by sending XML requests in a SOAP envelope. This greatly improves interoperability between industry web service products and OGC web services. The SOAP binding of WPS is also used in this chapter for the development of the Grid-WPS framework.

The respective service metadata is similar for all OWS. It is given in a structured, human-readable format, which can be queried by a `getCapabilities` request. The common metadata document includes the service provider, contact details, and a list of the provided operations. Data and map services, for example, have operations for performing spatial queries on specified data sets, while the WPS offers a list of geoprocessing operations.

### 4.1.2. Data and Measurement Services

The OGC data service specifications have the goal of defining a standard interface for accessing geographic data over the web. Typically backed by a database or cascade of storage servers, they allow clients to obtain filtered subsets of data by querying a web service. Such filtering simplifies the management and processing of large sets of geodata, such as digital terrain data (see Section 4.2 and Chapter 6), which is important for geoprocessing in the grid. Likewise, observation and measurement services provide access to real-time or historic sensor data. Such data is both used as input to a flood simulation, e. g. rainfall intensities or river water stages, and obtained as a result of the simulation (see Section 4.3 and Chapter 7).

A service for vector data is the Web Feature Service (WFS). The service is also standardized as ISO 19142. According to [Vre10], a WFS provides

> [...] transactions on and access to geographic features in a manner independent of the underlying data store. It specifies discovery operations, query operations, locking operations, transaction operations and operations to manage stored parameterized query expressions.
>
> Discovery operations allow the service to be interrogated to determine its capabilities and to retrieve the application schema that defines the feature types that the service offers.
>
> Query operations allow features or values of feature properties to be retrieved from the underlying data store based upon constraints, defined by the client, on feature properties.
>
> Locking operations allow exclusive access to features for the purpose of modifying or deleting features.
>
> Transaction operations allow features to be created, changed, replaced and deleted from the underlying data store.
>
> Stored query operations allow clients to create, drop, list and described parameterized query expressions that are stored by the server and can be repeatedly invoked using different parameter values.

Geographic features are exchanged over WFS encoded in the Geography Markup Language (GML) [Por07],

> [...] an XML grammar for expressing geographical features. GML serves as a modeling language for geographic systems as well as an open interchange format for geographic transactions on the Internet. [...] A GML document is described using a GML Schema. This allows users and developers to

> describe generic geographic data sets that contain points, lines and polygons. However, the developers of GML envision communities working to define community-specific application schemas that are specialized extensions of GML. Using application schemas, users can refer to roads, highways, and bridges instead of points, lines and polygons.

Friis-Christensen et al. [FO+07] pointed out that the WFS specification is not suited to serving large amounts of data in its current design. They attempted to process a large CORINE land cover data set. Part of the problem is the exchange of geodata in the GML format and thus comes with a large overhead of its text-based representation and transport over HTTP. Neither does GML have an integrated spatial index.

The Web Coverage Service (WCS) is a service for data covering an area, such as raster data [Bau10]. ISO 19123[1] provides a definition of the term *coverage*:

> Coverages support mapping from a spatiotemporal domain to attribute values where attribute types are common to all geographic positions within the spatiotemporal domain. A spatiotemporal domain consists of a collection of direct positions in a coordinate space. Examples of coverages include rasters, triangulated irregular networks, point coverages, and polygon coverages. Coverages are the prevailing data structures in a number of application areas, such as remote sensing, meteorology, and bathymetric, elevation, soil, and vegetation mapping.

For use cases, in which sensor data needs to be used in interoperable ways, the OGC has issued the Sensor Observation Service (SOS) specification [BSE12] as part of its Sensor Web Enablement (SWE) group of standards [Sim08]. [BSE12] describe SOS as

> [...] a standardized interface for managing and retrieving metadata and observations from heterogeneous sensor systems. Sensor systems contribute the largest part of geospatial data used in geospatial systems today. Sensor systems include for example in-situ sensors (e. g. river gauges), moving sensor platforms (e. g. satellites or unmanned aerial vehicles) or networks of static sensors (e. g. seismic arrays).

### 4.1.3. The Web Processing Service

The Web Processing Service (WPS), version 1.0, is the de-facto standard for designing interoperable, distributed geoprocessing services [SG+05; FH+07]. The WPS specification aims at answering four questions that arise when designing a service-oriented geoprocessing architecture:

[1] ISO 19123:2005 (Geographic information – Schema for coverage geometry and functions)

- What are the inputs and outputs of a geoprocessing operation and what do they look like?
- How are inputs passed from a client to the service and outputs passed back?
- How is the execution of a geoprocessing operation started?
- How does the client know when execution is finished? How can he query the current status of the execution?

In the following, the two WPS operations related to describing and executing the solicited geoprocessing operations, called processes, will be described in detail. More information regarding the WPS and examples of its use can be found in [HK05; KGH06; SZ07; SZ08; KGH07; NKB07; FH+09; SS09].

#### Process Description

Full information about each of the offered processes can be obtained via a `describeProcess` request. This includes a description of the number and types of input and output parameters. Parameters are either literal (simple types such as numbers or character strings) or complex (such as XML or binary). Complex parameters may either be embedded into the request (in-line) or are a reference to a web-accessible resource. Using referenced inputs and outputs simplifies service composition and handling of large data.

The syntax of a complex input is defined by an internet media type (MIME-type), e. g. `text`, `image`, `application/gml+xml` (for GML data), and a link to an XML or GML application schema, if appropriate. Using this solution, it is not possible, in a simple way, to express that a complex parameter conforms to a certain XML schema element or type *inside* the provided schema. This missing option has already been accepted for change in the next version of the WPS specification.

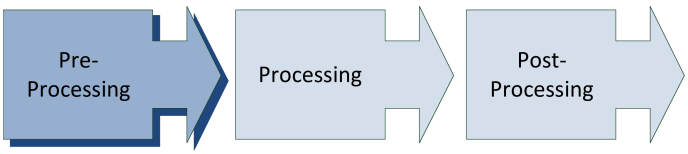
#### Execution

A geoprocessing operation is started using the `execute` request. The current WPS specification does not offer the possibility to cancel, pause, or resume a currently executing process. Such operations will be included in the next WPS version. The process may either return the process outputs literally in the response or store them as web-accessible resources. A service may also support asynchronous execution by returning a status document in place of the results. Asynchronous execution is recommended for long-running processes, e. g. jobs submitted to a computational

grid. The status document is continuously updated. It includes information about the progress of the process and which results are currently available. For storing the status document and result files, the server needs access to an appropriate storage location, which could be part of a data grid.

A generic WPS client is available as part of the uDig open GIS framework[1] [FS07]. It finds out about available processes as well as the number and type of process inputs by parsing the process description. The information is used to display a dynamic user interface for entering concrete input values or references.

## 4.2. Pre-Processing – Mesh Generation




The first phase of hydrodynamic modeling has the aim of building a mesh used in the finite element discretization of the shallow water equations. In this section, a mesh generation strategy is developed that can ultimately be executed in the grid (Chapter 6).

### 4.2.1. Automatic Mesh Generation

At the beginning of the computer age, when the simulated problems were much smaller and computer power was limited, meshes were created mostly by hand. Nevertheless, there were efforts to devise automatic meshing schemes. Nowadays, drawing the mesh element by element is not feasible anymore, so modelers have to rely on automatic mesh generation or computational geometry tools, such as the Computational Geometry Algorithms Library (CGAL)[2], the GNU Triangulated Surface Library (GTS)[3], or the popular and widely-used Triangle[4] [She96; She02].[5]

An unstructured mesh is a finite polyhedral surface forming a two-dimensional simplicial complex, i. e. any two adjacent elements have exactly one common edge, and the

[1] `http://udig.refractions.net`
[2] `http://www.cgal.org`
[3] `http://gts.sourceforge.net`, last updated in 2006
[4] `http://www.cs.cmu.edu/~quake/triangle.html`, last updated in 2005
[5] Robert Schneiders maintains a web page with information on mesh generation and meshing software at `http://www.robertschneiders.de/meshgeneration/meshgeneration.html`.

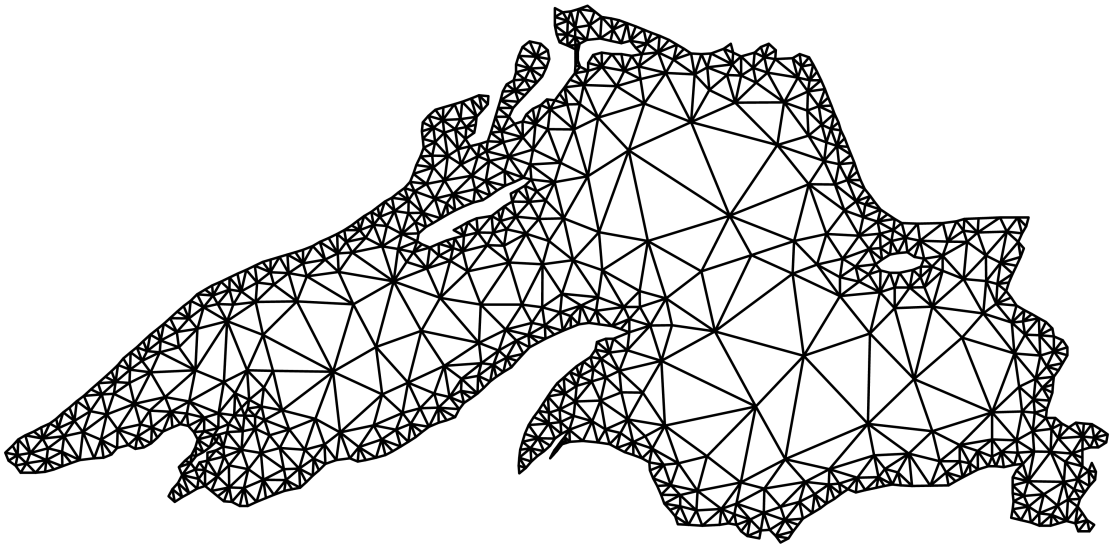


**Figure 4.2.:** Example of a mesh created using Triangle [She02, p. 45]. It is a constrained Delaunay triangulation (TIN) of a polygonal boundary (here: Lake Superior, USA/Canada) with a 25° minimum angle constraint for quality improvement.

set of all elements is a tessellation of the surface. Of different types of unstructured meshes, triangulated irregular networks (TINs, compare Figure 4.2) have been studied the most. A TIN is a tessellation with only triangles and is also called triangulation. A Delaunay triangulation ensures that the circumcircle of any triangle does not contain any other mesh point. Constrained Delaunay triangulations additionally enforce given edges to be part of the TIN. In general, a Delaunay discretization has good numerical properties with respect to the finite element method as it maximizes the minimum internal angle of each triangle. TINs are also supported by many flow models. Likewise, most meshing tools can create a Delaunay triangulation.

A finite element mesh needs to achieve two important goals: First, it has to provide a good approximation of the terrain, and second, the mesh must have sufficient detail to represent the expected flow, i. e. it needs a higher resolution in areas with a complex flow pattern. Unstructured meshes are generally better suited to meet these criteria than structured meshes, because the element size can be adapted where a higher resolution is desired. Naturally, meshes with a higher resolution (more elements) tend to form better approximations of the terrain. However, practical considerations urge modelers to keep the number of mesh elements as small as possible: A typical current desktop computer (or single computing node in a grid) can only simulate meshes with tens to hundreds of thousands of elements.

Mesh generation tools (e. g. Triangle) have no problem creating a triangulation with millions of elements. This reveals that there is a discrepancy between the size of the meshes that can be automatically generated and those that can actually be simulated on a single resource. As a result, this thesis does not try to create a parallel triangulation routine for the purpose of flow model discretization. More information on the Delaunay triangulation process can be found in [She02]. Parallel triangulation algorithms have been previously investigated in [CC+10; ST+07]. The actual problems relevant for this thesis are found in managing and analyzing huge amounts of terrain data to find the features relevant for flow simulation in the terrain surface.

### 4.2.2. A Meshing Methodology for Flow Simulation

The proposed method starts with a basic terrain data set. Terrain data is given either as an unstructured point cloud, e. g. from airborne measurements, or as a regular raster of points in a 3D Cartesian coordinate system (x-, y-, and z-axes). The 3D points can easily be projected onto a 2D surface by dropping the z-coordinate, so this kind of data is sometimes referred to as 2.5D. It is assumed that sufficient terrain data is available to completely cover the study area. The terrain model that serves as the basis for the subsequent mesh generation steps will be called *base terrain model* (BTM). An example of rasterized terrain and bathymetry data sets forming a BTM for the Hamburg Metropolitan Region can be found in Figure B.1 on page 139.

#### The Study Area

Each flood model can only represent parts of a real flow situation, so a suitable boundary of the area under investigation (study area) has to be defined. The boundary of the hydrodynamic model has to be chosen according to several parameters: the type of flood, the expected flood extent, and the available flood hydrograph locations. At this point, the flood modeler should know which flood scenarios have to be simulated. A preliminary assessment of the flooding process helps to delineate the study area.

The study area is represented by a polygonal region. This polygon defines the mesh boundary. The polygon is allowed to contain holes, e. g. to exclude certain areas from the flow simulation. Points on the BTM surface can be classified as being located either inside the study area or outside.

### Overview of the Process

The finite element mesh can be regarded as an approximation, or generalization, of the BTM inside the study area. In the generalization process, the essential structure of the terrain affecting the flow processes must be identified and retained in the mesh. The meshing tools mentioned above can be applied to the BTM in two ways. The first is to create a triangulation of the points (in 2D) to obtain a terrain surface representation, discard all triangles outside the mesh boundary, and then simplify this surface in 3D. In the second, the study area polygon and a number of structural edges are triangulated (in 2D), then the mesh surface elevations are interpolated from the BTM surface (back to 3D).

The first approach has the drawback that the mesh points are a subset of the terrain or arbitrary intermediate points. A common problem with the BTM, which is only an approximation itself, is the fact that the set of terrain points (or raster cells) almost never contains exact representatives of the terrain features, which determine the flow pattern when the terrain is flooded. In addition, point clouds are noisy and, once non-ground points have been filtered out, they have a highly varying point density or even contain holes. For these reasons, the quality of the mesh would depend largely on the quality of the BTM. Meshes created in the second approach, on the other hand, require a well-digitized study area boundary and structural edges. The quality of approximation is determined by whether the structural edges represent significant features of the terrain. An additional step is required to derive these features from the BTM. Nevertheless, the second approach has the crucial advantage of having complete control over the mesh structure.

In the following an automatic meshing methodology based on the second approach is presented. It includes structural terrain features as constrained *breakline* edges into the mesh. Here, the term breakline refers to any kind of edge that can automatically be detected in a raster DEM. The method has been inspired by [Rato7] and includes the following steps (compare Figure 4.3):

- Create a raster from the BTM, or resample a raster BTM. The raster resolution determines how well breaklines can be located.
- Perform breakline detection. The user determines a threshold for breakline strength.
- Create a constrained Delaunay triangulation of the boundary polygon and the breaklines. The user determines the quality and resolution of the triangulation.
- Interpolate elevations from the BTM to the mesh.

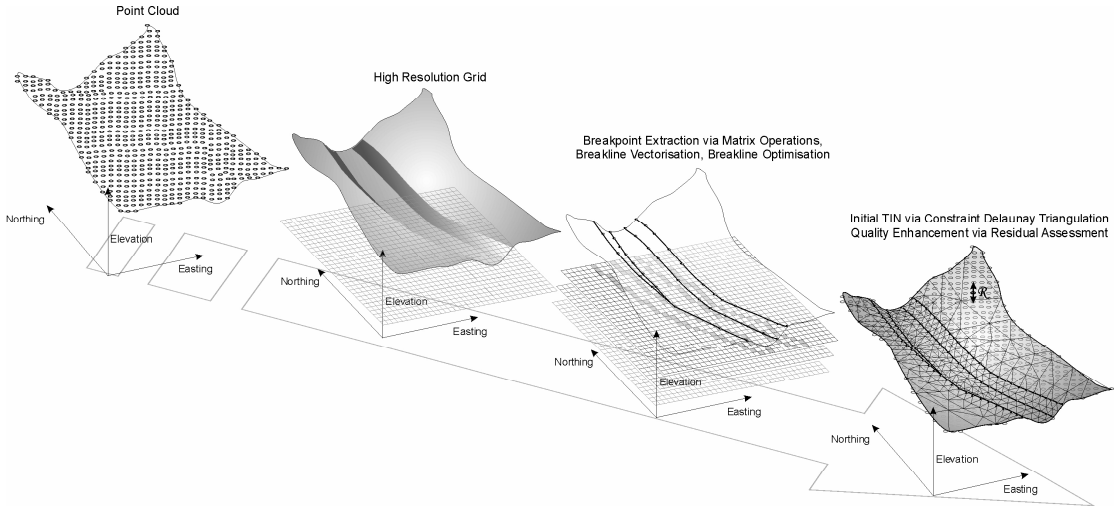



**Figure 4.3.:** Process template for unstructured mesh (TIN) generation based on a point cloud DEM, an intermediate raster DEM ("high resolution grid"), terrain breakline detection, and constrained Delaunay triangulation [Rato7, p. 67].

### 4.2.3. The Base Terrain Model

The BTM is a ground surface representation of the topography and bathymetry of the study area with the highest detail available. The creation of a BTM is not easily automated as a geoprocessing operation, so this step will not be covered in this thesis. It is the modeler's responsibility to prepare an adequate BTM in a GIS, either as a regular raster or as a point cloud. An unstructured BTM has the advantage of being flexible with regard to the insertion of vector features, while a raster BTM is more easily stored and managed.

Errors in the BTM might carry over to the final mesh, so extra care should be taken in its preparation. Another part of data preparation consists in the integration of the topography and bathymetry in a common data set. A good example showing the problems that arise in the integration of different digital terrain models can be found in [GW02]. If there are additional topographic object in the area, such as very small channels, hydraulic structures (e. g. bridges, weirs), or artifacts of urbanization that do not find adequate representation in the data, the terrain model should be adapted to contain these objects.

For purposes of geoprocessing operations in this thesis, the BTM is a provider for elevation information. It can be queried for point elevations in arbitrary regions inside the study area, independent of the underlying terrain representation. Regardless of its representation, the BTM can be regarded as a kind of coverage.

The rasterization step of the terrain data takes as input an unstructured point cloud and produces a regular raster of cells. Typical horizontal raster resolutions for practical application in hydrodynamic modeling range from $< 1$ m to $\approx 10$ m. The rasterization is essentially an interpolation of cell elevations from the BTM using some interpolation method. Popular methods for interpolating from a terrain include nearest neighbor interpolation, inverse distance weighting (IDW) interpolation, and bilinear or bicubic interpolation. The bilinear and bicubic interpolation methods generally give the best results. However, they require that the BTM is triangulated locally.

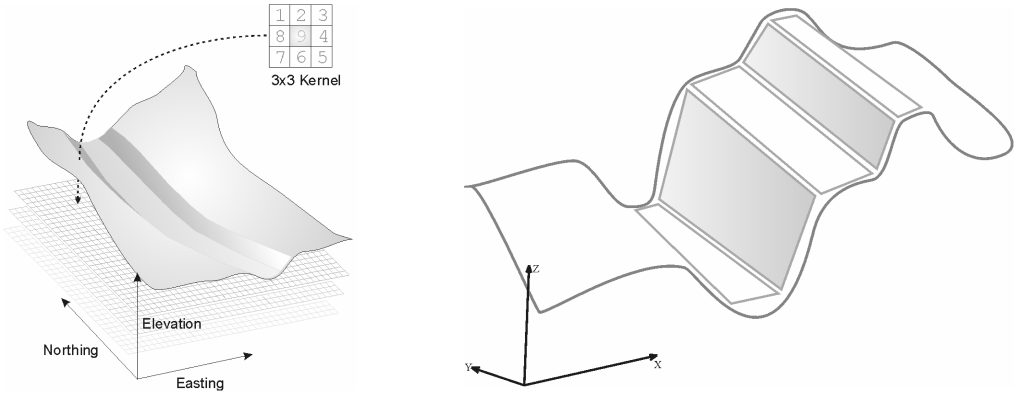
Processing of terrain data becomes increasingly difficult as airborne and spaceborne measuring systems deliver larger data volumes of tens to hundreds of Terabytes. This bears great potential to employ the computational and storage resources available in a grid. Without novel technologies it will be impossible to work with these data sets. The OGC Web Coverage Service has support for both structured and unstructured coverages. It is suggested that WCS is used for serving the BTM. The advantage of this approach is that WCS decouples the implementation and storage of the BTM from the subsequent geoprocessing operations. It also allows dynamic tiling and resampling of the BTM for partial processing, which will be advantageous for the gridification.

### 4.2.4. Detection of Structural Features

Structural features of a terrain denote locations where the topography has a sudden interruption, a local minimum/maximum, or a similar phenomenon that is essential to the correct representation of the terrain. The most common type of structural features are breaklines. They are located where a change in the slope occurs, e. g. at river banks or dikes. Elevation isolines (contour lines) are another example of structural features with application in the field of hydrodynamic modeling as flow at river banks typically occurs along lines of the same elevation. In urban settings, structural features have the form of building footprints, curbs, or walls.

Many authors have performed automatic feature detection in high-resolution terrain models obtained from laser scanning data with a focus on breaklines relevant to hydrodynamic simulation [Kha09; BH+08; Cog08; Rat07; BMH03; CM+03; Hor00]. These feature detection methods either work directly on the filtered LiDAR data or on an intermediate raster-based DEM representation. This thesis focuses on a raster-based approach similar to [Rat07].

Detecting breaklines in a raster DEM correspond to the detection of edges in an intensity image (see also [PM08; Can86; Der87; Ber87; SB97]). By the application of a gradient filter to the raster, it is possible to derive the surface slope (magnitude of inclination) and aspect (which direction the slope is facing). Due to noisy data, the raster is usually




**(a)** Applying a surface gradient filter to the raster DEM.

**(b)** Classification of slope area boundaries as breaklines.

**Figure 4.4.:** Breakline detection by edge filtering. The images have been taken from Rath [Rato7, pp. 73 and 76].

smoothed as part of this step. Breaklines are then classified, for example, as the boundaries of slope areas exceeding an experimentally determined surface gradient threshold (compare Figure 4.4). Taking the gradient of the slope magnitude image yields the local curvature of the surface. Other breakline detection approaches find lines of maximum local slope or maximum curvature via non-maximum suppression [Can86]. For purposes of this thesis, any thresholding approach will work. The detected breaklines then need to be vectorized and simplified.

In the subsequent triangulation step, it is possible to include additional structural lines into the mesh. This allows the modeler, e. g. to separate areas of different roughness characteristics, e. g. different vegetation or soil types. Attention must be paid to the proper geometric integration of such external lines with the detected breaklines, however, or disadvantageous elements may be created in the triangulation step. The integration of linear features from heterogeneous sources is a problem by itself that will not be detailed here. Geographic information systems often provide tool support for this task.

### Analysis of the Pre-Processing Phase

The pre-processing phase is dominated by tasks related to the creation of a discretization mesh from digital terrain data. The modeler must first manually discretize the study area boundary and acquire all topographical data required for creating a base terrain model that completely covers the study area. The latter task may involve providing the

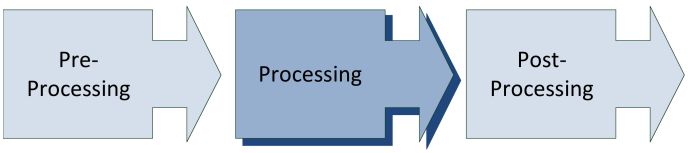
BTM in the form of a Web Coverage Service. The automatic mesh generation process consists of nine consecutive geoprocessing operations on the BTM, as can be seen in Table 4.1.

**Table 4.1.:** Sequence of geoprocessing operations for the pre-processing phase.

| | |
|---|---|
| I | Interpolate a regular raster from the BTM, e. g. by querying a WCS. |
| II | Apply a noise reduction filter to the raster. |
| III | Apply a gradient filter to the raster. |
| IV | Classify breakpoints according to a given threshold criterion. |
| V | Vectorize breakpoints to form breaklines. |
| VI | Simplify the breaklines. |
| VII | Create a quality constrained Delaunay triangulation of the breaklines inside the study area boundary. |
| VIII | Interpolate elevations from the BTM to the mesh. |

All of these operations can be chained and executed automatically in a computational grid. Steps I through IV can easily be parallelized because geoprocessing is inherently parallel on a spatial decomposition (tiling) of the BTM. However, care has to be taken when processing large volumes of terrain data, so that unnecessary data replications are avoided. Chapter 6 presents a geoprocessing workflow for the pre-processing phase.

## 4.3. Processing – Definition and Simulation of Flood Scenarios




Processing refers to the second phase in hydrodynamic modeling. Flood scenarios are defined based on a hydrodynamic model discretization using the finite element mesh created in the pre-processing phase. They require additional information: flow resistance, boundary conditions, initial conditions, and control parameters. Chapter 7 will give a demonstration how the simulation of flood scenarios is implemented as a geoprocessing grid service.

### 4.3.1. Flow Resistance

Flow resistance due to turbulence and bottom friction depends largely on the surface material properties as well as the type and height of the prevailing vegetation. These parameters are typically based on field measurements, aerial photography, as well as land use, cadastral, or geological maps. Areas with the same characteristics are assigned the same set of flow resistance parameters, or class. Recent research has focused on automatically deriving a flow resistance classification from topographic data acquired using remote sensing methods [Rato7; MC+03]. It may be argued that this task is part of the pre-processing phase.

In the numerical model, flow resistance is included in the form of a bottom shear stress. It may be expressed, for example, using equivalent sand roughness coefficients or by the approach of Gauckler, Manning and Strickler [Gau67; Man91; Man95]. The flow resistance parameters are typically specified either uniformly distributed over the region, a very simple approach, or on elements of the mesh. The latter can be automated if a map of the flow resistance classification (a coverage) is available. The step then consists of a spatial intersection or a weighted overlay of the classification onto the mesh. Flow resistance data could be given in the form of a Web Coverage Service. CORINE[1] land cover data is readily available and may serve as the basis for a classification.

### 4.3.2. Boundary Conditions

Boundary conditions represent the course of a flood event. They are additional constraints on the shallow water equations, which lead to a mathematical boundary value problem. Boundary conditions should be chosen so that the boundary value problem is well-posed, i. e. there exists a unique solution to the problem. Typically, some sections of the boundary are assigned water level boundary conditions, and other sections are constrained by discharge boundary conditions, while at the remaining boundary the so-called no-slip condition is applied. The no-slip condition implies that the flow velocity at the boundary is zero. This is physically justifiable because adhesive forces between water and the boundary surface are stronger than cohesive forces of the fluid.

The unsteady SWE are of parabolic or hyperbolic character, depending on the flow state (subcritical or supercritical flow). This influences the number and kind of boundary conditions that lead to well-posed problems. As a fluvial flood typically goes from upstream a river to downstream, the model needs to be sufficiently large so that the flooding process is not biased towards the downstream model boundary. Storm surges

[1] CORINE stands for Coordinated Information on the European Environment.

in estuaries, on the contrary, get their main influence from the downstream boundary, whereas the upstream discharge only plays a minor role.

The boundary conditions are specified at the mesh boundary in the form of water stage or discharge hydrographs, i. e. time series of discrete flow properties often measured by sensors. It is possible to apply the OGC SOS for accessing hydrological measurements and even artificial hydrographs, as they are used in future flooding scenarios.

The water level and discharge time series for a flood scenario may come from stream gauging station data or hydrological forecasts. A lot of historical data for rivers in Germany is provided by the hydrological information system PEGELONLINE[1] and the Electronic Waterway Information System (ELWIS)[2] operated by the German Federal Waterways and Shipping Administration (WSV). The time series are usually available in a higher resolution than needed for the simulation, so they have to be clipped to the necessary interval and filtered in order to reduce the data volume.

Once a hydrograph is available, it has to be applied to the boundary nodes or distributed to a section of the mesh. Or, thinking in reverse, the availability of hydrographs for certain locations could be queried in a catalog of data services.

### 4.3.3. Initial Conditions

Initial model conditions form a plausible starting flow pattern for the flood. Some models support dry starts, where there is, initially, no flow in the system. As a general rule, though, models require a more or less realistic initial flow situation for a stable execution. An often used possibility, termed *lake solution*, is to start with a completely wet domain and slowly lower the water level by adapting the boundary conditions to reach a quasi-steady flow state [Kin93]. In more complicated flow regimes, e. g. tidal rivers, this procedure is not going to produce a realistic result. Instead, an artificial tidal wave will have to be set as model boundary condition and many tidal cycles have to be calculated until the effect of the initial condition has been "washed out" of the system.

Obtaining an initial condition may require a considerable numerical effort. This step may be automated by setting appropriate boundary conditions that produce a desired starting flow state and calculating the lake or tidal solution beforehand. Another option is to keep a repository of pre-computed flow states that can be queried. The OGC Web Coverage Service has an extension for temporal data sets suited for this purpose. This solution decouples the providers and consumers of initial conditions and could be used

[1] http://www.pegelonline.wsv.de
[2] http://www.elwis.de

as part of a virtual organization sharing flood simulation results in a grid. The initial conditions would have to be overlaid onto the mesh prior to a simulation.

### 4.3.4. Control Parameters and Model Calibration

The hydrodynamic simulation is partly controlled by empirically determined parameters, which influence numerical aspects of the model, such as the temporal discretization and timespan, the flow resistance characterization, a turbulence model, or wetting-drying procedures. The simulation of a flood scenario requires a calibrated hydrodynamic model. The calibration process requires a series of simulations, in which the control parameters are varied. These simulations merely have the goal of finding the best approximation of a known (observed) scenario.

More often than not, the effects related to turbulent and dispersive shear stresses due to time- and depth-averaging of the Navier-Stokes equations are summarized into a single term. The magnitude of the mentioned effects depends on the spatial and temporal (in-)homogeneity of the flow field and the quality of its discretization in the mesh. For this reason, the shear stress parameter can only be guessed or taken from experience. Similarly, the quantification of bottom shear stress is error-prone because it depends on an accurate representation of vegetation and soil parameters. Furthermore, vegetation is changing with the seasons and over longer time periods, which adds to the uncertainty in roughness classification.

### Analysis of the Processing Phase

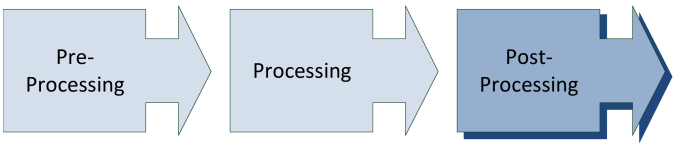
The tasks in the processing phase deal with setting up a flood scenario based on a previously generated mesh. In detail, a set of flow resistance parameters, initial and boundary conditions have to be applied, and the hydrodynamic simulation has to be executed. A possible sequence of geoprocessing operations for this phase is shown in Table 4.2.

OGC data services could be used to store and query a flow resistance classification, boundary conditions, and simulation results. Especially task V, the simulation, is a complex geoprocessing operation possibly taking hours or days to finish, and producing large amounts of output data. It will be shown in Chapter 7 how the actual simulation can profit from execution in the grid.

**Table 4.2.:** Sequence of geoprocessing operations for the processing phase.

| | |
|---|---|
| I | Overlay flow resistance onto the mesh. |
| II | Filter boundary condition time series to fit the simulated time span. |
| III | Apply boundary conditions to mesh boundary sections. |
| IV | Overlay initial conditions onto the mesh. |
| V | Run the hydrodynamic simulation. |

## 4.4. Post-Processing – Evaluation of the Results




Once a flood scenario has been successfully simulated, the outputs of the hydrodynamic model have to be evaluated. As the numerical model computes a vast amount of raw output data, the relevant information has to be filtered and transformed to a format suitable for analysis and exchange in a GIS. Post-processing reduces the total amount of model output data and derives information that can be presented to the public, e. g. as a flood map.

### 4.4.1. Characterization of the Results

The unsteady hydrodynamic model calculates discrete values for the absolute water level, model-relative water depth, flow velocity, and flow direction at all mesh nodes and for each simulated time step. All results are time-dependent coverage data sets. Some extended, or hybrid, models provide additional properties, such as the concentration of a solute or bed evolution, i. e. a change in the model topography due to sediment transport. These are usually not of primary interest to flood modelers, but may play a role in some special cases.

### 4.4.2. Creating a Flood Map

The goal in this thesis is to support the flood mapping process, so only the geoprocessing tasks relevant to creating a flood map are considered: deriving the flood extent and a

flood hazard map.

Using model-relative water depths for flood maps directly introduces some problems: flooding depths are only available at the mesh nodes, the mesh elevation at these nodes is most likely different from the elevation in a more detailed terrain model, and the actual flood extent, one of the most important results, is yet unclear.

In order to derive the real flood extent, both the water surface and a reference terrain elevation are required. If the flow model does not calculate near-subsurface flow, i. e. water depths in the transition zone between wet and dry areas, the water surface has to be extrapolated into the dry areas. The flood extent can then be determined as the three-dimensional intersection line of the extrapolated water surface and the terrain surface.

Alternatively, a flood depth map can be created based on the BTM (see Section 4.2) or another detailed terrain model. The flood depths are then calculated as the difference between the water surface elevation and the BTM at a number of sampling points. Typically these sampling points are given by the measurement points or raster cells of the more detailed model. The flood extent can then be found as the zero-contours in the difference surface.

Flood extent and flood depth maps could be created for each simulated time step. For mapping purposes, however, it proves to be more meaningful if only the maximum flood depth and, possibly, the inundation time is shown. This information has to be collected from all time steps. It is less common to show how the flood extent changes, although this knowledge might be useful for a-priori emergency management and evacuation planning, or for a-posteriori reconstruction of the course of a flood event.

### 4.4.3. Flood Hydrographs and Profile Sections

Sometimes it is desired to know the course of a flood at a specific location in the study area over the simulated time. Most often, this includes the water surface elevation at a point, the longitudinal profile of the water surface at the river central line, or the discharge through a cross-sectional profile.

In the multi-dimensional result coverage, hydrographs are linear sections across the time dimension, while profile sections are linear sections across the space dimensions. Combinations of both are possible, e. g. time series of longitudinal water level sections for projecting two-dimensional results to a single dimension.

Hydrograph locations and line sections do not necessarily have to correspond to node locations in the mesh. In that case, it is required to interpolate values from the results.

### Analysis of the Post-Processing Phase

Flow model results of an unsteady flow simulation can be regarded as a multi-dimensional data set with time as an additional dimension. As already noted in Section 4.3, the OGC Web Coverage Service is well-suited for results management. Post-processing tasks work on the outputs of the hydrodynamic model and create derivative, typically simpler, products. The operations shown in Table 4.3 belong to the post-processing phase. Some of these help to automate the flood map creation process.

**Table 4.3.:** Sequence of geoprocessing operations for the post-processing phase.

| | |
|---|---|
| I | Extrapolate simulation results into dry areas. |
| II | Find the flood extent as the zero-elevation contour line of the difference coverage. |
| III | Get the maximum result coverage of a flood simulation or a coverage representing the duration of flooding. |
| IV | Calculate the difference between a water level coverage and a terrain model. |
| V | Calculate the flood hydrograph at a given location or across a given section. |

Task I is required if the model results do not extend to dry areas, e. g. if the model computes water levels only at wet nodes, which produces a ragged, raw flood extent. II derives the flood extent for the corresponding flood depth coverage. In this way, the data volume can be massively reduced.

The geoprocessing operations III and IV, in this order, serve to create a flood map of maximum water depth. If applied to all time step results of a flood scenario, operation IV creates a time-dependent flood hazard map, which could be used in emergency planning.

# 5. Grid Services for Geoprocessing

The objective of this chapter is to develop a method of bringing geoprocessing operations into the grid in the form of modular geoprocessing grid services. In Chapter 4, the claim has been made that any kind of hydrodynamic modeling task can be represented as a sequence of geoprocessing operations. As the amount of available geographic data for hydrodynamic modeling increases, performing an analysis of these data becomes increasingly challenging. The idea is compelling to apply grid computing for this purpose. Geoprocessing services in the grid could use the power of many resources to efficiently process these vast amounts of geodata.

The challenges of conforming to both grid standards and geospatial standards related to hydrodynamic simulation will be described in Section 5.1. A solution to the problem is given by a framework for grid service implementation of the WPS, the de-facto standard for geoprocessing. Based on this framework, which will be called *Grid-WPS* (Section 5.2), geoprocessing services for the pre-processing, processing, and post-processing phases of hydrodynamic modeling can be implemented in the grid.

In effect, a Grid-WPS delegates resource-intensive geoprocessing operations to computing resources in a grid. Even though there is a considerable overhead in the proposed interface and its abstractions, Grid-WPS is designed not to impede the development of applications that process large amounts of data. The solution further promotes interoperability and reuse of geodata and geoprocessing components in the flood modeling community.

## 5.1. Challenges of Geoprocessing in a Service-Oriented Grid

This section gives an analysis of the problems that arise when geographic data are to be processed in the grid and shows the state-of-the-art in this field.

Since the first mention of the term "grid" in the context of distributed systems [FK+02], there have been many endeavors to combine the advantages of geographical services and grids. Some provide (partial) solutions to the main challenges, which include

- distributed management and processing of large volumes of geodata,

- improved Quality of Service for spatial analyses (e. g. performance, scalability, and security),
- geoprocessing and grid service orchestration, and
- bridging the technological gap between geoprocessing and grid services.

### 5.1.1. Distributed Management and Processing of Geodata

Geodata often reside in GIS databases or as raw files on a server. Spatial databases, in addition to regular SQL queries, provide optimized spatial indices and queries or functions that act on the geometry of spatial objects. In this manner, spatial databases have the limited capability of processing geodata as part of a query. Raw files can have an associated spatial index to the contained objects. A well-organized structuring of a data set in the form of many files (e. g. Shapefiles[1] or raster tiles) may also simplify access to a subset of that data, depending on the type of processing request that will be performed. It is possible, for example, to tile the data set or to split the data by equal attribute values. In this thesis, such efficient low-level storage technologies are taken for granted.

Large amounts of geodata (in the order of terabytes and petabytes) are usually distributed, and possibly replicated, across storage servers in multiple administrative domains. For this reason, managing and processing this data requires specialized technologies, which can be provided by OGC data services (WFS, WCS), grid services, as well as appropriate catalogs and metadata.

One of the earliest results in this field were shown by Panatkool and Laoveerakul [PL02]. The authors investigated distributed data and processing services even before the first web services for geographic data were standardized by the OGC, and only shortly after service-oriented architectures started to evolve. At that time, geodata was mostly processed in systems connected by a central database containing all of the required data. This approach bore the risk of a single point of failure and made data integration difficult. Because data was provided by different geographically distributed authorities, there was the need to decentralize data management and processing. SOAP web services and GML were the technologies of choice to create a grid of distributed geodata services. Security was considered an important aspect that the Globus Toolkit grid middleware could provide.

Similarly, Di et al. [DC+03] applied OGC standards and Globus Toolkit 2 to develop the so-called *NASA Web GIS Software Suite (NWGISS)*, an application that integrated the WCS with data storage and job submission capabilities of a grid. They saw a WCS

[1] Esri Shapefile is the de-facto standard for storing spatial vector data.

query as a kind of processing request for coverage data in the grid. The queried server either delegated the processing request to the server that owned the needed geodata files or obtained these files via an internal grid file transfer. In a subsequent publication [DC+08], additionally, developed a grid-enabled version of the OGC web service catalog using Globus Toolkit 4 grid services. The catalog contained metadata about a number of WCS instances in the grid. A portal served as the main entry point for regular OGC clients. All interactions with the grid services was handled by the portal.

Access to databases and other data sources can be realized using Open Grid Services Architecture for Data Access and Integration (OGSA-DAI). OGSA is an open source framework for accessing and integrating distributed data resources based on grid services (e. g. relational or XML databases, files, or web services). Examples for its use can be found in [MRS07]. Using OGSA for geospatial data sharing using for distributed query processing technology was first mentioned by Shu et al. [SZZ06]. The combination of these technologies and OGC standards was supposed to "facilitate the distributed management" of geospatial data "in a controlled and secure manner". For this to happen, grid technologies needed to become aware of the spatial dimension and OGC web services had to be grid-enabled. The the SEE/SAW-GEO projects[1] at EDINA, University of Edinburgh, have extended OGSA with support for geospatial data services according to the WFS and WCS specifications [KH08]. The projects filled a missing gap in the work of [SZZ06]. Requests to OGC data services could now be executed in a pipelined OGSA workflow. They created a prototype that incorporated an activity for "spatial data processing" into the workflow.

### 5.1.2. Improved Quality of Service for Spatial Analyses

OGC specifications have been found to provide a good basis for developing a service-oriented geoprocessing architecture within a spatial data infrastructure. Grid computing technology, on the other hand, promises an improved Quality of Service (QoS), e. g. performance, scalability, and security.

In addition to distributed management and high performance, Li et al. [LC+04] see a benefit of the grid in providing a high QoS by load balancing through dynamic resource discovery. Their motivation is to increase the efficiency of geoprocessing for spatial analysis. They implemented a web-based portal and a load balancer and request broker based on the Globus Toolkit 3 Monitoring and Discovery Service [CF+01] to give clients access to redundant geodata services in a Virtual Organization. Equivalent OGC-compliant data services in a Virtual Organization (i. e. WFS and WCS)

[1] `http://edina.ac.uk/projects/seesaw/seegeo`

registered themselves to an index service and could update current QoS information dynamically.

Research performed at University of Twente, Faculty of Geo-Information Science and Earth Observation (ITC), lays out the interesting idea of coupling mobile agents and grids to solve GIS problems in a distributed computing environment [GSW05; Ghi05]. They use the term *geoinformation services* to denote geoprocessing and geodata services. The integration of such grid-enabled geoinformation services into a standards-compliant spatial data infrastructure is seen as a substantial improvement when processing large data sets. Moreover, mobile agents are supposed to reduce network traffic by "flexible relocation" of geoinformation services. Computing resources in a grid are seen as a service execution facility, i. e. services are distributed across computing resources. This deployment model is currently not supported in typical grid infrastructures. Grid resources can only be reserved for a specified job and for a limited time. Likewise, Zhang and Tsou [ZT05] suggest an architecture that unifies the grid, a geospatial semantic web, and mobile agents. The foundation is a grid infrastructure for high-performance geospatial analysis based on geodata services. In practice, it remains unclear how these data services are enabled to use the existing grid infrastructure, e. g. as provided by the Globus Toolkit 3 grid middleware.

Díaz et al. [DGG08] specified a suite of OGC-conforming geoprocessing services with application in the domain of hydrological modeling. The WPS was used whenever complex geoprocessing operations needed to be performed. In particular, they developed basic services for topological overlay operations (e. g. intersection, buffer), image processing (e. g. vectorization, iso-elevation zones), creating charts, coordinate transformation, and data transformation (e. g. Shapefile to GML). Although this work does not make use of grid technology, the results are a valuable proof-of-concept that using WPS is a promising step towards modular and interoperable hydrologic geoprocessing services in a spatial data infrastructure. The results support the approach taken in this thesis. Furthermore, Friis-Christensen et al. [FO+07] highlighted the asynchronous communication capability of the WPS as vital to providing long-running processes to the user.

Hybrid (public and private) clouds may also be a solution to the problem of complex geoprocessing models with high computing loads due to an increasing number of geoprocessing requests. A private cloud could be extended dynamically — to meet peak demands — by using a commercial, public cloud service. The issues addressed by the utilization of cloud computing are [BSR11]:

- Efficient use of available hardware by dynamic increase of processing capacities,
- outsourcing of hardware and software to third-party providers on a pay-per-use revenue basis, and

- automatic, on-demand scaling of a business processing infrastructure with a guaranteed QoS.

For example, [FB+10] employed a load balancer that could forward requests to one of multiple virtual, redundant WPS service instances running in the cloud.

### 5.1.3. Orchestrating Geoprocessing and Grid Services

Geoprocessing workflows are a method to represent complex geoprocessing tasks as a composition of simpler geoprocessing operations. [FO+07] demonstrated a distributed geoprocessing workflow for a fire damage assessment use case. Furthermore, a workflow for the pre-processing, processing, and post-processing phases in the hydrological and hydrogeological domains (see Chapter 4) have been investigated in [HOS08].

The orchestration of OGC and grid services together is limited due to missing interoperability of the respective standards. [HFJ07; HF+10] stated that a solution to the problem is the creation of standard web service interfaces to OGC services (i. e. using SOAP and WSDL) and using a standard workflow description (i. e. Business Process Execution Language (BPEL)) for service orchestration. The authors adopted a WSRF interface for grid services, but do not name the advantages thereof. The problems of orchestrating stateful WSRF grid services and the characteristics of the WPS in BPEL workflows are not addressed either. Appropriate BPEL workflow patterns for the orchestration of stateful, secure grid services have been developed in [Has10].

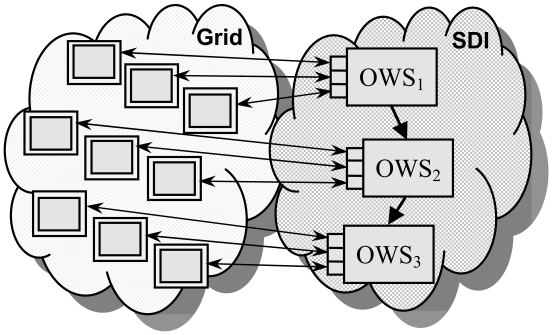
Krüger and Kolbe [KK08] dealt with the question of how a geoprocessing workflow, or service chain, using geoprocessing and geodata services in a spatial data infrastructure (denoted OWS on the right in Figure 5.1) is practically executed on grid resources in a grid infrastructure (boxes on the left). Three main approaches for mapping OGC web services to grid resources were identified:

1. The grid is used for the operation of an individual geoprocessing service. Multiple independent grid resources run grid service instances or grid jobs to process different parts of geodata in parallel. It is the responsibility of the geoprocessing service to distribute the work to the available grid resources and gather the results. There is no direct flow of data between grid resources (see Figure 5.1a).

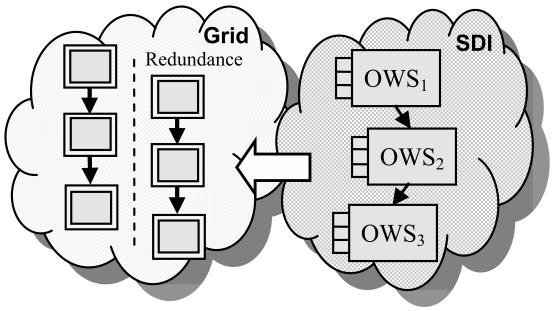
2. A geoprocessing service chain is mapped to a chain of grid services. Several instances of this chain can be executed in parallel for separate sets of data. There is no flow of data between geoprocessing and grid services, which implies that it is impossible to mix grid services and regular WPS in the same chain. It is not even necessary that there is a one-to-one mapping between WPS and grid services for this solution (see Figure 5.1b).

3. Grid services are directly associated with a WPS. A chain of WPS maps directly to a chain of gridified WPS. Data flow can occur either between grid services, between a grid service and a WPS residing outside the grid, or between WPS. In this most complex solution, it is possible to combine gridified and non-gridified WPS in a single service chain using the fast data transfer capabilities of the grid (see Figure 5.1c).

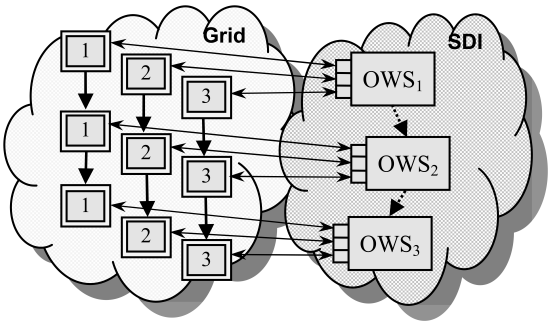
This thesis employs the third approach, which promises the highest degree of flexibility and performance.




**(a)** Using the grid only for geoprocessing calculations.




**(b)** Mapping a geoprocessing service chain to a grid service chain.




**(c)** Using the grid for calculations and data transfer.

**Figure 5.1.:** Three approaches for mapping OGC web services (OWS) in a spatial data infrastructure (SDI) to grid resources in a grid infrastructure [KK08, pp. 1562-1563]. Both the data flow between geoprocessing services and the data flow between the two infrastructures are displayed.

OGC web services and GML were previously used to build a scalable, distributed system for the investigation of earth system problems (iSERVO), such as coupling numerical simulation models and online observational data [AA+05; AA+06]. Data

flow between services was based on file exchange, while services were developed with standard SOA technology (SOAP/WSDL) and could be orchestrated by a simple workflow service. iSERVO integrates aspects of both computing and data grids. A parallel application was deployed in a computing cluster and wrapped by a single web service.

### 5.1.4. Bridging the Technological Gap

One of the major problems lies in the harmonization of the standard for grid service development, the WSRF (see Chapter 2), and the OGC web service standards, in particular the geoprocessing standard WPS (see Chapter 4).

Baranski [Baro8a; Baro8b] coined the term "gridification" as the adaptation of existing geoprocessing applications to the requirements and expectations of a grid environment. He suggested two different levels of gridification:

1. an OGC interface is used as a wrapper to an implementation using the grid in the back (low-level gridification), or
2. fully integrating an OGC service into the grid middleware as a grid service with a standards-compliant OGC proxy service in front (high-level gridification).

The author only implemented the low-level gridification approach reaching the conclusion that there is no generic way of providing an OGC grid service conforming to the original OGC standard. The implementation had the disadvantage that data had to be transfered to and from the grid. The second approach was successfully demonstrated by Hobona et al. [HFJ07] and Foerster et al. [FS+11]. Padberg and Kiehle [PK09b; PK09a] focus on the extension of an existing spatial data infrastructure through the use of grid infrastructures. They suggest to implement the WPS standard as a grid middleware service with an OGC-compatible proxy service. Not conforming to OGC interfaces on the grid service level was not considered to be a hindrance.

In this thesis, a high-level gridification of the WPS is sought. Additionally, the grid service interface will be made OGC-compliant in the Grid-WPS framework (see Section 5.2).

Lee and Percivall reported on the collaboration between the two organizations for standardization in the grid world, the OGF, and in the geographic information community, the OGC. They identified the technologies from the geospatial and grid communities that have to be integrated in order to build a distributed geospatial system, i. e. key OGC and OGF standards. According to the authors, a major challenge is to hide the complexity of the grid and to simplify tools for use in the geospatial application domain.

The Global Earth Observation System of Systems (GEOSS) is a community that would benefit from such an integration effort [Lee08].

Krüger and Kolbe [KK08] made the main observation that OGC Web Services are lacking a standard security concept. They summarized many results of the German Spatial Data Infrastructure Grid (GDI-Grid) project, which gave a motivation to this thesis. The solution in [PK09b; PK09a] used a MyProxy service (see Chapter 2, Subsection 2.3.3) to obtain a grid certificate for access to the grid using user name and password credentials submitted as parameters of the WPS process execution. These allowed the WPS process to submit grid jobs and manage the required storage resources.

[LS+08a; LS+08b; LZ09] used WPS for processing large digital elevation models. They implemented a conventional WPS that was able to run grid jobs using Globus Toolkit 4. A grid certificate was also obtained from a MyProxy service. It remained an open question how storage resources are used to publish data in the grid for subsequent calculations. This problem was then addressed in [LK+09], which motivated the development of the terrain discretization service [KP09] that lead to the flow model discretization service (see Chapter 6) developed in this thesis. The Grid-WPS framework to be developed in this chapter will provide a solution to the problem of intermediate data storage in the grid.

An extension to the WPS standard, called the *Temporary Resource Store (TRS)*, was suggested by Keens [Kee06]. The author stated that WPS ought to consider adding support for stateful resource management to the WPS specification, such as it is defined in the WSRF. Resource management is, in effect, required for processing large data. He further suggests to introduce the operations `putResource`, `getResource`, `destroyResource`, and `setTerminationTime`. The `getResource` operation would be used to retrieve the output of a process and `putResource` would send a resource to the TRS, which returned a resource identifier that could then be used as an input parameter in the `execute` request. Resources in the TRS would be subject to a termination time determining when the resource was going to be destroyed automatically by the server. `setTerminationTime` and `destroyResource` allowed to control the lifetime of resources in the TRS. Such functionality, as suggested by [Kee06], could be implemented on top of the Grid-WPS framework to be described in Section 5.2.

### An Overview of Grid and Geoprocessing Standards

Many of the publications dealing with OGC standards in the context of grids deliver valuable information that can be applied to the gridification of OGC web services. The WPS specification is a very recent development, so only literature newer than about

2006 makes a direct reference to it, after the WPS 0.4 request for comments document [SG+05] was issued.

Krüger and Kolbe [KK08] presented an overview of different options for gridification of OGC web services. In their view, grid services are seen as an orthogonal, supporting technology to WPS. The authors make the implicit assumption that users in a spatial data infrastructure have no knowledge of grid technology. However, this assumption only holds for the first two approaches to gridification. The last option (on 66), on the other hand, which also seems to be the most promising one, requires a more integrative view on SDI and grid infrastructures. WPS and grid standards are going to be harmonized in the Grid-WPS framework that is part of both infrastructures.

It should be noted, however, that there is no standard for processing services in the OGSA issued by the OGF. In fact, most of the specifications mainly deal with grid infrastructure, execution and data management on computing clusters, service security, as well as resource management and information services [GM+09].

The OGSA is defined mainly by the OGSA WSRF Basic Profile [FMS06], the Job Submission Description Language (JSDL) [AB+05; AB+08], the OGSA Basic Execution Service (OGSA BES) [FG+07], the HPC Basic Profile [DH+07], which incorporates the JSDL HPC Profile Application Extension [HS+07] and the HPC File Staging Profile [WH08]. In the following table (Table 5.1), a comprehensive overview and comparison of the relevant features and standards of both WPS and OGSA grid services is given. They provide the requirements for the Grid-WPS framework that can co-exist in both a grid infrastructure and an SDI.

**Table 5.1.:** Comparison of the WPS and OGSA WSRF Basic Profile standards.

| Feature | Web Processing Service 1.0 | WSRF Basic Profile 1.0 |
|---|---|---|
| Interface | OGC Web Services Common Specification 1.1.0, KVP or XML encoding over HTTP | WS-I Basic Profile 1.1[a], WS-Addressing 1.0 – Core |
| State | WPS status document (optional) | WS-ResourceProperties 1.2, WS-ResourceLifetime 1.2, WS-BaseNotification 1.3 |
| Security | — | OGSA Basic Security Profile 2.0[b] |
| Discovery | OGC Catalogue Service for the Web (CSW) | Universal Description, Discovery and Integration (UDDI) Version 2 |
| Metadata | Capabilities document | — |

[a]WS-I Basic Profile 1.1 dictates a WSDL 1.1 interface description restricted to a SOAP 1.1 "rpc" or "document" style binding with "literal" message encoding and transport restricted to HTTP/1.1

[b]Including WS-I Basic Security Profile 1.0, Secure Addressing Profile 1.0, Secure Communication Profile 1.0, WS-Security 1.0 with X.509 Certificate Token Profile, WS-SecurityPolicy 1.2, WS-Policy 1.5 with Attachment

**Table 5.2.:** Comparison of the WPS and HPC Basic Profile (with HPC File Staging Profile) standards.

| Feature | Web Processing Service 1.0 | HPC Basic with File Staging Profile 1.0 |
|---|---|---|
| Operations | `getCapabilities`, `describeProcess`, `execute` | `createActivity`, `getActivityStatuses`, `terminateActivities`, `getActivityDocuments` |
| State model | — | `Pending`, `Running` (`Stage-in`, `Executing`, `Stage-out`), `Finished`, `Terminated`, `Failed` |
| Input/Output | In-line or by reference URI, results in response document or raw data | File-based JSDL Data Staging (source/target URI definition) |
| Credentials | — | User name / token or X.509 certificate |

## 5.2. The Grid-WPS Framework

To obtain a "real" geoprocessing grid service, a Grid-WPS framework will be developed in this chapter that conforms to both the WSRF and WPS standards. From the experiences gained in the previous section, it can be concluded that a harmonization of the WPS with grid service standards is only possible if its implementation is based on the widely accepted SOAP/WSDL style of building web services. The Grid-WPS is consequently designed as a stateful grid service with operation semantics defined by the WPS specification and conforming to standard web protocols.

The Grid-WPS framework is integrated with the Grid Security Infrastructure (GSI). Authorization is done based on user certificates and Virtual Organization membership. If the Grid-WPS accepts the delegation of X.509 proxy certificates, it can use secured grid resources on behalf of a client, i. e. for data transfer and job submission. Jobs are submitted over OGSA BES according to the HPC Basic Profile. The grid resources for execution are specified according to the Job Submission Description Language (JSDL) [LP08].

Instances of web services, grid services, and Grid-WPS must further be able to be orchestrated using a standard workflow language. BPEL is used for this purpose. Workflows must allow for an efficient transport of data between services, grid jobs, and the end user. Efficient end-to-end data flow for large amounts of data is realized using GridFTP and OGSA. By conforming to OGC standards, in particular GML and complex feature schemata, syntactical interoperability of services can be ensured, which favors their reuse. Out of scope is formalizing the semantics of geodata and geoprocessing operations, although such a feature would enable intelligent service lookup and the verification of workflows.

In order to gain additional benefits from the WSRF in the context of long-running geoprocessing operations, the Grid-WPS supports the delivery of status response documents and storage of outputs as web-accessible resources [FO+07]. WS-Notification [NG05] is used to notify about status changes asynchronously. This allows more flexible client software and interacting geoprocessing services to be developed, e. g. for coupling complex simulation models (see Chapter 7).

### 5.2.1. Harmonization of the WSRF and WPS Standards

As a solution to the problem of harmonization of OGC and grid standards, this section shows how to define the WPS specification as a WSRF grid service. The first step is to declare the WPS interface in WSDL with a SOAP binding. However, the WPS specification is not clear about how to construct a typed WSDL interface description

for a process. The Grid-WPS solves this problem by reusing the XML types defined in the WPS schema document to define the `execute` operation in WSDL. In this way, the original `execute` operation can directly be represented, so it is easy to adapt existing clients to work with the new interface.

This contrasts the example WSDL schema in the WPS specification in which only custom types are used to describe inputs and outputs. The specification suggests a more concrete implementation of `execute` for a specific process. It defines an `execute_ProcessID` operation specific to the offered process with the given identifier. The `execute_ProcessID` request message is restricted to the concrete inputs, and the `ExecuteResponse` contains only the concrete outputs of the process in question, which can be derived from the result of a `DescribeProcess` request.

Both of the interfaces generated are functionally compatible to the original `execute` operation. Dorka [Doro9] has developed a tool that automatically performs a translation to WSDL for an existing `ProcessDescription` document. Unfortunately, it is not possible to generate an interface that is both process-specific and, at the same time, message-compatible to the original `execute` request due to the way XML schema handles restrictions of an existing schema.

If the WS-Addressing extension to the SOAP protocol is supported, a geoprocessing grid service may be implemented as a WS-Resource, i. e. a combination of a grid resource and a service. A factory service is typically responsible for resource creation. Resources can be accessed via an *endpoint reference*, which contains an address URI and a resource identifier.

A WPS resource could be designed to have a number of properties according to the WS-ResourceProperties specification. In addition to providing the input parameters to an `execute` operation in the request and the outputs in the response, parameters may be mapped to resource properties. The identifier of an input or output parameter is used as the identifier of a corresponding resource property. Once a process is called, an input parameter can be taken from the current value of a WPS resource's property. Analogously, results can be stored in output resource properties.

Properties containing execution results can be queried via a standard call to the `get-ResourceProperty` operation on this resource. A client may even register for change events on this resource property using the WS-Notification standard. In this way, the pull-based mechanism of the original WPS can be converted to a push-based one. Especially clients calling a long-running geoprocessing task or submitting jobs to a computational grid will make use of such a possibility.

All Grid-WPS operations need to be secured, which requires a valid grid user certificate. This certificate can either be transmitted in the request using GSI or the service authorizes a user based on user name and password credentials. In turn, it then obtains

a valid grid proxy certificate from a MyProxy server. Authentication and authorization of the user in the grid can then be based on this user or proxy certificate. Additionally, a chain of trust is established by delegation from the user or proxy certificate over the Grid-WPS to any other grid service invoked.

### 5.2.2. Submitting Geoprocessing Grid Jobs

Many geoprocessing operations that work on big datasets require large amounts of processing resources. Most standard spatial analysis operations on feature collections, like buffering of feature geometries, are even inherently parallel, so they can be easily parallelized. However, the real problems in geoprocessing that benefit from grid computing either work on gigantic feature collections ($\geq 10^8$, the size of the OpenStreetMap road network[1]), involve raster datasets with a high resolution or a large extent ($\geq 10^8$ raster cells), or contain a temporal component.

Complex numerical simulation in the domain of water resources engineering, such as, for instance, unsteady two- or three-dimensional flow simulation, can be regarded as a geoprocessing operation itself. Secondly, a simulation also produces large amounts of output data to be analyzed. And, finally, the generation of meshes for the simulation requires the processing of detailed digital terrain models. Many of these operations are suited for execution as jobs in a computational grid. Typical grids combine the resources of several computing clusters, supercomputers, or single computers. Grid jobs run either a sequential program, meaning an independent, stand-alone task, or a parallel program with many dependent subtasks (see Chapter 2).

The Grid-WPS framework supports job submission to a Globus Toolkit-based computational grid, according to the OGSA BES, and is suited in the hydrodynamic modeling context. The framework covers the aspects of building a job description, preparing the input data, submitting the job to a Grid Resource Allocation and Management (GRAM) grid job submission service, monitoring its execution, and getting the results back. The hydrodynamic modeling domain has the need for large amounts of data and long-running processes, so the flexibility of using referenced or in-line inputs, supporting server-side storage of outputs, and the capability of delivering information about the process state are retained from the WPS specification.

At the beginning of a geoprocessing workflow, an OGC geodata service or file-based data source delivers geodata to a computing resource in the grid. Ideally, the source of data is already located somewhere in the grid as well, e. g. in the case of intermediate workflow data, so that efficient data transfer mechanisms (GridFTP and OGSA) can be used.

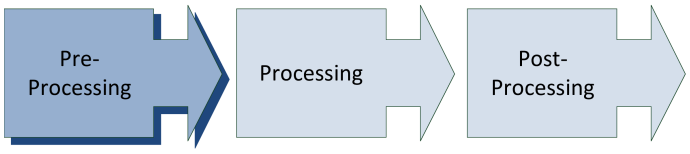
[1] `http://www.openstreetmap.org/stats/data_stats.html`, last visited in Aug. 2012

It is not viable to create copies of process inputs and outputs when handing data over from one geoprocessing operation to another. Fortunately, the WPS specification allows to specify references to input and outputs resources using a URI. Once such a resource is accessed as part of the geoprocessing operation, the Grid-WPS opens a stream to obtain the data. Using OGSA-DAI it is possible to deliver database records to a file in the working directory. Result files are written to the working directory as well, to another specified output directory, e. g. on a grid storage resource, or to a spatial database in the grid. One can think of a scenario where the results are put in a place that is dynamically served via an OGC web service. In case the Grid-WPS wants to deliver an output reference, it constructs a URI using the the GridFTP scheme from the address of the computing resource (or cluster head node) and the local path to the resource inside the working directory. End users may then obtain this resource using a GridFTP client or directly pass it on to a following geoprocessing service, e. g. as part of a workflow.

### Conclusion

Grid-WPS is a framework for the implementation of the OGC Web Processing Service as a grid service. The grid service implementation is based on the Web Services Resource Framework and can submit geoprocessing jobs to computing resources in a grid. By adhering to both OGC standards and grid standards, it is possible to orchestrate geoprocessing services in the same way as grid services using a grid-enabled workflow engine. At the same time, compatibility with wide-spread technologies from the geospatial community is retained.

Two concrete grid services will demonstrate the feasibility of applying the Grid-WPS framework to the domain of hydrodynamic modeling. The first one shows that large amounts of terrain data can actually be processed in the grid using a geoprocessing workflow. The second service deals with a complex, long-running geoprocessing operation. Both services aim at a parallelization of the respective process, but require different strategies to reach this goal. The Grid-WPS framework provides the necessary abstractions in each of the cases. Moreover, it promotes reuse of the developed geoprocessing operations and interoperability with spatial data and grid infrastructures.




# 6. Flow Model Discretization Service

The flow model discretization service developed in this chapter aims to provide support in the pre-processing phase of hydrodynamic modeling. This phase, as formalized in Chapter 4, Section 4.2, consists of data intensive geoprocessing operations that create a computational mesh for subsequent flood simulations from a digital elevation model (DEM). The service implements standard-compliant geoprocessing operations related to the meshing process by employing the Grid-WPS framework from Chapter 5. It allows to execute these geoprocessing operations in a computational grid.

The individual geoprocessing operations provided by the service are then incorporated into a geoprocessing workflow of calls to the Grid-WPS. Individual tasks of the workflow are executed in parallel on geographically distributed computing resources. This is made possible by partitioning the DEM and adapting the meshing process in a way that parts of the DEM can be processed independently.

First, Section 6.1 gives the motivations for developing a grid service for flow model discretization and defines the objectives of the prototypical implementation. The geoprocessing operations required in the meshing process have been implemented in a software library, Gaja3Dpar. This library, the parallel process, and two partitioning strategies are presented in Section 6.2. How Gaja3Dpar is interfaced with the flow model discretization service is explained in Section 6.3. Finally, Section 6.4 illustrates the flow model discretization grid workflow.

## 6.1. Introduction

The relevance of the prototype to be developed in this chapter lies in the elaboration of a methodology for parallel mesh generation and in the demonstration of a geoprocessing workflow using the Grid-WPS framework that implements this methodology for automatic execution in the grid.

### 6.1.1. Motivation and Objectives

Two-dimensional flood modeling requires a discretization of the river bathymetry and floodplain topography suitable for numerical simulation. Ideally, such a discretization is an unstructured mesh created automatically based on digital terrain elevation measurements. An automatic meshing methodology for flow model discretization and its representation as a sequence of geoprocessing steps with the goal of obtaining a mesh for computer simulation has been presented in Chapter 4. These steps are typically performed in a pre-processing phase of flood modeling, before the actual flood simulation can be conducted.

The amount of available data to be processed in this phase is massively growing as remote sensing has become the technology of choice to generate DEMs in a very high resolution. Computer simulation of 2D hydrodynamics is lacking behind in this development creating the need to simplify, or generalize, the available data without eliminating essential details that determine the flow paths in the terrain. Tool support in the pre-processing phase is of special importance for flood modelers, e. g. consulting engineers, who have to set up models for large domains. However, they do not always have access to the necessary computing resources to process such a large terrain model. This prototype shows that grid computing can be used to make these resources accessible to flood modelers. In addition, the provided implementation demonstrates that modelers can manage the associated programming interfaces because they conform to de-facto web and workflow standards.

An important step in the mesh generation process is the detection of structural features in a regular raster DEM (see Chapter 3). It has to be noted that structural features which are close together may only be detected in rasters with a cell width at maximum of half the distance of the features, so that they are separated by at least two cells. For example, in order to detect a trapezoidal channel with a bottom width of 2 m and a top width of 6 m, the raster resolution should be at most 1 m, preferably less, otherwise corresponding upper and lower bank breaklines might not all be derived correctly. If such a precision is desired, the amount of raster data that has to be processed grows massively. This is where a parallelization and automation of the process will be of great benefit.

### 6.1.2. Flow Model Discretization Use Cases

The flow model discretization service to be developed in this chapter is, essentially, a single geoprocessing grid service automating the pre-processing workflow. The individual steps that will be covered by the prototype are rasterization of the digital elevation model, breakline detection, and mesh creation by triangulation of breaklines

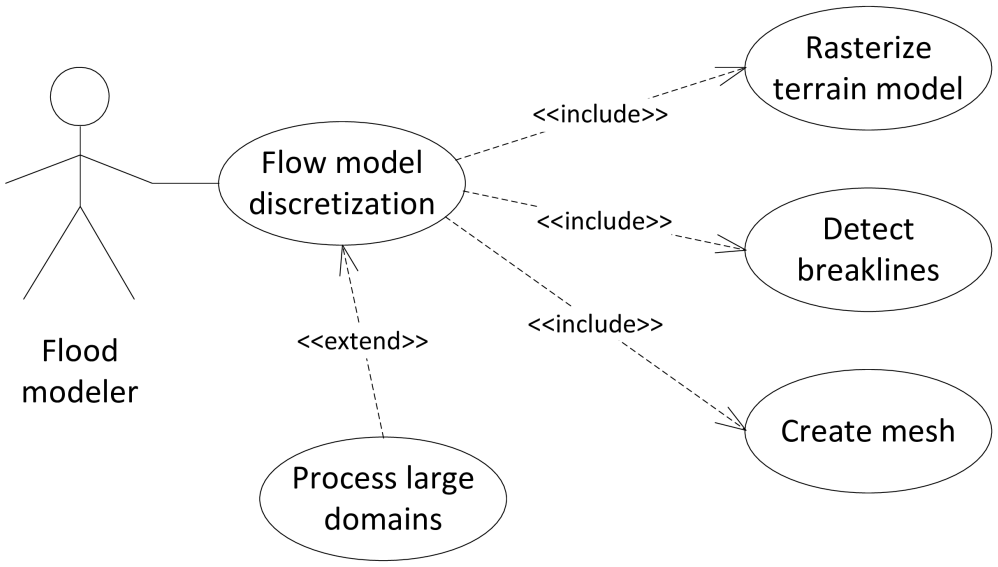



**Figure 6.1.:** Flow model discretization use cases. The contribution of this thesis lies in the support for processing large model domains using a parallel approach.

(see Figure 6.1). The use case elements contained in the displayed extension towards large model domains, which form the main contribution of this chapter to the overall process, are concretized in Figure 6.2 on the next page. Flood mapping of large areas requires massive elevation data to be processed. Especially, two-dimensional flow model discretization poses the problem of deriving structural features (breaklines) for the whole domain. Users who wish to discretize such models will have to split their model domain and DEM into suitable parts, process one at a time, and later aggregate (merge) the detected breaklines from each part to a coherent set. A parallel process for this methodology of discretizing a large domain will be presented in the following section.

## 6.2. A Methodology for Parallel Mesh Generation in the Grid

The methodology developed in this section is a parallel version of the process presented in Chapter 4, Section 4.2. It generates a 2.5D mesh for a polygonal region, the study area, from massive digital terrain data with constrained structural terrain features

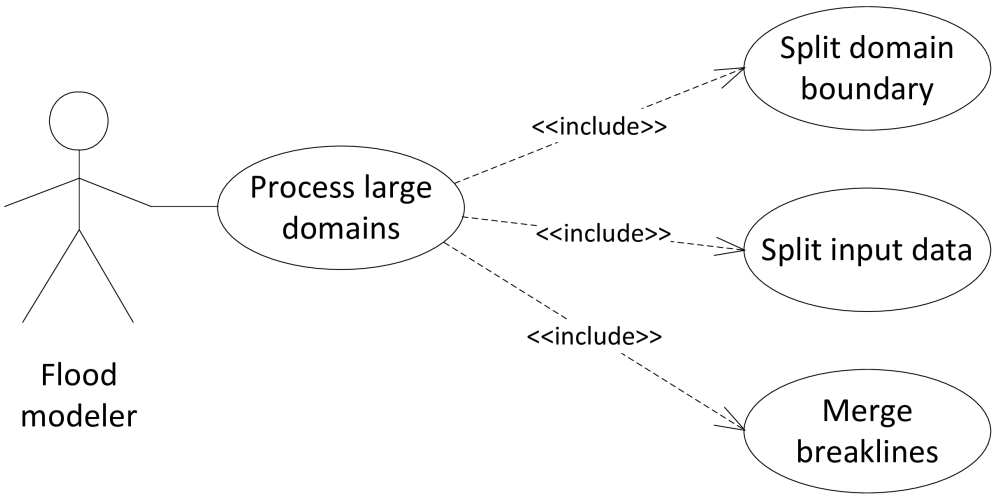



**Figure 6.2.:** Use case elements related to parallel flow model discretization for large domains.

(breaklines). Whereas the original methodology by [Rato7] performed breakline detection for the complete study area at once, the proposed methodology splits the study area and DEM data into parts, detects breaklines in each part, and later merges the results. The detected breaklines for all parts are finally used for the generation of a consistent global mesh.

The geoprocessing operations for mesh generation have been implemented as functional components in a software library called *Gaja3Dpar* (Subsection 6.2.1). Two algorithms for partitioning the study area have been developed as part of this thesis (Subsection 6.2.2). How a study area partition is used to split the DEM into parts is shown in Subsection 6.2.3. The presented methodology is inherently parallel on the parts of the study area if certain rules are obeyed when splitting the elevation data for the breakline detection process (Subsection 6.2.4). The final mesh is created by triangulating the breakline results from all parts and assignment of DEM elevations, which is explained in Subsection 6.2.5.

### 6.2.1. The Meshing Library Gaja3Dpar

Gaja3Dpar is a is a new software library for hydrodynamic model creation that forms a major result of this thesis. It was motivated by the Gaja3D meshing software developed

by Rath [Rato7] at Hamburg University of Technology, Institute of River and Coastal Engineering[1]. The original version of Gaja3D had not been designed with modularity and parallel processing in mind. The new library allows all intermediate mesh generation steps to be based on multiple, independently processed terrain patches for parts of a large study area.

Gaja3Dpar provides an API for easy MATLAB scripting using a single point of entry for the mesh creation process via the class `Gaja3D`. Users of the flow model discretization service will never see this API, but it is required to build the service executable. The `Gaja3D` class offers methods to set data (domain boundaries, DEM tiles) and to configure and perform the required discretization steps. The API also includes classes for working with vector data (`Curve`, `Polygon`), raster data (`RectifiedGridCoverage`), triangulations of scattered point data optimized for interpolation (`TriangulatedSurface2`), and the triangulated final mesh (`TriangulatedSurface`). For a class diagram see Figure A.1 and Figure A.2 in Appendix A.

Gaja3Dpar provides three scripts for the flow model discretization service. Each performs a short sequence of geoprocessing operations for the pre-processing tasks (see Figure 6.3):

- Raster creation from an unstructured digital elevation model (`CreateRaster`),
- breakline detection in the DEM (`DetectBreaklines`),
- mesh creation by Delaunay triangulation with interpolation of elevations from the DEM (`CreateTin`).

It was a design decision to bundle the tasks in this way. The motivation for this design has been to improve the performance through fewer data transfers of large terrain data sets between the individual geoprocessing operations when executed in the grid. As the first step in the mesh generation process, the study area needs to be partitioned.

### 6.2.2. Partitioning the Study Area

The study area (model domain) is a region delineated by a *boundary polygon*. This polygon constrains, in the first place, the outer mesh boundary. In the second place, it determines the relevant area of the DEM for assigning elevation information to the mesh and for the detection of structural terrain features in this area. A partition of the boundary polygon into smaller polygonal parts (tiles) determines the units of work for the parallel mesh generation process. For the purpose of this prototype, two geometric partitioning algorithms have been developed. They will be compared with respect to

[1] `http://www.tu-harburg.de/wb`

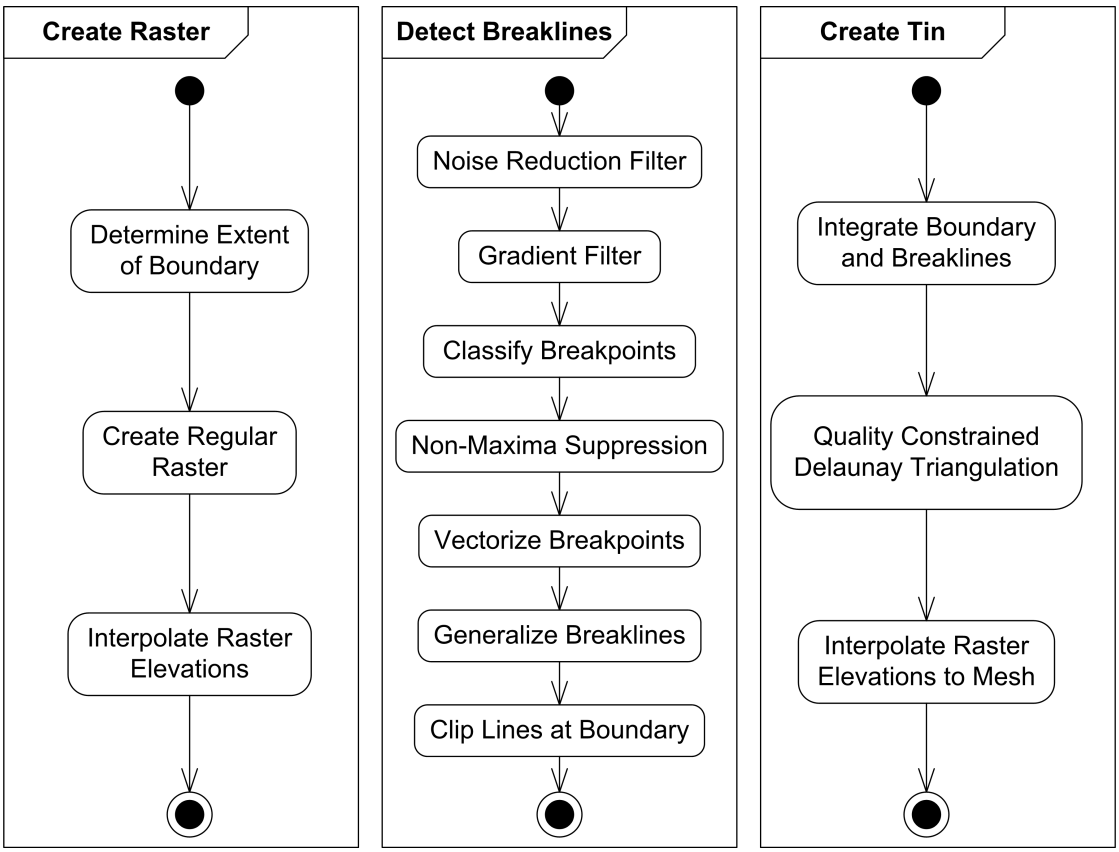
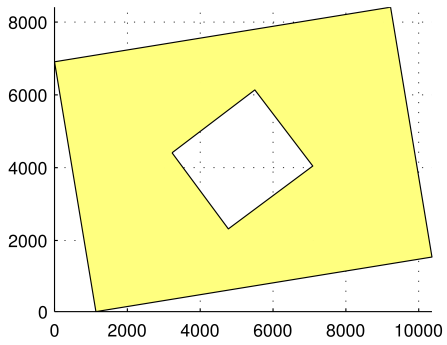



**Figure 6.3.:** Gaja3Dpar internal geoprocessing workflows of pre-processing tasks.

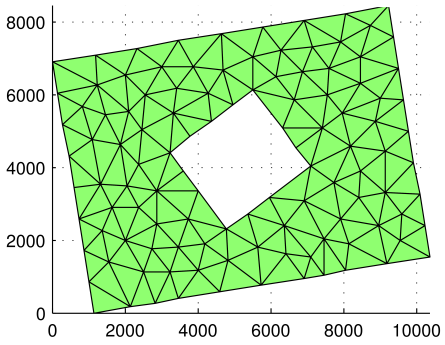
the total amount of overlap of the study area boundary polygon and balance of the partition.

The tiles will not necessarily be aligned to a regular Cartesian grid or even be rectangular. All tiles together completely cover the study area. The two partitioning strategies prefer "compact" tiles of similar area. Compactness means that the difference between a tile and its bounding box area should be small. This optimization criterion is equivalent to the minimization of the total area of all bounding boxes. The minimization problem for a given boundary polygon $P$ into a given number $i$ of tiles $P_i$, with $\bigcup P_i = P$, where $\bigcup$ is the geometry union operator, can thus be specified as $\sum E_{P_i}$, where $E_{P_i}$ is the area of the bounding box of $P_i$. The second algorithm to be presented will allow that the bounding box for a tile is rotated to better align with the corresponding tile boundary polygon. It will be shown that this algorithm is better suited to fulfill the optimization criterion.

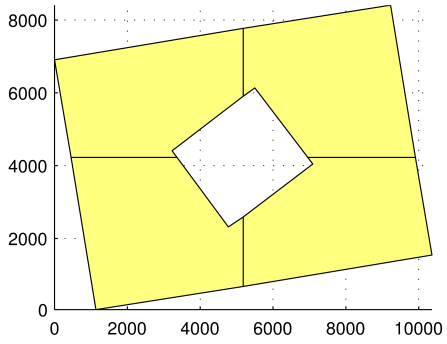
The study area boundary polygon is a set of simply connected regions, i. e. closed, not



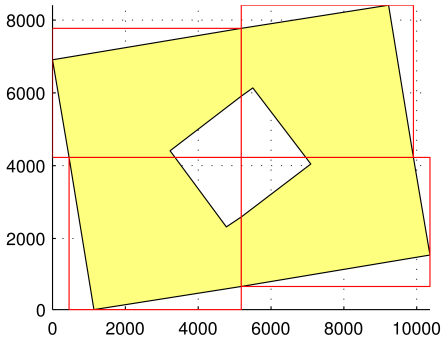

**(a)** The study area boundary polygon.



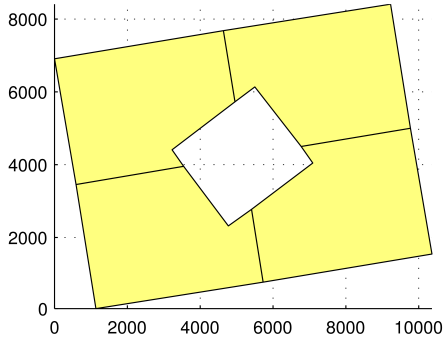

**(b)** The corresponding mesh for the polygon with its hole excluded.



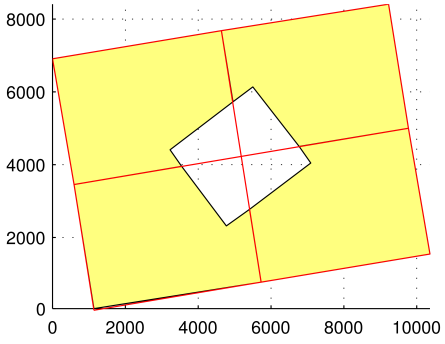

**(c)** Simple partition of the boundary polygon from the first algorithm.




**(d)** Regular bounding boxes of the tile polygons.




**(e)** Improved partition of the boundary polygon from the second algorithm.




**(f)** Rotated bounding boxes (identical to the tile boundaries).

**Figure 6.4.:** Demonstration of partitioning and meshing a small, artificial study area boundary polygon (58 $km^2$) containing a hole.

self-intersecting vertex chains. One of these regions delineates the exterior boundary of the model domain. All other regions are contained in the exterior boundary and represent holes in that domain (see Figure 6.4a). These regions shall be excluded from the generated mesh (see Figure 6.4b). The figures show a possible final result of the meshing process. This mesh has been triangulated without consideration of breaklines, but with 30° minimum angle and 0.5 $km^2$ maximum triangle area conditions. Figure 6.4c and Figure 6.4d show a simple partition, according to the first algorithm, and the respective bounding boxes for the artificial boundary polygon. The result of applying the second algorithm to the simple boundary can be found in Figure 6.4e and Figure 6.4f.

Real-world model boundaries imposed by domains of rivers, coastal areas, and the hinterland that have to be meshed, typically, have a very irregular shape, which leads to complications in deciding for a suitable partition. An example for such a complex study area boundary is the Elbe river. An overview map showing the complete partition of this study area using both algorithms is included in Appendix B on page 141.

**Practical Considerations for the Meshing Process**

A raster DEM is required for the derivation of breaklines. Instead of using one single raster for the whole model domain, smaller rasters are created for each tile of the partition. Each raster is then bounded by a bounding box of the respective tile. In contrast to the tiles, DEM rasters are always rectangular grids. For this reason, the rasters are, typically, going to extend over the tile boundary polygons. It is allowed, and required for the breakline detection process, that adjacent rasters have an additional overlap. This means the each raster is a little larger than a tile's bounding box. In the overlapping parts, the detected breaklines should be the same.

The parallel efficiency of the meshing process depends on the partitioning strategy, the tile shape and size (or number of tiles), and the amount of redundancy due to raster overlap over the boundary. A balanced partition helps to distribute the computational work equally among a number of computing resources. It is easier to distribute the work equally among these resources if the number of work units is much larger than the number of resources. However, it is not an option to create a large number of very small tiles — to compensate for an imbalance of the partition — because the overlap between tiles would predominate the total relevant tile area.

The optimal tile size highly depends on the computer architecture (e. g. available memory, number of computing resources, and caching strategies) and is best discovered by test runs. For good performance, it is required that the complete data for a tile fits into the computing resource's main memory and enough memory remains for the

breakline detection algorithms. These algorithms produce intermediate rasters of the same size as the original raster. Generally, three intermediate double precision rasters (smoothed terrain, slope in X and Y direction) and one boolean raster to store the breakpoints are created during breakline detection.

For a rectangular model with an area of 100 $km^2$ and a raster resolution of 1 $m$ (10000 by 10000 cells) the memory requirement for a complete Gaja3Dpar data set is about 3.1 GB. This is roughly at the limit of what a typical current desktop computer with a 64 bit architecture can handle. A limit to the raster size, where algorithms perform efficiently on a single computing resource, has been found to be between around 1000 by 1000 and 2000 by 2000 cells, so a partitioning of the model domain into mostly rectangular tiles of this size results in a good overall performance of the parallel program.

#### Algorithm 1: Gridded Boundaries

The first algorithm simply splits the model domain by overlaying a regular rectangular grid. An advantage of the first algorithm is that there is no overlap between the bounding boxes of adjacent tiles. The main difficulty, on the other hand, lies in the intersection process with the regular grid. Cutting the study area polygon with a straight line of the regular grid may cut off small "ears", which easily leads to an unbalanced partition. If the polygon is concave, it may even fall apart into more than two connected parts when split by a single grid line. Each tile polygon must be a simply connected region, however. A solution to this problem is to return a tile for each connected component. A detailed view of part of the Elbe river study area partition can be seen in Figure 6.5. One can see from this figure that, first, the variation of tile area is large (standard deviation of 0.38 $km^2$, and second, a major part of the bounding box of many tiles extends over the actual tile polygon.

#### Algorithm 2: Minimum Bounding Rectangles

The second algorithm tries to improve over the first with regards to a more balanced partition. It is based on a recursive subdivision of the *minimum bounding rectangle* of the study area polygon. The minimum bounding rectangle of a polygon is a rotated rectangle enclosing the polygon with minimized area. The algorithm works as follows:

1. Start with the study area boundary polygon.

2. Determine the current polygon's minimum bounding rectangle. If the minimum bounding box area is smaller than a given threshold, stop splitting this polygon.
3. Split the current polygon's minimum bounding rectangle into two half rectangles across the longer side. These two rectangles are then used to clip the current boundary polygon into tiles.
4. In case the current boundary polygon falls apart into more than two simply connected regions during step 3, the largest region in each half is considered as the tile. All smaller regions are joined back to the tile in the other half.
5. For both of the resulting tile polygons of step 4, the algorithm continues at step 2.

Applying this algorithm to the Elbe river study area boundary polygon results in the partition seen in Figure 6.6. Also shown are the respective minimum bounding rectangles for each tile. These rectangles have a much smaller extension over their tile polygons than the bounding boxes in algorithm 1. Only a few exceptions occur where the study area has elongated "ears". In addition, they are mostly of similar size. However, in regions with a complex geometry, such as the port channels in the upper middle of the picture, adjacent minimum bounding rectangles overlap to some extent.

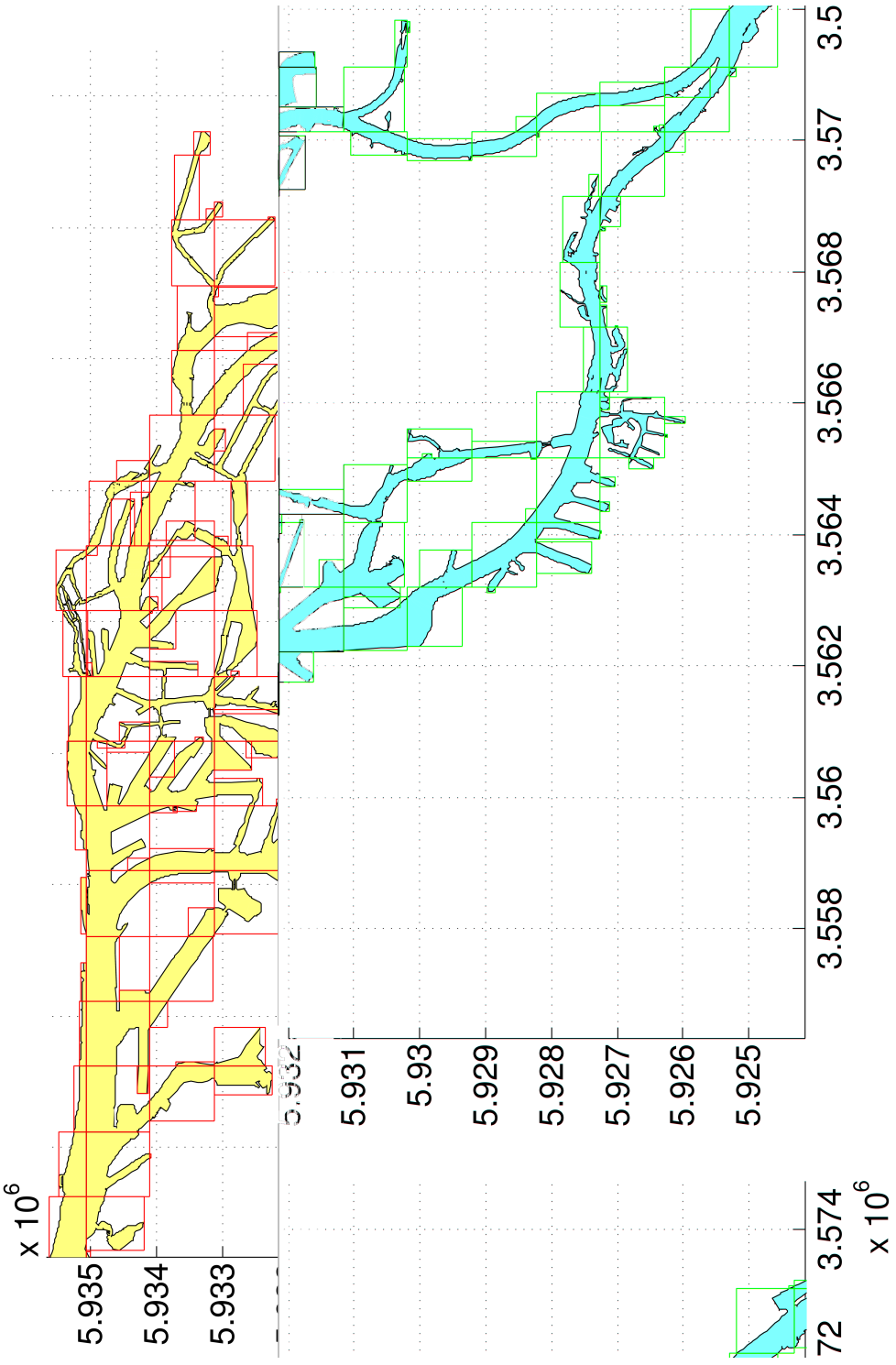



**Figure 6.5:** Algorithm 1: Simple, gridded boundaries for the Elbe river water body around the island of Wilhelmsburg, Hamburg (grid size 1 by 1 km). The bounding boxes induce a 54% overhead compared to the real boundary area. The mean bounding box area is 0.4 $km^2$, the standard deviation is 0.38 $km^2$.

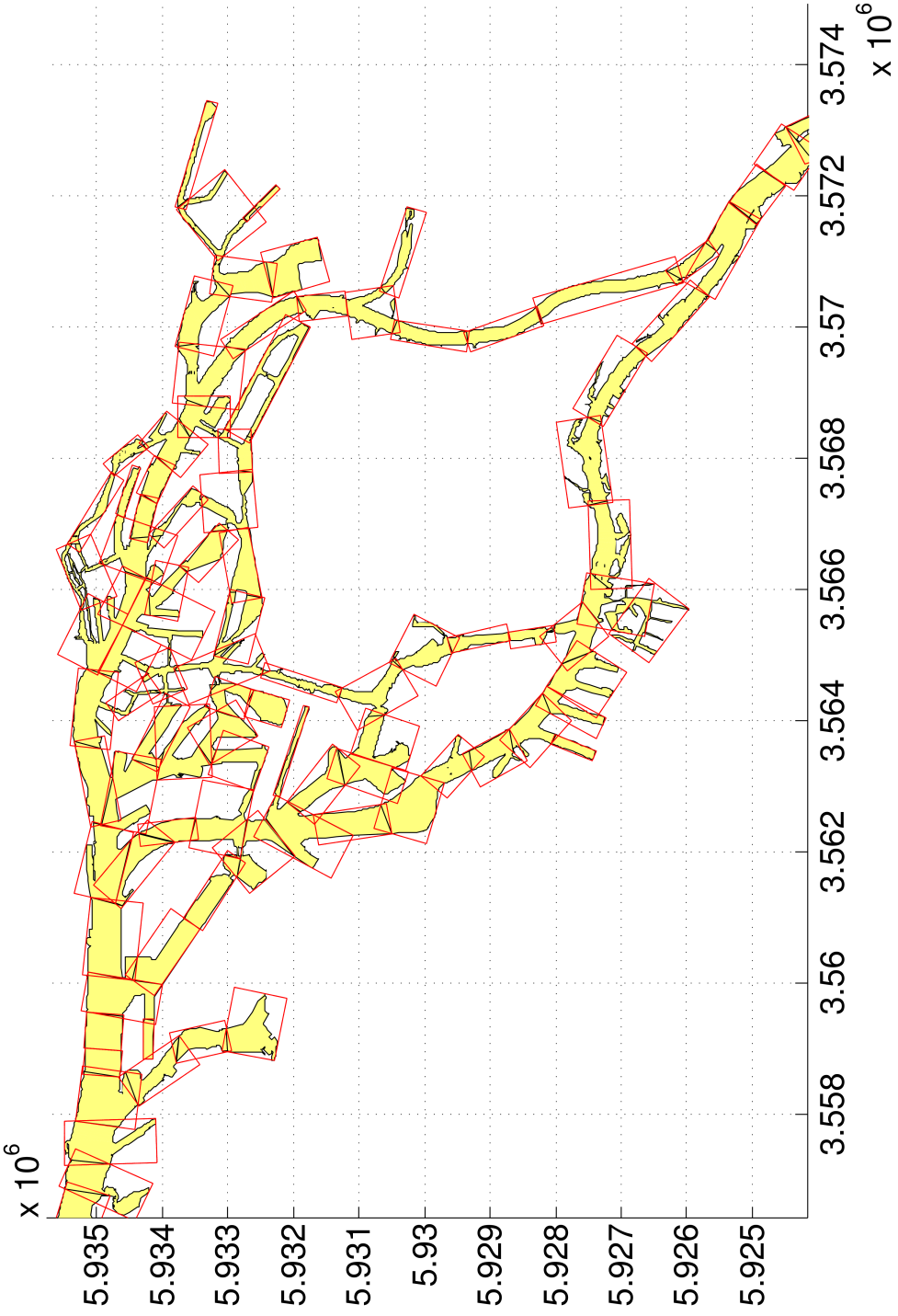



**Figure 6.6.:** Algorithm 2: Improved partition of the Elbe river water body boundary by minimum, rotated bounding boxes (1 $km^2$ maximum bounding box area). This partition only has a 43% overlap over the real boundary. The mean bounding box area amounts to 0.6 $km^2$ with a standard deviation of 0.20 $km^2$.

### Evaluation of the Two Algorithms

The two algorithms have been compared with regards to the total number of tiles generated for a similar targeted tile size, the total overlap of bounding boxes and their extension over the study area boundary, and the mean and standard deviation of the tile size. Even though the first partition has a larger number of tiles, a higher overlap is incurred compared to the second partition. This disadvantage arises because the minimum bounding boxes in the second partition were allowed to be rotated. The second partition is, in addition, much more balanced, measured by the standard deviation of 0.20 $km^2$ compared to 0.38 $km^2$ in the first algorithm. This advantage is mainly owed to the divide-and-conquer type of the second algorithm.

Nevertheless, even the second algorithm is not optimal in the sense that it cannot produce the optimal partition seen in the following simple, nevertheless hard, example (Figure 6.7). The problem of study area partitioning is similar to rectangular polygon cover problems in computational geometry (see e. g. [STA95]). O'Rourke and Supowit [OS83] showed that many polygon cover problems are NP-hard if the polygon is allowed to contain holes or if rectangular covers are sought. To the author's knowledge, polygon covers using overlapping, rotated rectangles have not been investigated so far.

In case neither of the two proposed algorithms give a satisfactory result, or if an existing partition, e. g. one of tiled DEM data, shall be used, it is possible to provide the meshing library with a manual partition of the model domain as an ordered list of tile polygons.

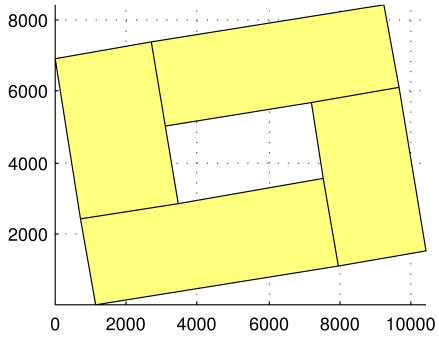



**Figure 6.7.:** Partition of a polygon, where both presented algorithms fail to derive the optimal solution. In this case, the tile polygons are identical to their respective minimum bounding boxes.

### 6.2.3. `CreateRaster`: Splitting the Input Data

Before the breakline detection process can be started, the DEM data set for the study area has to be split into smaller sets for each tile obtained from a study area partition using one of the algorithms in the previous section. The partition will be used to split a large DEM data set of unstructured points (point cloud) into smaller point clouds and convert these to convenient raster DEMs for breakline detection. The Gaja3Dpar library provides the geoprocessing operation `CreateRaster` for this purpose.

#### Rasterization of Unstructured Point Data

A service for rasterization of a point cloud, which is described here, can be reused as a component in other applications that require such a transformation, such as the preparation of DEM data obtained from remote sensing for use in a GIS. Gaja3Dpar implements this rasterization step as 2D interpolation of the terrain data based on two complementary steps: First, an algorithm for $k$-nearest-neighbors and bounding box search described in [AG11], and second, a scattered data or triangulation-based interpolation. The algorithm in the first step returns the $k$ nearest points to an input location in the point cloud within a given search radius or all points in a tile. [AG11] have implemented an efficient, spatially indexed data structure for the mentioned spatial queries. Nearest points can be input directly to scattered data interpolation of raster cells in each tile, e. g. using the inverse-distance-weighted (IDW) [She68] or kriging [Kri51] interpolation methods. For more advanced methods, such as natural neighbor [Sib81], bilinear or bicubic [Key81] interpolation, a triangulation of those tile points has to be built in the second step.

Gaja3Dpar will have to load the point cloud into memory before it can build the spatial index required for data splitting. If the total number of points in the point cloud is very large, i. e. more than a few million points, it is not feasible to build this index. The data set has to be split first. For interpolation, there need to be sufficient sampling points inside the tile boundary and in a small distance around it. This distance depends on the interpolation method and point spacing, i. e. the resolution of the point cloud. It needs to be specified as a parameter. A user-specified boundary buffer distance ensures that the tile data extends across the boundary.

In order to reduce in-memory data, the straightforward solution to data splitting is to use an external database with a suitable spatial index and spatial query possibilities, e. g. PostGIS, an extension of the free PostgreSQL database. This database could then be queried for points inside a tile boundary. An approach to data splitting that fits better to the conception of a spatial data infrastructure is to use a geodata service for

the point cloud, more precisely, the WFS or WCS. Both services are applicable because unstructured points are, at the same time, geographic features and coverages.

The WFS supports immediate spatial filtering of the data upon request. Unfortunately the existing WFS implementations that were tested (deegree2[1] and Esri ArcGIS Server) deliver points only as GML features and cannot deal with the large amount of data, because of its structured, textual representation in XML. Besides, the overhead of representing point data as individual GML features is extremely large. As a counter-example, the commercial LiDAR Server by QCoherent Software[2] has promising capabilities for handling very large point clouds, but it only supports the WMS specification. Its data delivery request format does not follow any OGC standards. The result format is the efficient, binary ASPRS LiDAR Data Exchange Format Standard (LAS). Unfortunately, such a proprietary interface is unsuitable for use in a spatial data infrastructure.

As the second option, in a manner similar to the WFS, a WCS can filter data when queried. Although the WCS (version 2.0), adopted in August 2010, specifies point cloud coverages, in addition to raster coverages, it seems that no implementation supports this feature yet. WCS would have the advantage that it can deliver point cloud coverages not only as GML features, but in an arbitrary external encoding, e. g. comma-separated values (CSV), NetCDF, LAS, or Esri Shapefiles[3].

Tests in the GDI-Grid project using a point cloud with 4.5 million spot measurements of the Elbe river bathymetry have shown that a PostGIS database is suited for querying with a polygonal region. The point cloud was imported into the database and then queried using `psql`, the PostgreSQL administrative command-line tool. Delivering all the points in the table took roughly two minutes, which was an acceptable response time. If the Elbe river domain was split into 16 tiles, the total time until termination depended, largely, on the number of points in each tile: queries took at least 1 second per 10000 points.

#### Starting With Raster Data

Digital elevation models for large areas can be organized in an efficient way in a GIS. Elevation data in a GIS is usually split into smaller tiles, which are either kept in separate files or organized in a database. The tiling model in the GIS, however, is most likely different from the partition of the domain boundary and there is no

[1] `http://deegree.org`, deegree3 postulates streaming capabilities, which might solve the problem

[2] `http://lidarserver.com`

[3] Shapefiles come with separate files for the payload data, feature index, and spatial index, so they cannot be used as an encoding without bundling them into a single file, e. g. by zipping them.

overlap between tiles. Under these conditions, for parallel breakline detection, a manual re-tiling of the data cannot be circumvented.

The preferred alternative over splitting and converting the terrain data set directly, however, is to employ a WCS serving this data. It is anticipated that such a service will soon be available in spatial data infrastructures, e. g. according to the European INSPIRE directive [Euro7a], providing DEM coverage for whole countries. This service can dynamically provide a raster for each tile when queried using the tile polygon's bounding box (ad-hoc tiling). Unfortunately, current WCS implementations do not seem to support rotated bounding boxes, as produced by the second algorithm, for specifying the queried spatial domain. The WCS specification actually allows that any `gml:RectifiedGrid` may be used for the query. The XML schema defines `gml:RectifiedGrid` as "a grid for which there is an affine transformation between the grid coordinates and the coordinates of an external coordinate reference system. It is defined by specifying the position (in some geometric space) of the grid "origin" and of the vectors that specify the post locations"[1].

### 6.2.4. `DetectBreaklines`: Parallel Breakline Detection

The result of the preceding partitioning and data splitting steps has been a subdivision of the study area into tile boundary polygons and DEM data into raster tiles. In this step, a tile polygon determines the area in which the breaklines generated for a raster tile will be regarded as valid or relevant. In order to make sure that the results for neighboring rasters match, special attention needs to be paid to the tile boundaries. An artificial overlap between the generated raster tiles ensures that the `DetectBreakline` operation derives matching breaklines on the interfaces of adjacent tiles.

As described in Chapter 4, image processing algorithms are used in the breakline detection process, such as raster smoothing, DEM slope calculation, edge detection, and classification of raster cells (image pixels) as *breakpoints*. Breakpoints are raster cells that, when connected, form a breakline. Most of these image processing algorithms use a filtering method to calculate a cell value based on the surrounding cells inside a window of a certain size (image convolution). The width of the filter window determines the required overlap so that a filter gives the same result for cells on the interface of adjacent tiles, i. e. the set of breakpoints matches (see Figure 6.8). A Gaussian low-pass filter (for noise reduction or smoothing) in general requires more overlap than a simple edge filter (for slope derivation) due to its bigger window size.

Breaklines are first vectorized, i. e. converted from a raster representation of breakpoint sets to linear features, and then clipped at the boundary. Breakline sections outside

[1] `http://schemas.opengis.net/gml/3.2.1/grids.xsd`

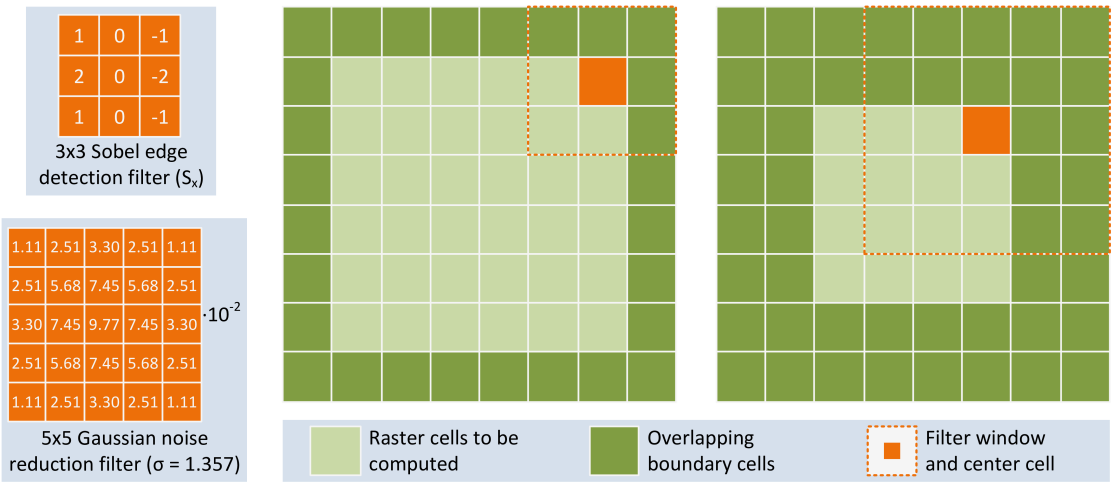



**Figure 6.8.:** Required overlap on the tile boundaries in the breakline detection process depending on the size of the filter window.

the tile boundaries are not used for further processing. When connecting breakpoints to a breakline, there needs to be an additional cell width of overlap to account for the correct derivation of breakline directionality. The first cell outside the boundary determines whether a breakline will extend perpendicularly or diagonally from the last cell inside the boundary. By the application of this rule, breaklines at the common boundary of adjacent tiles will have identical endpoints. It allows to match pairs of detected breaklines and merge them, using a small tolerance, to a single, coherent line (compare Figure 6.9).

A problem is imposed by feature detectors based on image processing algorithms with two thresholds, low and high, that classify a breakpoint if its gradient is higher than the low threshold (weak condition) and if it is part of an edge with at least one breakpoint meeting the high threshold criterion (strong condition). This may lead to the classification of a breakpoint in one tile, whereas the detector fails to identify the same breakpoint in an adjacent tile because it is not locally part of a strong edge. A workaround to the problem is not to use the weak condition, but only to classify based on the strong condition, i. e. to set both the low and high threshold to the same value, e. g. the lower threshold.

### 6.2.5. `CreateTin`: The Final Mesh

In the final triangulation step, realized by the `CreateTin` operation, the original study area polygon and the breaklines of all tiles are set as constrained edges of the mesh.

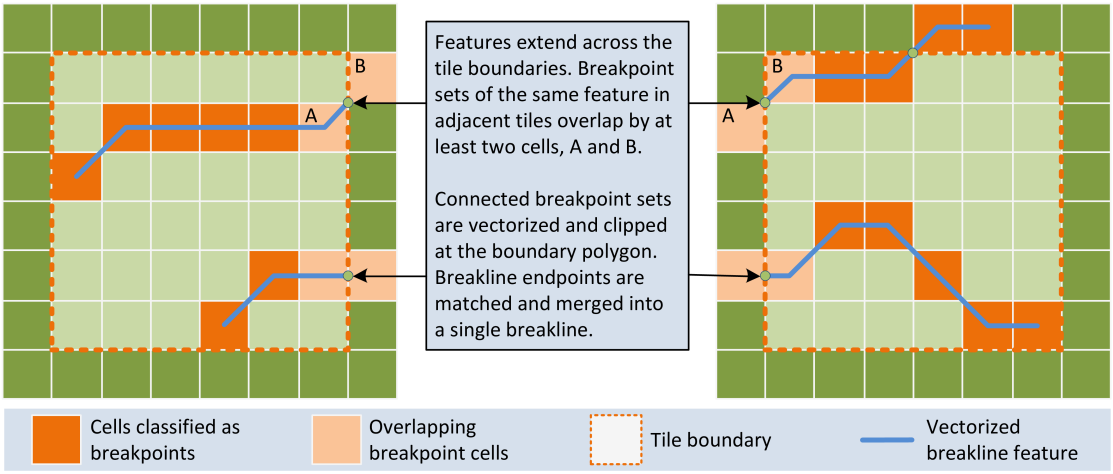



**Figure 6.9.:** Coherent breaklines can be detected across tile boundaries.

This operation is aware that breaklines need to be merged and snaps the endpoints of lines that are closer together than a given distance. Triangulating the breaklines and the model boundary is not very challenging because the amount of data has been drastically reduced in the breakline detection step.

Assigning the elevations to the nodes of the triangulated mesh is also done in this step. Again, the study area tile polygons and DEM tiles are required. Sequentially, for each tile, the elevation information for nodes in this part of the mesh are interpolated from the respective raster tile. This could be done in parallel, too, if another split-and-merge type of parallel construct were used. However, the prototype is not able to partition the mesh nodes for this purpose.

## 6.3. Implementation of the Methodology in the Grid-WPS Framework

After having designed a methodology for parallel mesh generation, the next challenge is to interface the operations of the meshing library, Gaja3Dpar, with a geoprocessing grid service so that the operations can utilize the resources in a computational grid. The gridification requires a WPS process specification of all geoprocessing operations (Subsection 6.3.1), an executable file for the grid (Subsection 6.3.2), and an implementation of the process specification using the Grid-WPS framework from Chapter 5 (Subsection 6.3.3) that submits the executable to the grid.

### 6.3.1. WPS Profile

The following tables list the input and outputs of the three Gaja3Dpar WPS processes (`CreateRaster`, `DetectBreaklines`, and `CreateTin`). For each item, the direction (I/O, in or out), the unique identifier, form (*C* for complex or *L* for literal), data type, and multiplicity (minimum and maximum number of occurrence) is shown. Since the WPS specification does not say exactly how data types should be declared, this thesis will take on the convention of using XML simple types for literal data and GML types for complex data. In this way, the Gaja3Dpar WPS may easily be chained with data services in a spatial data infrastructure.

For purposes of this prototype, it is intended to primarily use efficient binary representations of vector data, in particular, the Esri Shapefile format, which applies to point, line, and polygon data all the same. However, there is no generic way of using a Shapefile where GML is required, so an application-specific content-type and encoding has to be defined in the Gaja3Dpar process description. Shapefiles consist of at least three separate files, which are bundled into a single ZIP file input. As there is no standard notion of how to define a Shapefile data type, this procedure is not recommended. In fact, it is only required for massive point data, which cannot, currently, be expressed as `gml:MultiPoint` in an efficient way. As of GML version 3.2.1 there exists a `gml:SimpleMultiPoint` data type for representation of a large number of points, but it does not pose a valid substitute for `gml:MultiPoint`. If this deficiency is solved in a future version of GML, it is recommended to use `gml:SimpleMultiPoint` by default.

Raster data can safely be represented as `gml:RectifiedGridCoverage` by default. Even though this is a textual XML format, it is common to use an external file reference to the actual payload data, i. e. the raster data can exist in a file separate from the XML fragment, which then only serves as a container for spatial reference information. The content-type of such a WPS input must either contain several parts or the file reference given inside the XML fragment must be remotely accessible, e. g. relative to the location of the XML document, if available, or an absolute URI with a well-known protocol.

### 6.3.2. Gaja3Dpar Grid Executable

To the grid resource, selected functionality of the Gaja3Dpar library must be made available as a batch executable for grid jobs. The required procedure is not straightforward for an interactive application based on the proprietary software MATLAB by the MathWorks. Two possibilities for gridification have been investigated, each having different licensing requirements: the MATLAB Distributed Computing Server and the MATLAB Compiler.

**Table 6.1.:** Gaja3Dpar `CreateRaster` process description.

| I/O | Identifier | Form | Data type | Multiplicity |
|---|---|---|---|---|
| In | `Boundary` | C | `gml:Polygon` | 1..* |
| In | `DemPoints` | C | `gml:MultiPoint` | 1..* |
| In | `GridX`, `GridY` | L | `xs:double` | 1 |
| Out | `DemGrid` | C | `gml:RectifiedGridCoverage` | 1..* |

**Table 6.2.:** Gaja3Dpar `DetectBreaklines` process description.

| I/O | Identifier | Form | Data type | Multiplicity |
|---|---|---|---|---|
| In | `Boundary` | C | `gml:Polygon` | 1..* |
| In | `DemGrid` | C | `gml:RectifiedGridCoverage` | 1..* |
| In | `EdgeFilter`, `SmoothFilter`, `FeatureDetector`[a] | L | `xs:string` | 0..1 |
| In | `smooth`, `highThresh`, `lowThresh`, `distanceTolerance` | L | `xs:double` | 0..1 |
| Out | `Breaklines` | C | `gml:MultiCurve` | 1..* |

[a]Default values will be assumed for these parameters.

**Table 6.3.:** Gaja3Dpar `CreateTin` process description.

| I/O | Identifier | Form | Data type | Multiplicity |
|---|---|---|---|---|
| In | `Boundary` | C | `gml:Polygon` | 1..* |
| In | `Breaklines` | C | `gml:MultiCurve` | 1..* |
| In | `MaxArea`, `MinAngle`[a] | L | `xs:double` | 0..1 |
| In | `DemGrid`[b] | C | `gml:RectifiedGridCoverage` | 0..* |
| Out | `ModelTin` | C | `gml:TriangulatedSurface` | 1 |

[a]Optional triangulation parameters.
[b]If omitted, no elevations will be assigned to the nodes of the final mesh.

The first — and most obvious — solution is to install a special MATLAB product for distributed computing on all computing resources in the grid, the MATLAB Distributed Computing Server (MDCS) [1]. It adds parallel programming directives to the scripting language and schedules independent tasks in close integration with an existing batch system. This product needs to be pre-installed, configured, and licensed for a specified maximum number of computing nodes in the grid. The current price for using MDCS with 256 computing nodes in a non-commercial grid for academic use is EUR 37,500[2], which prohibits its use in this thesis. Additionally, the installation is only possible in tightly-coupled cluster environments with Message Passing Interface (MPI) (see Chapter 2, Subsection 2.1.3) and most functions require a homogeneous computer architecture. For these reasons, MDCS is not regarded as a suitable choice for application in a grid infrastructure.

The second procedure (MATLAB Compiler) has been selected for the prototype. The advantage of this solution is that several instances of the application can be run in parallel on an arbitrary number of computing nodes without any MATLAB licensing issues and runtime costs. Only the creator of the Gaja3Dpar executable needs to own a MATLAB Compiler license. This product converts a MATLAB function and all its dependencies to a platform-dependent command line executable file. In the following is explained how the procedure can be applied to gridify the Gaja3dpar library.

The three service operations of the Gaja3dpar library — described in the previous section — have been bundled in a single MATLAB function that can be called with variable parameter-value pair arguments, `Gaja3dService`. This service function is realized as a Gaja3Dpar script. The arguments to the `Gaja3dService` function advise a certain sequence of internal Gaja3Dpar API operations, i. e. specification of the tiles to process, setting the tile input data, performing selected steps of the discretization process, and saving the results. For details of this API see Appendix A. The specification of tiles and tile inputs allows for a flexible distribution of work to different computing resources, while keeping the process description the same for all resources. This corresponds to a single process multiple data (SPMD) technique of achieving parallelism. Whenever the library changes, e. g. new interpolation or breakline detection methods have been developed, the internal discretization process has to be adapted by a software developer to make the new API functionality available in `Gaja3dService`.

Using the MATLAB Compiler, the `Gaja3dService` function is compiled as a stand-alone executable. This executable can then be run together with an installed MATLAB Compiler Runtime (MCR), a set of dynamic MATLAB libraries and runtime engine for

[1] `http://www.mathworks.de/products/distriben`

[2] MathWorks Products and Prices for the MATLAB Product Family, Euro Academic, March 2012: "Academic pricing is reserved for noncommercial use by degree-granting institutions in support of on-campus classroom instruction and academic research."

compiled executables. Similarly to the MDCS, the MCR needs to be installed in the grid before a grid job is to be submitted. This installation can run in an unattended way, automatically, for example inside the user space of the client that submitted a grid job, prior to running the application. For a computing cluster with a shared file system among the computing nodes there is only one installation of the MCR necessary. However, the installation takes time and space, which can be saved if the required software is pre-installed on all grid resources. The compiled `Gaja3dService` executable is then started by a regular Unix shell script, which finds the installation path of the MCR in an environment variable (`MCRROOT`).

### 6.3.3. Flow Model Discretization Grid-WPS

The flow model discretization service is a Grid-WPS implementation of Gaja3dpar. In particular, the implementation is based on WPS 0.4 and the WSRF, namely Kalypso and Globus Toolkit 4. It makes use of the WS-ResourceProperties, WS-ResourceLifetime, WS-BaseFault and WS-BaseNotification specifications in WSRF and implements the WPS profile described in Subsection 6.3.1. The general procedure adheres to the gridification methodology from Chapter 5. It does not use the automatic Grid-WPS generator developed by Dorka [Doro9] (see Subsection 5.2.1) because the prototype was created prior to the completion of this thesis. From the current point of view, it would be advantageous to use Dorka's modules.

First, the WSDL interface of the flow model discretization service needs to be described. The WSDL interface includes, in addition to the common WPS operations, a specific `Execute_ProcessID` operation[1] for each Gaja3Dpar operation. The grid service further defines the properties of a WS-Resource, `Gaja3DResource`, one property for each WPS input or output parameter. The `Execute_ProcessID` operations mark all inputs as optional and take values of matching resource properties as default. Upon successful termination of a process, the result is used to update the current value of the corresponding resource property. The rationale behind this is to be able to provide process inputs by setting the appropriate `Gaja3DResource` resource property, e. g. the domain boundaries, which are common to all processes. Other goals are to make intermediate results of a process execution available, enable notification of results via WS-Notification to interested clients, and maintain provenance information, e. g. parameters used in the process.

An additional resource property, `GramEndpointReference` contains a reference to the Globus Resource Allocation Manager (GRAM) service. This property is initialized when a `Gaja3DResource` is created and is further used for all grid job submissions.

[1] `Execute_CreateGrid`, `Execute_DetectBreaklines`, `Execute_CreateTin`

Implementing the WS-ResourceLifetime specification allows for user-initiated or scheduled destruction of a `Gaja3DResource`. Upon resource destruction, any pending or running grid jobs can be stopped and all results created by the service can be cleaned up. This property is also used to determine the location of the GridFTP file system root that is common to the computing nodes. Computing nodes without a shared file system are not supported by the prototype.

Grid job submission relies on the `Gaja3dService` executable described previously, its conformance to the architecture of the computing nodes managed by the GRAM service, and a proper set-up of the MCR. In this prototype, the executable file is transmitted to the computing nodes, alongside the required input data. If it was possible to determine the computer architecture of a computing node before job submission, a suitable executable file could now be selected to achieve platform-independence.

Before a grid job is submitted, the `Gaja3dResource` creates a sandbox directory in the client's home directory on the GridFTP file system. Any valid grid client is mapped to a Unix user in this file system automatically by his grid certificate. The sandbox is the target directory to store the `Gaja3dService` executable and all input data, and it serves as a working directory for execution. The input data is taken from the WPS process inputs. In-line complex input parameters are saved to a file in the sandbox. Referenced complex inputs are copied to the sandbox. The GridFTP protocol is used to allow third-party file transfers, where possible. The command line arguments of the `Gaja3dService` executable are assembled as parameter-value pairs using sandbox-relative filenames of complex inputs and all simple literal WPS input parameters.

The geoprocessing operations provided by the flow model discretization service are suited to orchestration in a grid workflow. This aspect will be highlighted in the following section.

## 6.4. Flow Model Discretization Grid Workflow

This section shows how control and data flow can be managed in a grid-aware workflow system using a standard BPEL description. The flow model discretization workflow is prototypically implemented in BPEL 1.1 and can be executed in the BPEL4Grid Engine.

### 6.4.1. Data Flow

The data flow diagram (Figure 6.10) shows that external data can come from either a Web Feature Service or a Web Coverage Service. As a WCS can, theoretically, deliver

both elevation point and raster coverages, its data are fed either into the raster creation activity or directly into the breakline detection process. One can see that the raster is, again, required in the tin creation step, where it is used for interpolation of elevations to the mesh nodes. The domain boundary and partitioning step is excluded from this diagram for better readability. From this point of view, the tile boundary polygons are an external user input and required in each process.

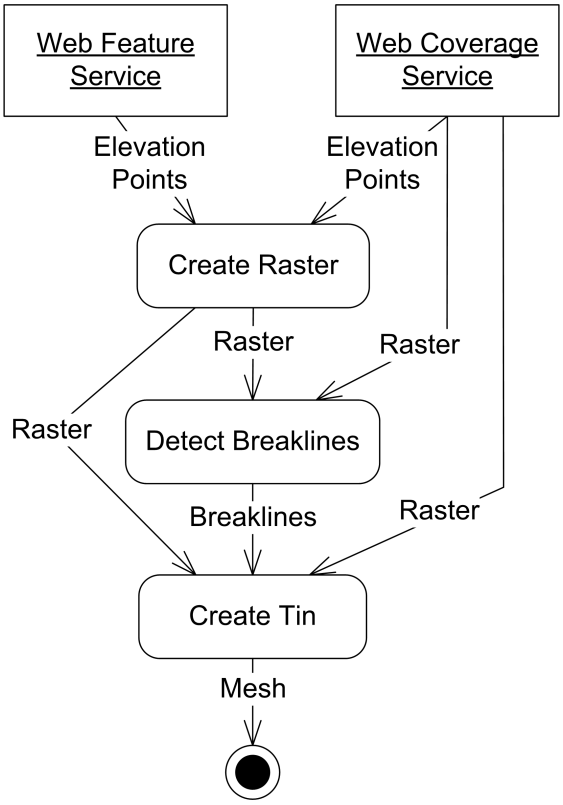



**Figure 6.10.:** Flow model discretization data flow. The arrows represent possible alternative data flows. Different courses of flow can be seen in the control flow diagram in Figure 6.11.

### 6.4.2. Parallel Control Flow

The Gaja3Dpar Grid-WPS monitors the submitted grid job, updates its WPS status document, and waits for completion. When done, the GridFTP resource locations of the process results are stored in the respective `Gaja3DResource` properties and in the status document. The service then reports either success or failure. The WS-Resource

is not destroyed at this point, so that it is possible to send subsequent requests to the same instance of Gaja3Dpar, e. g. with the goal of chaining the raster creation and breakline detection steps for a tile. It is foreseen that a `Gaja3DResource` does not only manage results by linking to them in a resource property, but it could also store them as OGSA resources. Dorka [Doro9] has investigated this possibility. However, the question remains how the actual payload data could be stored more efficiently, e. g. in a transactional WFS or WCS backed by a database.

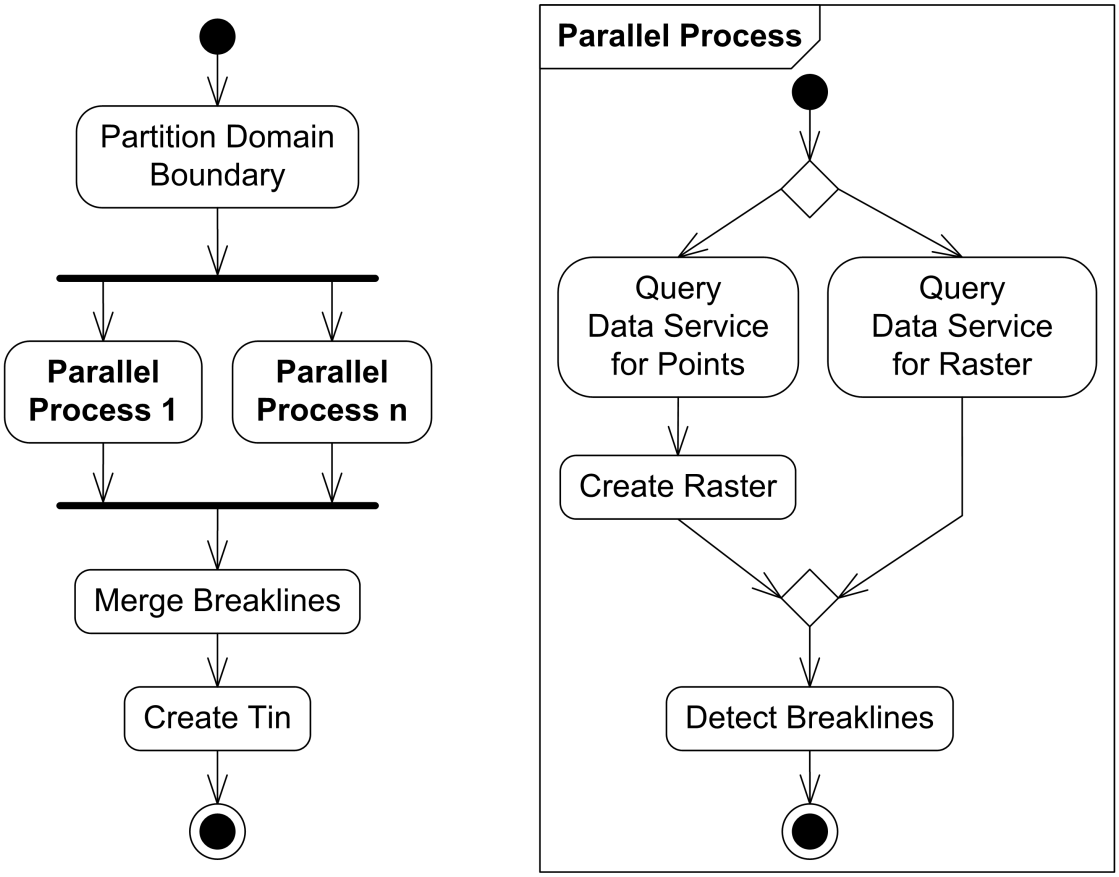



**Figure 6.11.:** Flow model discretization control flow with parallel region.

It was shown above how a partitioning of the domain boundary leads to tiles for which breaklines can be detected independently. The control flow diagram (Figure 6.11) includes the partitioning step as the first activity. The decision to start with either point or raster data is actually made at the beginning of the workflow and is thus constant for all tiles. For each tile a separate breakline detection process is initiated and executed in a parallel region. It consists of a data acquisition step and either one or two Gaja3Dpar

operations (`Create Raster`, `Detect Breaklines`). The breakline results of all tiles are joined at the end of the parallel region and merged prior to the final Gaja3Dpar mesh creation step, `Create Tin`.

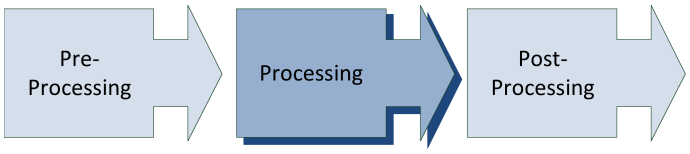
## Conclusion

The parallel process for mesh generation and the prototypical implementation of the flow model discretization service described in this chapter demonstrated that massive amounts of digital terrain data can efficiently be processed in the domain of hydrodynamic modeling using grid technology. For the duration of the GDI-Grid project that partially funded this thesis, the service and a flow model discretization workflow had successfully been deployed in the German D-Grid infrastructure[1].

Furthermore, the presented abstractions for service development allow for modularity, reuse, and interoperability of geoprocessing services in a spatial data infrastructure. By conforming to geoprocessing and grid standards — according to the described Grid-WPS framework — the geoprocessing functions and the grid workflow can be accessed by both WPS and grid clients with a valid grid certificate. In this way, the meshing software for flow model discretization has been made available as a distributed geoprocessing service to all users of a virtual organization for geoprocessing.

Yet, the service is not restricted to an application in the domain of hydrodynamic modeling. The operations may also be used separately or as part of other scientific workflows. Meshing a bounded domain is an operation that needs to be performed when creating unstructured discretizations for numerical models in other disciplines, as well, such as aerodynamics or structural mechanics. As another example, the rasterization of unstructured point clouds is a typical operation required in remote sensing applications. Finally, breakline detection together with mesh creation may also aid in data reduction and improved three-dimensional visualization of large terrain data sets.

[1] After the project ended, the respective infrastructure was suspended due to lack of funding.




# 7. Flood Simulation Service

The flood simulation service developed in this chapter shows how flood scenarios can be explored by two-dimensional numerical simulation of the flooding process using an existing flow model discretization. This is done by developing a standard-compliant Grid-WPS for flood simulation, following the gridification method demonstrated in Chapter 5, and by extension of a numerical flow model for parallel execution in the grid.

The Grid-WPS makes it possible to integrate the flood simulation service into a spatial data infrastructure in the form of a geoprocessing service. In this way, the service is enabled to request input for the hydrodynamic simulation dynamically from sensor observation services delivering flood hydrograph data. The Grid-WPS further executes and manages the hydrodynamic simulation in a way that utilizes the power of the provided distributed high-performance computing environment. The flood simulation service follows up on the flow model discretization service (Chapter 6) in that the created flow model can now be set up for the simulation and evaluation of flood scenarios in a computational grid. This adds support for the processing phase of hydrodynamic modeling. This has been evaluated through test runs on grid resources of the German D-Grid infrastructure.

The structure of this chapter is as follows: Section 7.1 briefly presents the motivations for developing a flood simulation service and lists the objectives and requirements of the prototype. In Section 7.2 an overview of the current state of research regarding flood simulation in a grid computing environment is given. The architectural design of the Grid-WPS prototype, its interface, as well as the parallelization of a hydrodynamic numerical model based on the Kalypso Simulation Platform is described in Section 7.3.

## 7.1. Introduction

In times of climate change, increasing flood risk, and the enactment of flood management policies and plans, detailed analyses of the dynamics of flooding and the creation of flood maps have become indispensable. There is a growing demand for the

reliable assessment of forthcoming flood events and thus the need for two-dimensional simulation of river flow and floodplain inundation, urban flooding, and storm surges capturing the course of a flood event with a high spatial and temporal resolution. Such simulation results are used in flood mapping and flood risk management, with the purpose of obtaining a basis for informing stakeholders and the population prior to new flood disasters about the current flood risk as well as possible future impacts, e. g. due to climate change. In this way, flood simulation can help to improve emergency plans and reduce the consequences of flooding [ZA+10].

### 7.1.1. Motivation and Challenges

The motivations for developing an approach to flood simulation based on geoprocessing grid services are manifold. First, such a service is attractive for users foreign to hydrodynamic modeling who regularly have to perform resembling simulations, e. g. flood forecasting for a river stretch using observed and predicted flood hydrographs. A geoprocessing workflow for flood forecasting would simplify and accelerate the flood simulation process for this user group. Second, by conforming to geodata and geoprocessing standards, a service for flood simulation can effortlessly be integrated into a Spatial Data Infrastructure (SDI). The flood inundation maps created in the flood simulation process add value to an Spatial Data Infrastructure (SDI) targeted at users in environmental engineering and flood risk management. Third, flood simulations are generally long-running and data intensive, because the underlying numerical model requires solving many large, sparse systems of nonlinear partial differential equations. A computational grid can help to speed up the calculation, e. g. for real-time flood forecasting, and to manage the simulation results.

Indeed, the multiplication of complexity and data arising in flood modeling are common to many problems in earth sciences, i. e. domains dealing with simulation models for complex or large-scale physical processes. Therefore, the motivations and challenges sketched above are not limited to the domain of flood simulation, but can analogously be applied to other domains, such as climate, atmospheric, ocean, or geophysical modeling.

### 7.1.2. Targeted User Groups

The intended flood simulation service automates the simulation of flood scenarios in the grid while hiding undesired model complexity from managers and planners. Such users are not proficient in hydrodynamic modeling and come, for instance, from the

domains of water resources management, spatial and urban planning, or emergency management. This user group will be termed *flood managers*.

Domain experts in hydrodynamic modeling, on the contrary, take up the role of *flood modelers* who create and set up hydrodynamic models that non-experts can use for flood simulation within certain plausibility limits. These users require additional flexibility to set up and calibrate the hydrodynamic model. For example, physical and conceptual model parameters need to be tested that, generally, remain fixed once the model has been calibrated. Flood modelers also fill the position of workflow composers chaining a sequence of pre-configured flood modeling steps. Such a flood simulation workflow could provide support to flood managers or other flood modelers, e. g. by automating the execution of a series of flood simulations with different parameters together with a subsequent analysis.

### 7.1.3. Objectives

The design of a flood modeling workflow was demonstrated in the previous chapter, so the prototype developed in this chapter will concentrate on the service's interface for the advocated user groups. In order that both flood managers and flood modelers can take advantage of the flood simulation service, two programming interfaces with different objectives will be developed.

One of the outcomes of the flow model discretization service was its ability to use geodata services for digital terrain data provided as part of an existing SDI. Hydrodynamic flood models, on the other hand, process a diversity of data sets that have not yet been discussed in the context of SDI. There is no common understanding of the model data required for hydrodynamic simulation. This data depends on the underlying numerical model and refers to the boundary conditions of the model (water level and discharge hydrographs), the initial conditions (flow state at the start of the simulation), and other numerical parameters, e. g. the simulated time span and model-specific options. The flood simulation service will address problems inherent to hydrodynamic simulation in order to demonstrate how numerical models that have a relation to space and time could be integrated into an SDI.

Even nowadays, where powerful computer hardware is cheap and readily available, two- (and higher) dimensional simulation of flow dynamics in large domains remains a challenging and time-consuming task. A grid provides access to parallel and distributed computing facilities that lend themselves to outsourcing the flood simulation and enhancing performance of the hydrodynamic model. It is a goal of the prototype to conduct the hydrodynamic simulation on grid resources in a parallel and distributed fashion.

## 7.2. Flood Simulation in a Computational Grid

Bringing together flood simulation and grid computing is a matter that has been looked at from several perspectives, as will be shown in this section, e. g. execution and data management, grid services and workflows, parallelization of the numerical model, collaboration, or semantic grid. In fact, hydrodynamic simulation is a classical example of high-performance computing (see Chapter 2), so it is vital to evaluate research endeavors regarding HPC in grid computing environments. Therefore, the approaches with the highest relevance for this study are related to the use of service-oriented grids for an HPC application.

### 7.2.1. Grid Services and Workflows for Flood Simulation

Few initiatives have previously explored the possibilities of integrating flood simulation services into the grid. In this context, the use of workflow technology promises an automation of complex processes involving those services. Most of the existing results have been achieved by Prof. Ladislav Hluchý[1] and his team in a series of successive research projects: ANFAS (FP5, 2000-2002), CrossGrid (FP5, 2002-2005), MEDIGRID (FP6, 2004-2006), K-Wf Grid (FP6, 2004-2007), and int.eu.grid (FP6, 2006-2008). Hluchý et al. have investigated workflows and portal-based user interfaces for flood forecasting. Their results are now explained in detail.

#### Flood Simulation in ANFAS

[HT+02; HT+03; TH04] performed flood simulations and flood damage assessment of the Vah river, Slovakia, in the ANFAS project[2]. A client-server architecture was developed that executed simulations in parallel on a computing cluster.

#### Flood Forecasting in CrossGrid

Collaboration components were added in the CrossGrid project[3] using a Virtual Organization (VO) for flood forecasting. The Virtual Organization members included users, data providers (in particular for meteorological input data), storage providers (for simulation outputs), and cycle providers (for computing resources). The only users were hydrological and meteorological experts [HT+04; HA+04]. The FloodGrid

[1] Slovak Academy of Sciences, Department of Parallel and Distributed Computing
[2] ANFAS: Data Fusion for Flood Analysis and Decision Support (`http://www.ercim.eu/anfas`)
[3] `http://www.eu-crossgrid.org`

forecasting application consisted of a "cascade" of simulation models ranging from meteorological models over hydrological models to a hydrodynamic model (compare Chapter 3, Subsection 3.1.2). A simple workflow management system based on Globus Toolkit 3 and a workflow description language were developed. A workflow consisted of interdependent activities representing parameterized grid jobs. Simulation results could be accessed either via a web-based portal or in a collaborative environment, the "Migrating Desktop"[1] fat client, a pluggable user interface for interactive grid applications [HH+05].

#### Advancements in MEDIGRID

In the MEDIGRID (Mediterranean Grid of Multi-Risk Data and Models) project new flood forecasting services were implemented as WSRF grid services using Globus Toolkit 4. Data transfers had to be done with a low-level transfer service as a replacement for GridFTP on the Windows platform. A specialized WSRF job submission service was created that allowed the execution of a pre-configured application locally, on the server, or by submission to a batch system [HH+06].

#### Semantic Grid Services in K-Wf Grid

The K-Wf Grid project added a user layer based on a knowledge management system that would "learn" from previous user experiences and thus help other users to take best advantage of the grid. A web portal was developed in GridSphere with a Grid Workflow User Interface (GWUI) and a User Assistant Agent (UAA) for sharing and communicating knowledge in the system [HH+06]. A Grid Workflow Execution Service (GWES) controlled the execution of workflows. A complete overview of all components of the workflow system can be found in [BU06]. The flood forecasting cascade is described in [HMH06], [BG+06], and [BH+07]. The hydraulic models integrated into the workflow are the one-dimensional models HEC-RAS and MIKE 11.

#### Interactive Grid Jobs in int.eu.grid

The Interactive European Grid, int.eu.grid, extended the workflow management system from K-Wf Grid (GWES) so that it can make use of an "interactive channel" from a user interface to the workflow manager. int.eu.grid was not strictly conforming to a service-oriented architecture, but was using a pure job submission system. For this reason, the GWES had to be integrated into an executable application that could be

[1] `http://desktop.psnc.pl`

submitted as an interactive grid job. The interaction allowed to control the running workflow during its execution, adapt it, and to exchange raw data with an application running in the grid through its standard input and standard output. With these preparations, the flood forecasting application from K-Wf Grid could be ported to int.eu.grid. The flood forecasting workflow was submitted as a grid job and controlled using the GWES interactive channel [SH+08]. As a second use case, an environmental application (particle-based air pollution simulation) was prototypically parallelized and enabled to use the interactive channel to control its parallelism during runtime, i. e. increase the number of simulated particles to improve precision [SH+08].

### 7.2.2. Parallel Applications and Services

Floros and Cotronis [FC04] argue that legacy parallel HPC simulation models implemented with MPI, such as in meteorology, hydrology, and hydraulics, should be exposed as OGSA grid services, which they call a "virtualization" of the application. This virtualization step would allow a diversity of clients to be developed without having to dig into the implementation details. Moreover, several of those virtualized MPI simulations could be coupled to simulate flood forecasting scenarios, similar to work done by Hluchý (see above). In [FC04], the authors identified two methods, with which the virtualization could be realized: (1) wrapping the MPI processes or (2) wrapping the data. As the first solution would require the grid service, or part of it, to be located on the process level of the MPI application, this approach was not investigated further. Instead, the second approach was implemented in Globus Toolkit 3. The grid services were designed containing a number of *provides* and *uses* quantities, which represent external input and output data items of the application processes, e. g. files, a database, or a data service. Grid service notifications may be used to inform interested clients (e. g. another application process) about changes in a quantity. In this way, *uses*/*provides* relationships could be modeled in a flexible fashion. It is the services responsibility to wrap and unwrap application data to store outputs into a quantity and deliver them as inputs to the corresponding process.

In a subsequent publication [FC06], the same authors developed the "ServOSims" framework for data-centric composition of service-oriented simulations with WSRF grid services and Globus Toolkit 4. The application was provided as a stateful WS-Resource with a `run` operation, *provides* and *uses* quantities were implemented as input and output WS-ResourceProperties, and the notification mechanism from [FC04] was replaced by WS-Notification. The notifications allowed a data-centric workflow composition of simulations via input and output file resources. If necessary, an intermediate grid service transformed the exchanged data on the way. The grid service implementation actually probed the applications's standard input and output streams for data exchange

with the application, in order to receive information from and give simple instructions to the application. This approach is very similar to the interactive channel described in [SH+08].

### 7.2.3. Real-World Hydrodynamic Models for Flood Simulation

Particular hydrodynamic models for flood simulation differ in their support of high-performance computing architectures and parallel or distributed computing. There is an overview of parallelization methods for 2D flood models in [NF+10]. The following list contains the most important flow models, which are accepted in hydraulic engineering practice and backed by an unstructured, two-dimensional discretization[1], and gives details about their degree of parallelization, e. g. usage of the MPI or OpenMP libraries (see Chapter 2, Subsection 2.1.3).

#### MIKE FLOOD

The commercial MIKE FLOOD model suite (including MIKE 21 and MIKE 3) from DHI Group simulate free-surface flow based on a cell-centered finite volume method. The flexible mesh version of the MIKE models can use a discretization of triangles and quadrilaterals. The software runs in parallel on a shared-memory system using OpenMP. A parallelization for distributed memory computers using MPI and domain decomposition has been investigated, but is currently not available to the public[2].

#### Delft3D

Developed by Deltares Systems, Delft3D provides an open-source two- or (layered) three-dimensional hydrodynamic simulation model using a finite difference method on a structured, rectilinear or curvilinear, boundary-fitted mesh. The model is parallelized using MPI and a non-overlapping domain decomposition approach[3]. Delft3D uses a direct, iterative solver for the continuity equation and an additive Schwarz method for the momentum and other transport equations (compare Chapter 3, Section 3.2). Flexible meshes and the parallel computing facilities are only available as part of their beta testing program.

[1] Some models call this a *flexible mesh*.
[2] `http://www.mikebydhi.com/upload/mikebydhi2010/publications/P030/P030_Paper.pdf`
[3] `http://delftsoftware.wldelft.nl`

### INFOWORKS

INFOWORKS 2D by Innovyze[1] is a commercial flow model which uses a finite volume method to solve the shallow water equations on an unstructured triangular mesh. This software package runs on shared-memory, multi-processor systems using OpenMP.

### TELEMAC

Originally developed by Electricité de France, this model is available as open-source and simulates 2D and 3D hydrodynamics on an unstructured, triangular mesh using a finite element or finite volume method[2]. TELEMAC can be run in parallel using the Parallel Virtual Machine (PVM) library, or, most recently, MPI.[3]

### ADCIRC

The proprietary ADCIRC coastal circulation and storm surge model solves free surface circulation and transport problems on a two- or three-dimensional unstructured grid using a finite element discretization [4]. ADCIRC can be executed on a computing cluster employing MPI. It is being developed jointly by University of North Carolina, the University of Notre Dame, University of Oklahoma, and the University of Texas.

### TUFLOW

This software is developed by BMT WBM Engineering & Environmental Consultants (Australia)[5]. There is a choice between a 1D and 2D finite differences model on a regular grid or a 2D finite volume model on a flexible mesh consisting of triangular and quadrilateral elements with an explicit solution scheme. TUFLOW is parallelized using OpenMP for multi-core machines.

[1] http://www.innovyze.com

[2] http://www.opentelemac.org

[3] Unfortunately, the open-source MPI version of TELEMAC had not been available when this study was initiated. Otherwise, it would have been a good candidate for further research.

[4] http://adcirc.org

[5] http://www.tuflow.com

#### UnTRIM

The UnTRIM model by Prof. Vincenco Casulli of University of Trento, Italy, solves the three-dimensional Reynolds-averaged Navier-Stokes equations and the transport equation for tracer concentration on an unstructured orthogonal grid [Cas99; CW00; CZ02; CZ05]. A two-dimensional, depth-integrated mode is available (UnTRIM-2). UnTRIM-2 includes sub-grid technology for a high-resolution representation of the bathymetry. The German Federal Waterways Engineering and Research Institute (Bundesanstalt für Wasserbau, BAW) uses both models, among others, in their hydraulic modeling practice. A distributed-memory parallel (MPI) version of UnTRIM, but not of UnTRIM-2, has been implemented by Jankowski [Jan07; Jan09] at BAW. This version is not publicly available.

#### Hydro_AS-2D

Hydro_AS-2D is a commercial, time-explicit finite volume model. It is developed and supported by Marinko Nujic[1]. Meshes are unstructured and may consist of triangular and quadrilateral elements. The program utilizes an OpenMP shared-memory parallelization.

#### RMA Model Suite

The suite of flow models by Resource Management Associates group (RMA), located at Lafayette, USA, consists of renowned two- or layered three-dimensional, unstructured finite element models. These include the flow model RMA-2, the salinity, temperature, and suspended sediment transport model RMA-10, the morphological model RMA-10S, as well as the water quality models RMA-4 and RMA-11. The US Army Corps of Engineers (USACE) Coastal & Hydraulics Laboratory (Vicksburg, USA) has made improvements and integrated the RMA model suite into their TABS numerical modeling system[2]. All of these models are sequential.

## 7.3. Multilevel Parallelization of a Hydrodynamic Model

Flood simulation, in general, poses problems that can be solved in parallel using a large number of distributed computing resources. Some of these problems are inherently

[1] http://www.ib-nujic.de/
[2] http://chl.erdc.usace.army.mil/tabs

parallel, e. g. parameter studies during model calibration, simulations of independent flooding scenarios, and the post-processing of simulation results by time steps or location. In the following, such inherently parallel problems will not be investigated further as their parallelization and distribution are straight-forward. At the core of flood simulation, however, is the numerical model, which describes the velocity field and water depths in the flooded areas. The numerical model is a tightly coupled application and is, hence, not inherently parallel.

The use of high-performance computing environments is nowadays indispensable due to the algorithmic complexity and scale of flood models. As shown in the previous section, a multitude of hydrodynamic models exist that are capable of performing two-dimensional flow simulation in parallel. Nevertheless, the minority of the listed hydrodynamic models are in a state that makes them readily applicable to large-scale flood simulation. Some of them have restrictions with respect to the problem sizes that can be modeled because they run on a single computing resource with shared memory only, e. g. MIKE Flood. Others can be executed in a computing cluster with MPI, e. g. the Delft3D and TELEMAC models.

After all, none of these models exhibit a loosely-coupled parallel algorithm required for a high-latency environment of distributed resources, i. e. a computational grid. In order for hydrodynamic simulation to gain performance benefits from a computational grid, different algorithms are required when the simulation is executed on a single cluster (level 1) or across distributed resources (level 2). The primary objective of this section is to show the feasibility of a multilevel parallelization with a set of two largely independent algorithms that integrate both the cluster and grid levels. First, possible algorithms for each level are sketched, then an integrative solution is presented. It will further be elucidated how this approach has been implemented. Nevertheless, the complete mathematical treatment or performance evaluation of the suggested algorithms is not a subject of this thesis.

As the author of this thesis has access to the source code of RMA·Kalypso (see below), it was chosen as the basis for implementing a grid-enabled version of a hydrodynamic model.

### 7.3.1. RMA·Kalypso

RMA·Kalypso is a derivative of RMA-10S from the RMA suite of hydrodynamic models (see Section 7.2) and is actively being developed at Hamburg University of Technology, Institute of River and Coastal Engineering. RMA·Kalypso uses an OpenMP-parallel sparse linear solver, PARDISO [SG04; SG06]). On request[1], the model is available for

[1] http://www.kalypso.wb.tu-harburg.de

free use together with the Kalypso Simulation Platform. RMA-10S has mainly been developed by Prof. Ian P. King at the University of New South Wales, Australia. The governing equations and numerical discretization in RMA-10S are described in [Kin88; King3]. Their theoretical foundation, the shallow water equations and the finite element discretization technique, have been explained in Chapter 3, Subsection 3.2.2.

The following first step in the parallelization of RMA·Kalypso enables it to utilize the resources of a distributed-memory computing cluster with MPI. This makes it effective for use in a computational grid that provides such cluster resources.

### 7.3.2. Level 1: RMA·Kalypso in a Cluster

On the cluster level, a simple, yet effective, improvement of RMA·Kalypso has been made by setting up the sparse linear system of equations in parallel and using an MPI-parallel sparse solver. This improvement was done as part of this thesis and has brought it to the state-of-the-art of parallel flow models.

An advantage of RMA·Kalypso over other numerical models has further been achieved by interfacing it with an extensible library, PETSc (see below), which acts as an adapter to a collection of distributed matrix management, preconditioning, and sparse system solver routines.

#### The PETSc Library for Parallel Computation

The Portable, Extensible Toolkit for Scientific Computation (PETSc, [BG+97; BB+11]) contains data structures for distributed sparse matrices and external interfaces to a large collection of preconditioners as well as direct or indirect sparse linear and nonlinear solvers. PETSc makes heavy use of MPI and thus requires a tightly-coupled computing environment. For good performance, PETSc needs a fast, low-latency interconnect — faster than gigabit ethernet — and high per-CPU memory performance, which is typically not available in standard multi-core machines. The reason for this is that solving sparse matrices depends more on fast memory than on fast processors [BB+11]. PETSc was chosen for its flexibility, good documentation, and easy integration into an existing software.

#### Solution Techniques for Sparse Linear Systems

In the case of a conservative weak formulation of the finite element discretization, the systems to be solved are of the general, unsymmetric form $\begin{pmatrix} A & B^T \\ B & 0 \end{pmatrix} \begin{pmatrix} \mathbf{q} \\ h \end{pmatrix} = \begin{pmatrix} f \\ 0 \end{pmatrix}$

(compare [Hei11]). Two alternative global solution methods in PETSc have been tested for this problem (see below): (1) a direct solve and (2) an additive Schwarz preconditioned, iterative domain decomposition method with direct solves on the subdomains (often referred to as a Krylov-Schwarz). MUMPS (*MU*ltifrontal *M*assively *P*arallel sparse direct *S*olver) was used for direct factorization and solution of the linear systems in both cases and a stabilized biconjugate gradient method (BiCGstab) was used for iterative solution.

MUMPS [AD+01] is a distributed multifrontal solver for general unsymmetric matrices arising from linear systems of equations. The software is based on MPI, is fully asynchronous, and has parallel LU factorization and solution phases. In order to reduce fill-in, it supports METIS or PARMETIS matrix reorderings[1], among others. PETSc provides an interface to MUMPS using its sparse, distributed matrix format (MPIAIJ).

BiCGstab($l$) (*Bi*Conjugate *G*radient *stab*ilized) is a Krylov subspace method for unsymmetric systems developed by Sleijpen et. al [SF93; Sv95]. For brevity the algorithm will not be described here. Besides, BiCGstab($l$) could easily be replaced by other Krylov methods, such as GMRES. The Krylov method requires $\mathcal{O}(l)$ matrix-vector products to find the next search direction. These matrix-vector operations lead to most of the network traffic.

Domain decomposition methods can be regarded as a family of hybrid methods between iterative and direct solvers where the problem is decomposed into subproblems on adjacent, possibly overlapping regions. Assigning one subproblem to each computing resource yields a natural parallelization of the problem. Domain decomposition can be done either algebraically (on the matrix) or geometrically (on the mesh). The additive Schwarz method is an algebraic domain decomposition method. It goes back to an iterative method originally developed by Schwarz [Sch70], which is now referred to as multiplicative Schwarz. Due to its practical applicability and natural parallelization, the additive Schwarz method has been rediscovered and improved several times to implement domain decomposition methods for the solution of partial differential equations [Bab57; DW87; SBG96; TW05]. If additive Schwarz is used as a preconditioner in PETSc, the global matrix is partitioned algebraically and distributed to all computing resources. On each resource, the local subdomain problem is either solved directly or by means of an iterative method. The results from all resources are gathered, added on the interface — giving the method its name — and scattered back.

[1] `http://glaros.dtc.umn.edu/gkhome/views/metis`

### Solution of the Nonlinear System

The shallow water equations result in a nonlinear system of equations. RMA·Kalypso employs Newton's method to transform the problem to a series of linear problems. In each linear solve, the Jacobian matrix, i. e. the matrix of all first-order partial derivatives, is assembled. Each computing resource is assigned a subset of elements for which to build the local element stiffness matrices. In each Newton iteration, all resources add their local matrices to a global, distributed Jacobian matrix. This matrix is managed by PETSc and used in the solution process. An inexact Newton method [DES82], where the linear problems are only solved up to some error, can also easily be implemented by relaxing the convergence limits of the inner Krylov iteration. Inexact Newton methods have the advantage that a fewer total of inner iterations may be needed to obtain convergence of the outer loop.

### Performance Comparison

PETSc allows to set the combination of a preconditioner and a Krylov solver by a simple configuration file. This makes it easy to compare the performance of the two different global approaches, i. e. Newton with (1) a direct linear solve or (2) an iterative Newton-Krylov-Schwarz solve, where the Schwarz domain decomposition is treated as a parallel preconditioner of the Krylov method. The results of this performance comparison can be found in Appendix C on page 145. In the examined test case, the second approach performs slightly better than the first, but both show a total improvement over the original shared-memory implementation. It was not a goal of this thesis to find the optimal configuration for the level 1 parallelization.

### Limitations and Suggestions for Improvement

Only parts of the calculation core RMA·Kalypso have been parallelized, while the rest of the program still runs sequentially, i. e. performing the same operations on the same data. All data is kept on all computing resources, which results in the same memory requirements for the parallel application as if it was executed on a single computing resource. This has the implication that the simulation does not scale to arbitrarily large domains. The implementation of an approach for data distribution could have be done based on the PETSc `DMMesh` or `DMComplex` distributed mesh objects [BB+11], but this was out of the scope of this thesis.

Although the presented level 1 MPI-parallelization of RMA·Kalypso with PETSc could as well be executed across several clusters (e. g. using MPICH-G2, see Subsection 2.1.3),

performance would decrease dramatically. The reason for the expected degradation is the large number of messages and amount of data that has to be sent in both direct and iterative solvers. The major requirement of PETSc, a fast, low-latency interconnect, is not met in this case. However, no performance measurements have been done that would demonstrate this performance loss, because MPICH-G2 is still not supported in the German D-Grid infrastructure.

The following section introduces a loosely-coupled algorithm suited for multi-cluster environments (level 2). The computations on each cluster will then be parallelized using one of the level 1 algorithms presented before. This choice is independent of the level 2 algorithm.

### 7.3.3. Level 2: Domain Decomposition in the Grid

In order to overcome the inter-cluster communication problem, a second level of parallelism with fewer messages to be exchanged is to be developed. The requirement on this level is that the amount of communication between subdomains shall be minimized. Changes in the calculation core shall be kept to a minimum. This section shows an attempt to parallelize the execution of RMA·Kalypso in the grid using a geometric domain decomposition method. An à-priori partition of the model is to make it possible to scale to larger flood models.

The class of geometric domain decomposition methods is motivated by a partition of the mesh on which the (local) numerical discretizations are based. This is in opposition of algebraic methods that partition the matrix resulting from the (global) discretization of the problem. The hydrodynamic model RMA·Kalypso has a two-dimensional finite element discretization, in which the unknowns (water level and flow velocities in x and y) are located at the nodes of the mesh. A finite element mesh partition into multiple non-overlapping subdomains (substructures) is a mapping of the set of finite elements in RMA·Kalypso (triangles and quadrilaterals) onto subdomains.

If the total domain covered by the set of elements is denoted by $\Omega$, a partition of this set into $n \geq 2$ subdomains $\Omega_i$, $1 \leq i \leq n$, is

$$\Omega = \bigcup_{i=1}^{n} \Omega_i.$$

The respective subdomain boundaries containing the boundary nodes (and associated boundary unknowns) are denoted as $\partial\Omega_i$. It will be assumed that subdomains do not overlap and that the boundary nodes of adjacent subdomains are coincident. Furthermore, no finite element may be split between subdomains.

**Variant 1: Direct Substructuring**

The goal of this section is to derive a distributed parallel domain decomposition algorithm based on a parallel direct substructuring method found in Saad [Saa03]. First, this original method, which Saad presented in a global formulation, will be explained. Afterwards, this global formulation will be rewritten in a form suitable for implementation in a distributed system. The following assumptions are common to both Saad's formulation and the distributed formulation to be developed.

The matrix associated with the linear problems $Ax = f$ resulting from the discretization and linearization of the shallow water equations in RMA·Kalypso will conceptually be split into interior and boundary parts. $A$ is called stiffness matrix, $x$ is the vector of unknowns at the nodes of the mesh, and $f$ is the load vector.

$$A \begin{pmatrix} x \\ y \end{pmatrix} = \begin{pmatrix} f \\ g \end{pmatrix} \text{ with } A = \begin{pmatrix} B & E \\ F & C \end{pmatrix}.$$

Solving for $x$ yields

$$x = B^{-1}(f - Ey),$$

which can be substituted for $x$ in the second equation $Fx - Cy = g$. The resulting system

$$(C - FB^{-1}E)y = g - FB^{-1}f \tag{7.1}$$

is called *reduced system*, where $C - FB^{-1}E$ is called the *Schur complement* matrix of $B$ in $A$. For practical purposes in the implementation of the following approach, [Saa03] sets

$$\begin{aligned} E' &= B^{-1}E \\ f' &= B^{-1}f \\ S &= C - FE' \\ g' &= g - Ff'. \end{aligned}$$

Substituting into Equation 7.1 gives the final solution in the simple form

$$\begin{aligned} y &= S^{-1}g' \text{ (boundary nodes)} \\ x &= f' - E'y \text{ (interior nodes).} \end{aligned}$$

For a partition of the mesh into $n$ subdomains, [Saa03] further decomposes the matrices

$B$, $E$, $F$, and $C$ into the respective parts local to subdomain $i \in [1, n]$:

$$\begin{pmatrix} B_1 & & & & E_1 \\ & B_2 & & & E_2 \\ & & \ddots & & \vdots \\ & & & B_n & E_n \\ F_1 & F_2 & \cdots & F_n & C \end{pmatrix} \begin{pmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \\ y \end{pmatrix} = \begin{pmatrix} f_1 \\ f_2 \\ \vdots \\ f_n \\ g \end{pmatrix}. \tag{7.2}$$

In this block structure $B_i$, $f_i$, and $x_i$ are the stiffness matrix, load vector, and vector of unknowns in the interior of subdomain $i$. Further, $y$ is the vector of all boundary unknowns, $g$ the boundary load vector, and $E_i$ and $F_i$ are coupling matrices between the interior and boundary nodes in subdomain $i$. Finally, the square block $C$ is the coupling matrix of all boundary nodes. $C$ contains contributions from all subdomains, i. e. $C = \sum_{i=1}^{n} C_i$.

Based on this partition of the matrix, a distributed formulation can be derived intuitively. It results from an à-priori partition of the mesh and the local assembly of the linear systems on all subdomains:

$$\begin{pmatrix} B_i & E_i \\ F_i & C_i \end{pmatrix} \cdot \begin{pmatrix} x_i \\ y_i \end{pmatrix} = \begin{pmatrix} f_i \\ g_i \end{pmatrix}.$$

The Schur complement approach depicted above results in a 3-step domain decomposition method titled *block-Gaussian elimination*, which entails solving the systems involving the interior nodes for $f'$ and $E'$ (step 1), assembling and solving the reduced system $Sy = g'$ for the boundary nodes $y$ (step 2), and correcting the solution $f'$ for the interior nodes by back-substitution of the boundary solution $y$ in the equations for $x$ (step 3). Step 1 and step 3 may be carried out in parallel on all subdomains.

In step 1, both $E'$ and $f'$, and, accordingly, the local contribution of subdomain $i$ to $S$ and $g'$ in Equation 7.1, can be computed in parallel:

$$\begin{aligned} E'_i &= B_i^{-1} E_i \\ f'_i &= B_i^{-1} f_i \\ S_i &= C_I - F_i E'_i \\ g'_i &= g_i - F_i f'_i. \end{aligned}$$

This step requires solving systems involving $B_i$ with $(n_i + 1)$ right hand sides and $2(n_i + 1)$ matrix-vector multiplications.

Step 2 requires gathering the coupling matrix $S_i$ and right hand side vector $g'_i$ from all subdomains for setting up and solving the reduced system $Sy = g'$. Afterwards, the

boundary solution vector $y$ needs to be scattered back to all subdomains.

Step 3 can, again, be performed in parallel for all subdomains once the boundary solution vector has been distributed. Only one additional matrix-vector multiplication is required:

$$x_i = f_i' - E_i' y_i.$$

The assembly of the reduced system and its direct solution is, in general, regarded to be an expensive operation, as the density and size of the reduced system depend on the connectivity of the subdomains and the overall boundary size. As a consequence, the sketched algorithm can only be applied in practice if the interface size is small. There are only two synchronization points (between steps 1/2 and 2/3) and the extra work due to solving a system with several right-hand sides and matrix-vector multiplications is negligible, so this approach is regarded as a possible solution to loosely coupled simulations in a distributed environment. In this thesis, as the direct substructuring approach is employed on the second level of the multilevel parallelization of RMA·Kalypso, where only a few large subdomains will be present, the incurred overhead in step 2 is small compared to the computational cost of step 1. One possible alternative will be explained in the following section on iterative substructuring.

**Variant 2: Iterative Substructuring**

This variation of the substructuring approach also uses a partition of the mesh into non-overlapping subdomains with subdomain boundaries $\partial\Omega_i$. The common internal boundary of two subdomains $i$ and $j$ of the partition will be denoted as $\Gamma_{ij} = \partial\Omega_i \cap \partial\Omega_j$.

In the described iterative substructuring method, an internal boundary $\Gamma_{ij}$ is treated like an external boundary of subdomains $i$ and $j$ with either a Dirichlet or Neumann boundary condition. Dirichlet conditions fix the value for $g$ on the boundary, whereas Neumann conditions specify a value for the normal derivative $D_n g$, or flux across the boundary. In the shallow water model described in Chapter 3, Dirichlet boundary conditions correspond to a fixation of the water level $h$ and Neumann conditions dictate $\mathbf{q}$, the specific flow:

$$h_i = h_j \text{ on } \Gamma_{ij}$$

$$Q_{n,i} = \mathbf{q_i} \cdot \mathbf{n_i} = -Q_{n,j} \text{ on } \Gamma_{ij},$$

where $\mathbf{n_i}$ is the unit outward normal vector to boundary $\Gamma_{ij}$ in $\Omega_i$.

In the finite element method, Neumann conditions are also called *natural* boundary conditions, as they appear as an additional term on the right hand side $g$. Dirichlet conditions are called *essential* boundary conditions and lead to an elimination of

equations and unknowns in the system. Similarly, the elimination entails additional terms on the right hand side.

In the first step of the proposed algorithm, all internal boundaries $\Gamma_{ij}$ of a subregion are colored with $N$ for a Neumann condition in subdomain $i$ and $D$ for a Dirichlet condition in subdomain $j$, so that each subregion has at least one $D$-colored boundary. This ensures the non-singularity of the problem in steady-state simulations. $D$-colored boundaries in a subregion receive the current values for $h$ from their neighboring region and solve a Dirichlet problem for $\mathbf{q}$, while $N$-colored boundaries receive the normal flow $Q_n$ and solve a Neumann problem for $h$.

A cross-exchange of Dirichlet and Neumann boundary values occurs at each iteration step. In order to obtain a global convergence result, all regions have to test for convergence on the boundary after each substructuring iteration. They report either convergence, non-convergence, or divergence to the root node, which issues the command to perform another substructuring step in case any process is not converged or stop the simulation if any process has diverged. Toselli and Widlund [TW05] have shown that iterative substructuring algorithms incorporating the cross-exchange of boundary conditions, such as the one sketched above, are equivalent to preconditioned iterative (Krylov) methods for solving the Schur complement system (Equation 7.1).

#### Application in a Nonlinear Solver

As the shallow water model results in a nonlinear system of equations, a Newton-like linearization, line search, or trust region method has to be applied. The convergence of the iterative substructuring method — or the solution of the reduced system (Equation 7.1) in the direct substructuring method — means that all processes can continue with the next nonlinear iteration. Another global convergence test is required to determine the convergence state of the nonlinear method. As all processes already report their state to the root node at the end of a substructuring step, information about the nonlinear iteration can simply be included in that report. As in inexact Newton methods, it is not strictly required that convergence limits of the linear solution are tight.

Mackens et al. [MMV99] and Menck [Men99] reported on a similar substructuring method for the nonlinear system motivated by the need to couple nonlinear, iterative solvers in chemical engineering. They assume that $f$ and $g$ are functions that have to be minimized independently by stepwise approximations of their internal unknowns $x$ and boundary unknowns $y$. Their joint Jacobian matrix (in the linearization) can be written as

$$J = \begin{pmatrix} B & E \\ F & C \end{pmatrix} = \begin{pmatrix} f_x & f_y \\ g_x & g_y \end{pmatrix}.$$

A step in the proposed method, which they call *Tangential Block-Newton* (TBN), consists of two half-steps in $f$ and a step in $g$. First a regular Newton step $x^+ = x_n + \Delta x$ for $f$ in the direction $\Delta x = -f_x^{-1} f(x_n, y_n)$ is performed. It is followed by a Newton step $y_{n+1} = y_n + \Delta y$ for $g$ along the tangential space of the manifold $M = \{(x,y) | f(x,y) = f(x^+, y_n)\}$, which results in the reduced system [Men99]:

$$\Delta y = -S^{-1} g(x^+, y_n).$$

In this case, matrix $S$ is the Schur complement of $f_x$ in $J$ (compare Equation 7.1):

$$S = g_y - g_x f_x^{-1} f_y$$

The first half step for $x$ is then corrected in the tangential direction of $M$, i. e. $x_{n+1} = x^+ - f_x^{-1} f_y(x^+, y_n)\Delta y$ [Men99]. TBN is equivalent to the application of the direct substructuring method described above to the Jacobian matrix in each Newton step.

### Limitations and Suggestions for Improvement

It turned out to be impossible to set up a simulation across multiple clusters in the D-Grid. There were technical problems with the grid middleware that could not be solved within the time frame of this work. The only tests used two separate workstations in a local area network connected via Ethernet, which showed that this approach works in principle. As will be described in the following section, there is still a lack of support for developing and deploying a distributed application in existing grid middleware.

No performance tests have been conducted for level 2 of the algorithm due to the missing software infrastructure in the German D-Grid, so the effectiveness of this approach has yet to be proven. In the prototype, the second variant (iterative substructuring) was implemented as it required fewer changes in the calculation core. For a small subdomain interface size compared to the number of internal nodes, the first variant may provide the better performance, especially in slow communication settings. This assumption could not be further examined. For larger subdomain interfaces, there is potential for improvement of substructuring variant one — or the TBN method — by calculating the $g$ step in an approximate, iterative approach. Menck [Men99], an author of the TBN method, has demonstrated that the global nonlinear iteration will retain a linear convergence rate even when several $f$ steps are performed in a row, followed by an approximate $g$ step.

In fact, the direct substructuring approach is very similar to the multifrontal method for direct solution applied in MUMPS. Direct substructuring keeps the symbolic analysis phase of the solution process local to the subdomains, which is more efficient, but prohibits load balancing between subdomains in the factorization phase [GMR05].

## 7.4. The "Big Picture": Flood Simulation Grid-WPS

This section describes the technical realization of the flood simulation service spanning both levels of the multilevel parallelization approach. The implementation is split into two components: a Kalypso 1D2D Grid-WPS and a flow model coupling service. As the first step, the requirements for these services will be analyzed. Second, the Grid-WPS interface will be described. This section concludes with a presentation of the flow model coupling service.

### 7.4.1. Flood Simulation Use Cases

In Section 7.1, the presumption was made that users of a flood simulation service assume one of two different roles: flood modelers or flood managers. The following flood simulation use cases (see Figure 7.1) have been identified, which represent the essential tasks carried out by flood managers when simulating a flood scenario. These use cases conform to geoprocessing operations in the processing phase, which have been defined previously (see Chapter 4):

- Set boundary conditions (water stage or discharge hydrographs). Hydrograph data is either provided by the flood manager or comes from a Sensor Observation Service.
- Set initial conditions (lake solution, artificial, or pre-computed flow state). Lake solutions, if applicable, can be generated automatically for a given boundary condition. Flow states from previous simulations can come from a Web Coverage Service.
- Control execution in a distributed environment (start simulation, monitor progress).
- Retrieve the results (intermediate or final).

### 7.4.2. Kalypso 1D2D Grid-WPS Interface

The purpose of the Kalypso 1D2D Grid-WPS interface to be developed is a grid extension of geoprocessing operations for flow simulation provided by the Kalypso Simulation Platform. These operations will define a WPS application profile for flood simulation. The profile will then be implemented using the Grid-WPS framework described in Chapter 5.

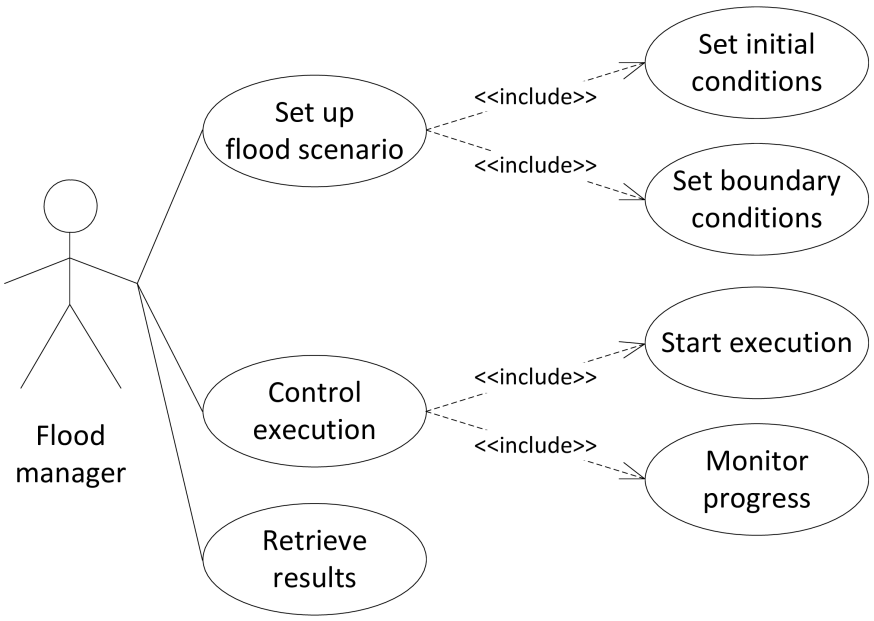



**Figure 7.1.:** Flood simulation use cases for flood managers.

### The Kalypso Simulation Platform

The Kalypso Simulation Platform provides a user interface to numerical simulation models in water resources, among which is Kalypso 1D2D [SA+09] employing the low-level calculation core RMA·Kalypso (see Section 7.3). Kalypso is also an OGC-conforming geoprocessing server for simulations. The server is fully integrated into the base of the Kalypso Simulation Platform and provides WPS processes for the different Kalypso simulation models. The Kalypso Simulation Platform either starts its own, private simulation server or connects to a remote server instance over the WPS interface. A Kalypso simulation server runs inside a Jetty web application container[1] embedded into Eclipse[2].

Kalypso 1D2D can be characterized as a fat-client GIS user interface. This system is appropriate for building a flood simulation service prototype because, to the author's knowledge, it is the only current service interface to a flow model supporting the WPS standard. The greatest benefit of using the Kalypso Simulation Platform lies in its extensibility based on Eclipse plug-ins, simple integration with an existing spatial

[1] http://jetty.codehaus.org
[2] http://www.eclipse.org/jetty

data infrastructure, as well as giving users of Kalypso 1D2D an easy-to-use portal to running their simulations in the grid.

### Kalypso 1D2D WPS Profile

WPS application profiles are supposed to enable interoperable processing, service discovery, and orchestration in a specific domain, such as two-dimensional spatial overlay operations (e. g. intersection of geographic features) or, as in the topic of this section, hydrodynamic modeling.

Kalypso 1D2D is not restricted to running the RMA·Kalypso hydrodynamic model. It provides a WPS profile to unstructured, one- and two-dimensional flow models, in general. As a precondition to interoperability with a diversity of numerical models, the flow model needs to be stored as a collection of standardized GML files in accordance with the Kalypso 1D2D GML application schema that describes the unstructured mesh for the study area (discretization), flow resistances, boundary and initial conditions, as well as simulation control parameters and results.

Flow resistances are a GML feature with a polygon geometry referring to a flow resistance class. All available flow resistance classes for a model are stored inside a database. They define eddy viscosities and equivalent sand roughness coefficients. For interoperability with a SOS, boundary condition hydrographs are kept in the form of GML observations[1] at a given point location and specifying the spatial direction of measurement. Discharge hydrographs (describing the amount of flow) typically refer to the total discharge across a profile section. In the terminology of Kalypso 1D2D, these sections are called *continuity lines* and are part of the mesh definition. Water level hydrographs can also be assigned to all mesh nodes of a continuity line. A flood scenario can, finally, be defined by a *calculation unit*, which bundles a (sub-) domain of the mesh, which can be either one- or two-dimensional, together with its control parameters, continuity lines, initial flow state, and boundary conditions. A *coupled calculation unit* for a 1D-2D model may be composed from calculation units for the respective subdomains.

The Kalypso 1D2D simulation for RMA·Kalypso has been split into three separate WPS processes defining the WPS profile for flow simulation in this thesis (compare Figure 7.1): `PreRMAKalypso` (for setting up a flood scenario), `ExecuteRMAKalypso` (for running the numerical model), and `PostRMAKalypso` (for evaluation of the results). `PreRMAKalypso`, prepares the RMA·Kalypso input files according to the GML data structure. Initial conditions may be provided in the form of previous results. `ExecuteRMAKalypso` then takes the prepared files, starts the simulation, and monitors

[1] According to the OGC Observations and Measurements v2.0 also published as ISO/DIS 19156.

its progress. Finally, `PostRMAKalypso` evaluates the result files and transforms them back to GML.

Due to this tripartite design, the calculation core can easily be replaced by a different one, under the condition that the new flow model abides by the Kalypso 1D2D WPS profile. In consequence, this core has to define conforming processes acting as an adapter between the numerical model and the Kalypso 1D2D model data structure.

#### Kalypso 1D2D Grid-WPS

The Kalypso 1D2D Grid-WPS is the entry-point for starting a flow simulation in the grid. It fulfills the purposes of configuring a coupled flow model for a given flood scenario, setting up the simulation in the grid, monitoring its execution, and retrieving the results. This service is usable by both flood modelers and flood managers.

There are two different ways simulation processes can be executed. In the original implementation, a simulation could just be run on the Kalypso server (`DefaultProcess`). The Grid-WPS presented in this thesis supplements Kalypso with a novel approach to submit simulation processes as grid jobs to a Globus Toolkit 4 GRAM service (`SimpleGridProcess`). This means a simulation can, for instance, be executed on a computing cluster or another grid resource. The `SimpleGridProcess` requires a GRAM endpoint URI and valid grid credentials. These credentials are obtained from the current Grid-WPS context. In this way, a simulation can use either the credentials of the Kalypso server or the credentials a Grid-WPS user has delegated to the server.

Both process implementations run the simulation in a so-called *sandbox*. The idea behind this concept is that the process is confined to run inside a working directory where all process inputs and outputs, and, usually, also the executable file, are placed (compare Chapter 5, Section 5.2). The process publishes its sandbox directory to the service that started the process, so that this information can be used, for example, to access intermediate outputs and to monitor the progress of its execution. The execution environment of the process, local or in the grid, remains transparent to the service, as long as it is entitled to access the sandbox directory. A `SimpleGridProcess` relies on the assumption that there is a GridFTP service running on the GRAM server and publishes the sandbox directory as a URL with the GridFTP protocol.

As a stand-alone component, the Kalypso 1D2D Grid-WPS already allows to run a simulation in the grid in conformance with the level 1 parallelism sketched in Section 7.3, i. e. submission to a computing cluster. The parallelization on level 1 is achieved by submitting an MPI job with multiple processes to the GRAM. The only communication on this level is taking place inside the cluster. From the perspective of the Kalypso

1D2D simulation, the MPI grid job is treated as an atomic process. Distribution of work for a single subregion is achieved on the algebraic level.

For the support of level 2 parallelism, i. e. execution across multiple computing clusters in a grid, a user can set up his model by defining a coupled calculation unit with multiple adjacent one- or two-dimensional subdomains. Originally, the coupled model could only be run in a single process. The Grid-WPS adds the alternative to run the calculation units in a distributed fashion using separate processes for each subdomain coupled at the interfaces (internal KALYPSO 1D2D continuity lines). In this way, the subdomains can now be distributed across different loosely-coupled clusters in the grid.

In case such a distributed simulation of a coupled model has been configured, `PreRMAKalypso` determines the adjacency information, internal boundary lines, and the types of boundary condition to be exchanged between adjacent regions automatically from the discretization model. This information is required to coordinate the distributed simulation. All calculation cores may immediately begin with the execution until the first synchronization point is reached.

In the following section, this Grid-WPS will be extended by a grid service component, called *flow model coupling service*, which has the task of coordinating the execution of a coupled simulation, i. e. when it is to be distributed across multiple grid resources using the second level of parallelism.

### 7.4.3. Flow Model Coupling Service

The flow model coupling service enables the interaction between coupled subregion models of a distributed KALYPSO 1D2D simulation. This service has been designed for use in scenarios where boundary information needs to be exchanged between flow models in a coordinated way. The implementation is tailored to the iterative substructuring algorithm sketched in Section 7.3 and the RMA·KALYPSO numerical model.

The distributed simulation setup ensures the correct execution of the algorithm on level 2 of the multilevel parallelization, i. e. communication across clusters. Each cluster handles one subdomain and uses many computational nodes to compute the subdomain solutions. For the subdomain solves, an algorithm from level 1 is used to distribute the work across the cluster nodes.

Once the separate simulations for a coupled calculation unit have been started on all clusters by calling `ExecuteRMAKalypso`, the flow model coupling service creates a grid resource in the flood simulation service at the respective cluster site. A flood simulation

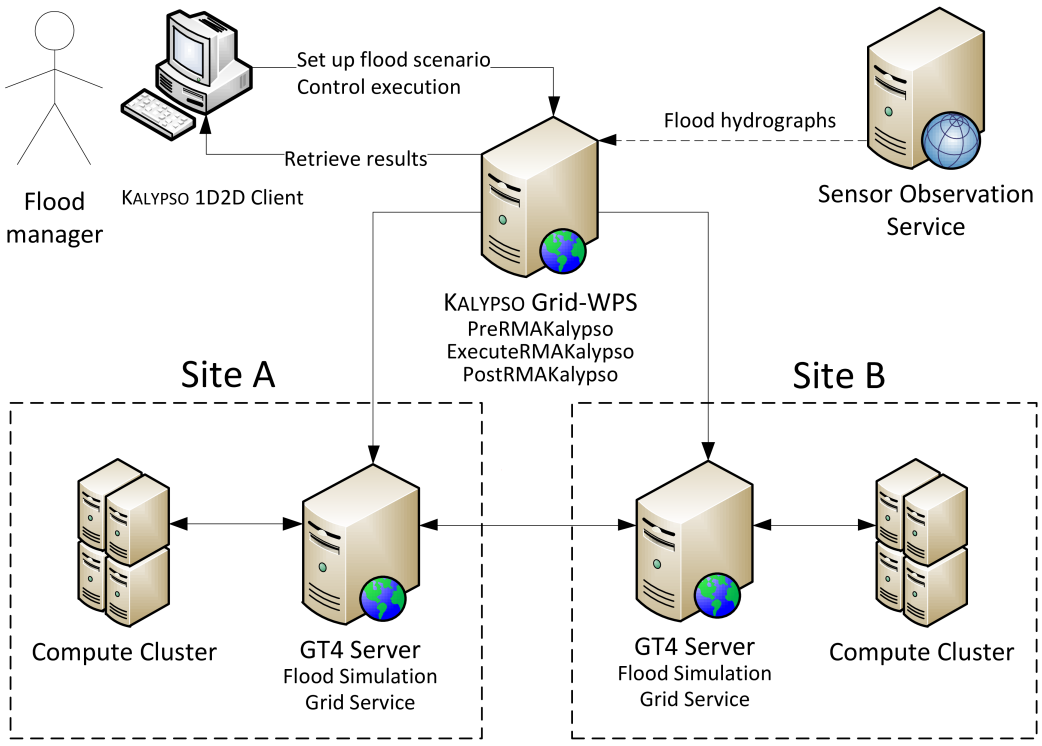



**Figure 7.2.:** Setup of the distributed KALYPSO 1D2D simulation across two sites.

grid resource is associated with the execution of exactly one subdomain on one cluster. It then sets up the communication between adjacent subdomains by monitoring the subdomain output files in the Grid-WPS sandbox directory (see Figure 7.2).

Once an iteration is completed, the flow model coupling service updates an internal resource property containing a reference to a file for each internal boundary line containing the current boundary data. Grid resources for adjacent subdomain are registered to receive notifications about new boundary data from their neighbors. Updated boundary data files are copied to the corresponding sandbox. The flow model coupling service operates in a similar fashion to the "ServOSims" framework developed by Floros and Cotronis [FC06] (see Section 7.2).

A crucial step is to determine the global convergence of the simulation. After each iteration, a subregion's flood simulation grid resource updates its convergence status. An arbitrarily selected root process in the tree of all subregions collects the convergence status of all other processes and reports back the global status. Each flood simulation

grid resource then issues the command to either continue the iteration, proceed with the next time step, or terminate the simulation. Commands are encoded in the file name of a special file in the sandbox directory.

Exchanging boundary information in files is a very crude approach. Nevertheless, it has the advantage that no further communication mechanism, like RPC, needs to be implemented between the flow model coupling service and the calculation core. The most obvious disadvantage is a performance degradation because (1) the sandbox is typically accessed from the grid service via GridFTP and (2) both the service and the executable have to poll for file existence in specified (millisecond) intervals. As long as there is no middleware support for creating this communication link from service to executable and vice versa, the depicted approach provides an acceptable yet unstandardized workaround.

## Outlook

The WPS application profile for flood simulation developed in this chapter together with the Kalypso 1D2D software is a step towards an open architecture for flood modeling. [KJ+04] have identified open architecture as the crucial advancement in software development maturity to creating interoperable software applications modeling hydraulic and hydrologic problems. They define software architecture as "the conceptual structure and logical organisation of a computer or computer-based system". Currently, hydraulic software is in the development stage of closed architecture lacking the possibility to "plug in" novel software components into existing systems. It is often based on proprietary, usually commercial products that are not compatible with each other and serve the specific needs of an organization. Open architecture, on the contrary, is based on open standards and allows users to shape an application to their needs from "off-the-shelf" software products.

A possible complementary approach to coupling hydrodynamic models in an open architecture is the Open Modeling Interface (OpenMI). Gregersen et al. [GGW07] state that "OpenMI is a pull-based pipe-and-filter architecture, which consists of communicating components [...] that exchange memory-based data in a predefined way and in a predefined format". However, the existing implementation of the OpenMI Environment does not work in heterogeneous systems. Components implemented in the .NET and Java languages, for example, cannot be connected [HOS08]. Even though the framework would generally be suited to coupling adjacent 2D domains and controlling the simulation, there have been no endeavors to support a real distributed calculation, as in grid computing, using OpenMI so far.

# 8. Conclusions

The aim of this thesis had been to make a contribution to the implementation of open standards from spatial data and grid infrastructures in the field of flood modeling. For the first time, the process of flood modeling by two-dimensional hydrodynamic simulation — flow model creation, flood simulation, and results evaluation — has been formalized as a sequence of geoprocessing tasks. This was accomplished by the development of geoprocessing grid services for flood modeling. Two time-consuming and data-intensive tasks in this process were selected for parallelization and prototypical implementation. The novel approach now enables flood modeling experts, for example, to set up large-scale two-dimensional hydrodynamic models using standard-conforming digital elevation data services, to run their flood simulations remotely over the web, and to store, manage, and analyze their simulation results on one or more computing and storage resources in a grid infrastructure.

## 8.1. Summary of Results

Flood modeling tasks have been mapped to geoprocessing operations, which integrate into a spatial data infrastructure and which can be executed automatically, efficiently, and repeatedly in a computational grid. As a first step towards flood modeling in a grid, two standards commonly used in spatial data and grid infrastructures, the Web Processing Service (WPS) and the Web Services Resource Framework (WSRF), have been harmonized. This harmonization resulted in a generic procedure for the development of geoprocessing grid services, which has been called Grid-WPS. The feasibility of this procedure was shown by two prototypical implementations of flood modeling tasks based on the Globus Toolkit 4 grid middleware.

Through the parallelization of those two tasks and the application of grid technology, it was demonstrated how large volumes of geographic data can be processed effectively in a distributed computing environment. The individual tasks have been implemented as interoperable geoprocessing services. These services may now easily be reused in the context of another service-oriented application or chain of services. The result further shown that, in all cases, geographic data can also be processed more efficiently than using traditional methods. The qualities of service provided by a computational grid

could all be leveraged, such as high performance, reliability, availability, and security. These qualities are also desirable in a spatial data infrastructure, which can now be guaranteed by the integration of grid technology.

The first prototypical implementation, the *flow model discretization service*, employs a new strategy for the parallel creation of large, unstructured, two-dimensional computational meshes. This use case confirms that the process of flow model creation can successfully be implemented as a workflow of geoprocessing and geographic data services that can be executed automatically on a large number of distributed grid resources. The flow model discretization service draws its inputs from a standard web service for digital elevation data, such as it could be provided in a spatial data infrastructure. In this way, a digital elevation model can now be processed in a high resolution across arbitrarily large areas, which has not been possible to date. The second implementation, the *flood simulation service*, further demonstrates how to execute a hydrodynamic simulation on multiple clusters using a parallel domain decomposition approach in the grid and a standard library for the parallel solution of sparse linear systems inside a cluster. This two-level parallelization has overcome the limitations of using a single cluster execution environment. While the gridification of a tightly-coupled numerical model was shown to be a challenging task, a number of possible solutions could be presented. Together, the flow model discretization and flood simulation services developed in this thesis provide an efficient technology for mapping flood hazards according to the EU Floods Directive. Their use could both greatly accelerate the flood mapping process and enhance the quality of the created hydrodynamic models.

## 8.2. Outlook

[RJS05] have highlighted that true interoperability and cooperation in the flood modeling community requires geoprocessing services and geographic data to be given a well-defined meaning. They argue that this could be achieved by agreeing on a conceptualization of flood modeling terms, a domain ontology, which would allow the semantics of the flow model discretization and flood simulation services — and their inputs and outputs — to be defined. The implemented prototypes in this thesis make a step towards such an ontology by defining geoprocessing service interfaces based on a GML representation of the flood model data. Nevertheless, there remains the task of mapping those GML entities to domain ontology concepts.

Furthermore, a vision for the future could be a real-time, world-wide flood forecasting service. The framework laid out in this thesis provides the foundation for such a service. A flood model could be created merely based on the delineation of an area under investigation, available geographic data services, and flood hydrographs projected

from a real-time sensor data service or precipitation time series from a meteorological forecasting service. The chaining of all these services would be greatly simplified by the Grid-WPS framework.

## 8.3. Final Remarks

Even though this thesis mainly uses terminology from the grid community, many of the achievements can be applied to cloud computing as well. Both grids and clouds provide some aspect of IT as a service. The flow model discretization and flood simulation services could, for example, be seen as a form of "software as a service". However, clouds — in contrast to grids — are currently lacking a high performance computing environment for numerical simulation. The presented two-level parallel domain decomposition strategy could enable the distributed simulation of flow models in the cloud.

A global grid infrastructure many people hoped for has not yet emerged. Indeed, the metaphor of a computational grid providing computing power in a way electricity is provided by the electric power grid has been misleading. Unlike electricity as a commodity, the service provided by grids is much more ambiguous. All different kinds of services, hardware, and software are united in a large "pool" of resources provided by and available to many different virtual organizations. In this sense, a grid can be seen as a kind of collaborative "infrastructure as a service" cloud, in which grid middleware and cluster systems provide "platform as a service" for high-performance computing or a plethora of domain-specific applications. An analogy to the new field of "Smart Grid" technology[1] seems more appropriate. Future electricity networks are foreseen to improve with respect to the flexibility, reliability, and efficiency of the electrical grid infrastructure.

As grid technology and standards are still evolving, it is not easy to make a recommendation for the best technological solution to the problem of bringing an existing piece of software into a grid environment. WSRF, OGSA-BES, and JSDL are promising standards for grid services and job submission, but the developer of a grid application should carefully investigate if these standards are supported in the anticipated grid middleware and client software. Nevertheless, as [SB+10] have pointed out, future grid and cloud technologies will lead the way to a more effective sharing and utilizing of resources. Their present study confirms that domains with the demand for processing huge amounts of geographic data, as flood modeling, can only benefit from these new developments if they "go with the flow" pertaining to information technology and standards.

[1] See, for example, `http://www.smartgrids.eu`.

# Appendix

# A. Gaja3Dpar Implementation Details

The purpose of Gaja3Dpar is to provide a tool for the generation of discretization networks applied in 2D hydrodynamics. The Gaja3Dpar meshing software and its underlying methodology for 2D hydrodynamic model discretization is implemented in the numerical computing environment MATLAB, developed by The MathWorks, and uses the Triangle 2D mesh generator by Shewchuk [She96] for Delaunay triangulation. Gaja3Dpar is licensed as free software under the GNU Lesser General Public License. A copy of Gaja3Dpar may be requested at the Institute of River and Coastal Engineering[1].

## A.1. Original Version of Gaja3D

In the original version of Gaja3D by Rath [Rat07], a graphical user interface was included. The user interface worked in one of two modes, automatic or manual. The automatic mode was for the inexperienced user, hid some functions that should only be used with care, and provided a guide through the discretization process. Guidance was given by a number of consecutive dialog windows that asked the user to select all the required input files and to specify breakline detection and triangulation parameters. Status information was displayed on the screen with figures showing the intermediate results, such as an interpolated raster or breaklines. For the tool to be effective in real applications, some input data needed to be prepared in other software packages, such as GIS or CAD programs. This applied to the model domain boundary polygon and a suitable DEM.

All entered inputs to the meshing process chain were saved as a structured text document (XML). It was possible, afterwards, to apply the same set of meshing parameters to a number of terrain patches, which each consisted of a domain boundary polygon and a DEM for that domain. Gaja3D would execute the meshing process for each terrain patch based on the information in the process chain description. Terrain patches were only processed sequentially and completely independent from each other.

[1] `http://www.tu-harburg.de/wb`

Due to this restriction, the original version of Gaja3D was not capable of producing a single mesh for multiple adjacent terrain patches of a large domain.

Based on a boundary polygon and the DEM, Gaja3D first interpolated a regular raster surface of the DEM covering the domain. The user specified the raster resolution based on the desired detail of terrain features in the model. The second step consisted in a detection of these terrain features by specifying a method and threshold for slope classification. The areas with high slope were first identified as breakpoints and then connected to form breaklines. The breaklines could later be generalized by a Douglas Peucker line simplification algorithm, if desired. The final step in the process was a constrained Delaunay triangulation of the model boundary and the detected breaklines. The resulting mesh could be converted to a format for import in the flow models BCE-2D (MicroStation, Bentley Systems) or Kalypso 1D2D. Additionally, Gaja3D supplied methods for mesh quality assessment and subsequent mesh refinement based on element residuals.

## A.2. Parallel Version of Gaja3D (Gaja3Dpar)

A modularization and speed enhancement of the original tool was conducted in the course of this thesis to make independent functions of Gaja3D available as a new software library, which was called Gaja3Dpar. This was needed to be able to execute parts of the meshing process as a service in the grid. Some functionality of Gaja3D was omitted in Gaja3Dpar, while other has been replaced by an improved implementation, interfacing with Java and C/C++ libraries. There is no graphical user interface to Gaja3Dpar, but it is possible to specify a meshing process as a MATLAB script. Input formats were originally all ASCII text files, but now it is possible to load and save to a selection of GIS formats, which makes it easier to specify complex boundary polygons and evaluate intermediate results.

The necessity for a new implementation arose when the software was to be executed as a standalone tool. MATLAB can compile a function to a command line executable file or to a C/C++ shared library using the MATLAB Compiler add-on. The additional product MATLAB Builder JA can convert functions to Java classes. Using the MATLAB Compiler seemed to require the least effort to provide a standalone version of Gaja3D, as opposed to a MATLAB-independent implementation in a different programming language, e. g. C/C++, Java, or Python. The latter option would have needed a fast implementation of the image processing and large matrix handling facilities provided by MATLAB. An advantage of using MATLAB that has been retained is its scripting-like user interface that allows for rapid prototyping in Gaja3Dpar. This makes it easy to perform additional data analysis beyond the original Gaja3D's capabilities.

Most library functions of Gaja3Dpar can be accessed via a single point of entry `Gaja3D` object that encapsulates data management and meshing parameters. Standard MATLAB arrays use value semantics, but all Gaja3D classes working with large datasets have been designed with MATLAB handle (call-by-reference) semantics, which prohibits memory copies of the object to be made. Handle objects also support events and listeners as well as dynamic properties. The `Gaja3D` object contains references to the loaded DEM data as one or more scattered point clouds (`demTin`), the interpolated rasters for each tile (`demGrid`), their detected breaklines (`breaklines`), and the final mesh (`modelTin`).

## A.3. Usage

A Gaja3Dpar user can interact with a MATLAB workspace by calling API methods on a `Gaja3D` object. The flexibility of the API (see Figure A.1 and Figure A.2) enables the user to control the results of each intermediate step interactively in the MATLAB workspace and to correct the meshing parameters to his needs.

A typical calling sequence is as follows: He first creates an instance of `Gaja3D` and then sets the domain boundaries and digital elevation model (point cloud) from file data. Afterwards he creates a regular raster (`createGrid`), configures a feature detection method and starts the breakline detection process (`detectBreaklines`). Finally, the discretization mesh is triangulated (`createTin`) and elevations are assigned by interpolation from the raster (`assignElevations`). Once the results have been saved, the `Gaja3D` instance can be destroyed.

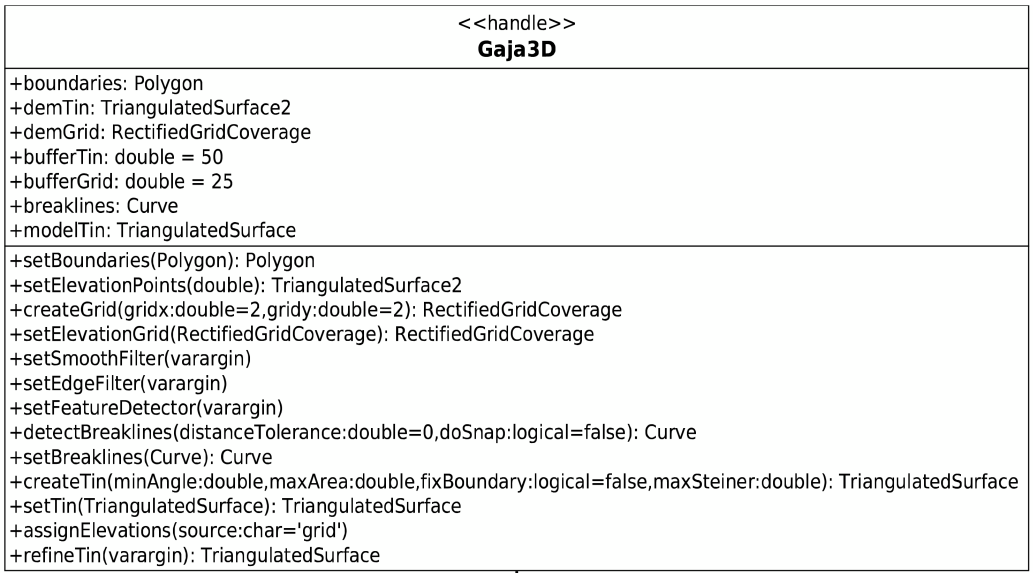



**Figure A.1.:** Class `Gaja3D` attributes and operations. Dependencies can be found in Figure A.2.

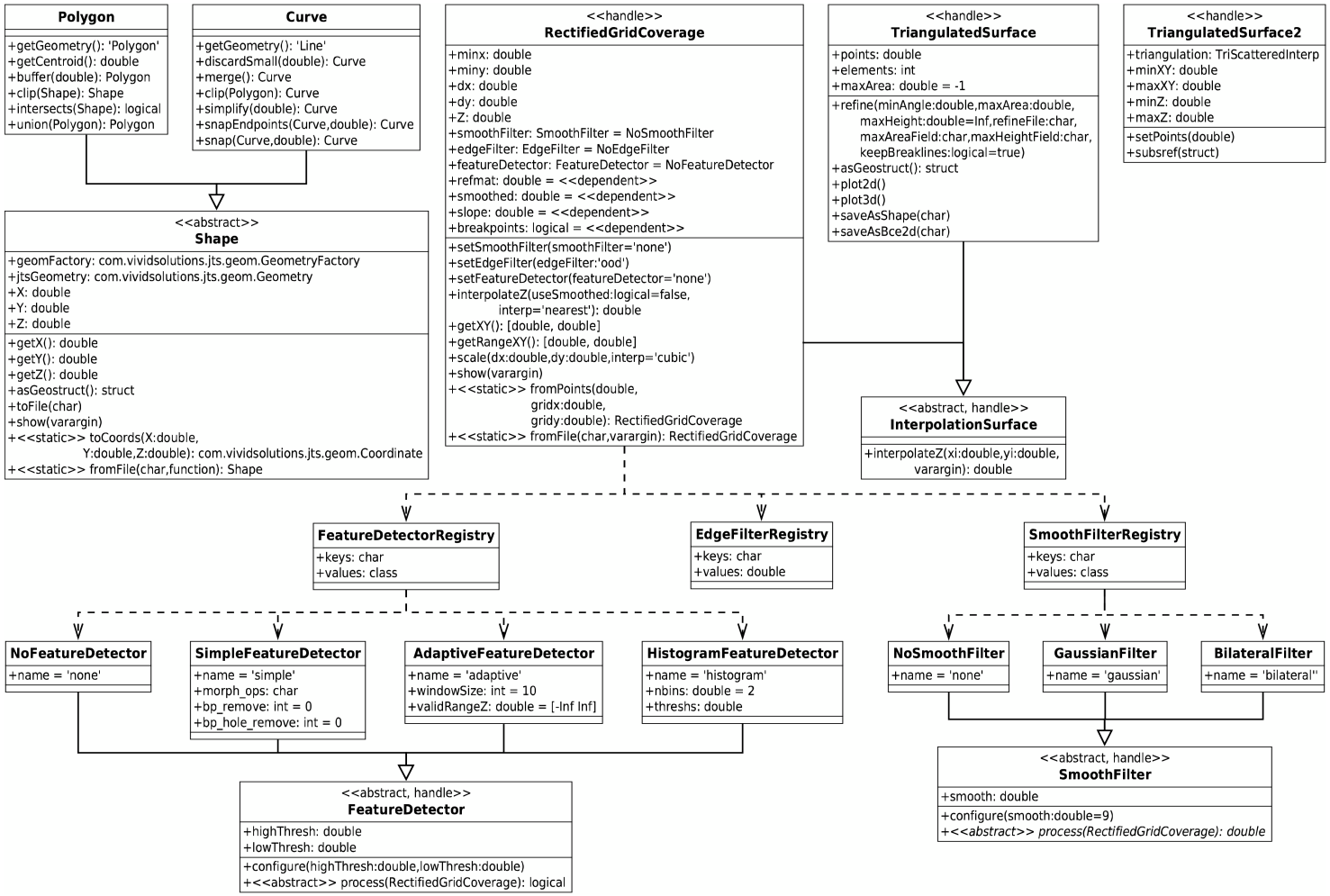



**Figure A.2.:** UML diagram of the dependencies of `Gaja3D`.

# B. Mesh of the Tidal River Elbe

The flow model discretization service prototype was applied in the KLIMZUG-NORD project[1]. In this way, the parallel meshing process could be validated at the example of digital terrain data from the Hamburg Metropolitan Region (see Figure B.1). The data was compiled from data sets delivered by different German federal state authorities. The data sets had to be integrated at the state boundaries to form a contiguous base terrain model and was provided to project partners via a Web Coverage Service.

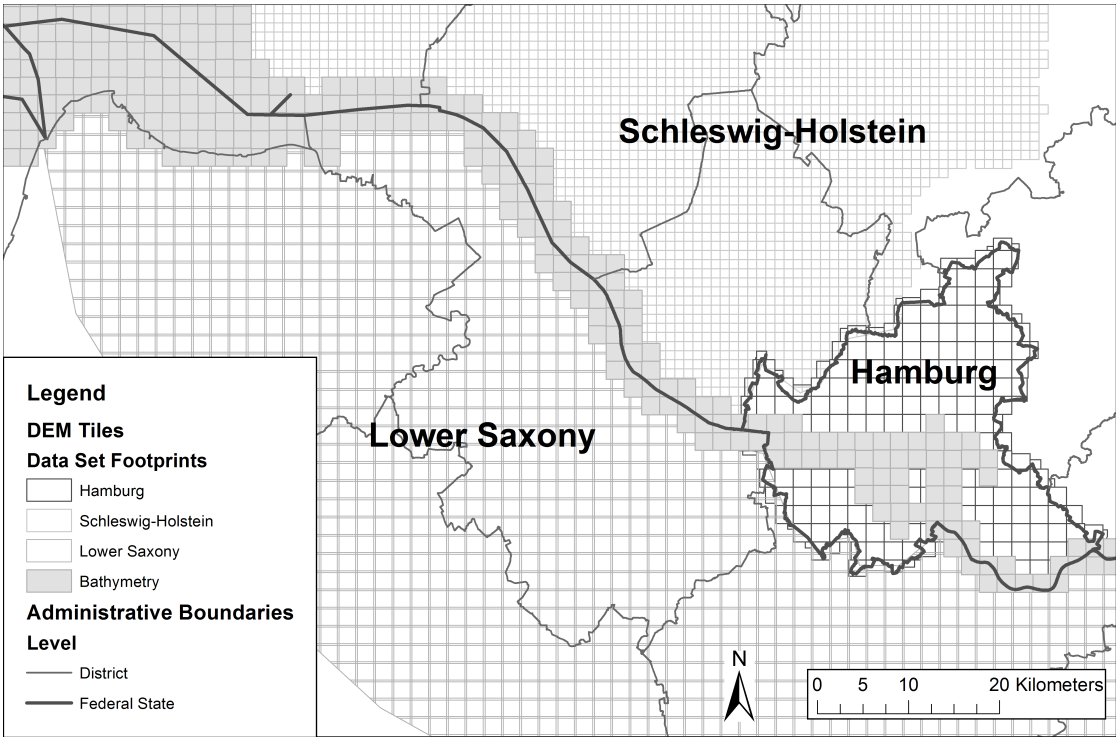



**Figure B.1.:** Digital elevation model in tiled raster format for the Hamburg Metropolitan Region including bathymetry data for the river Elbe. The data volume amounts to a total of $\approx 5 \cdot 10^9$ raster cells.

[1] `http://klimzug-nord.de`, funded (2009-2014) by the German Federal Ministry of Education and Research (BMBF)

The meshing process was distributed in the grid according to the simple partition of the Elbe river water body boundary between the North Sea and the upstream end of the tidally influenced part near Geesthacht (see Figure B.2). Unfortunately, the improved partition (see Figure B.3) could not be used because rotated rasters were not supported by our WCS implementation. The automatic detection of structural features resulted in a set of about 50.000 breaklines with a total length of 6.500 km partially depicted over the DTM in Figure B.4. Breakline strength is visualized by graduated line widths.

A finite element mesh consisting of ca. 1.2 million triangles was created from the detected breaklines. The average length of a triangle edge in the mesh is about 50 m. A 3D perspective view of this mesh near Hamburg is shown in Figure B.5. The camera direction is north-west from a point south-east of Hamburg. An embedded map displays a detailed view of the mesh for the same extend as in Figure B.4.

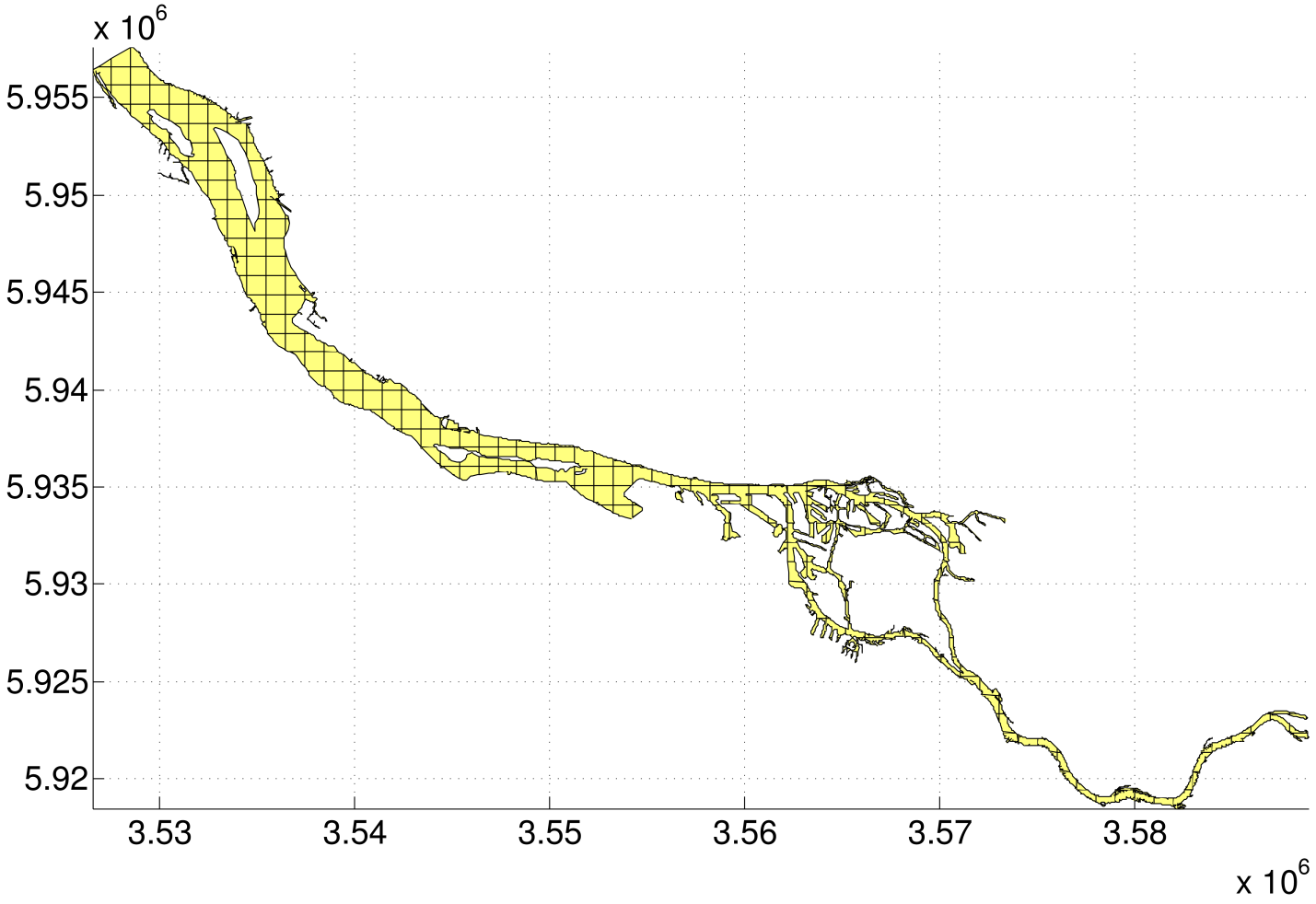



**Figure B.2.:** Simple partition of the Elbe river water body boundary (416 tiles, grid size 1 by 1 km). A detailed view can be seen in Figure 6.5.

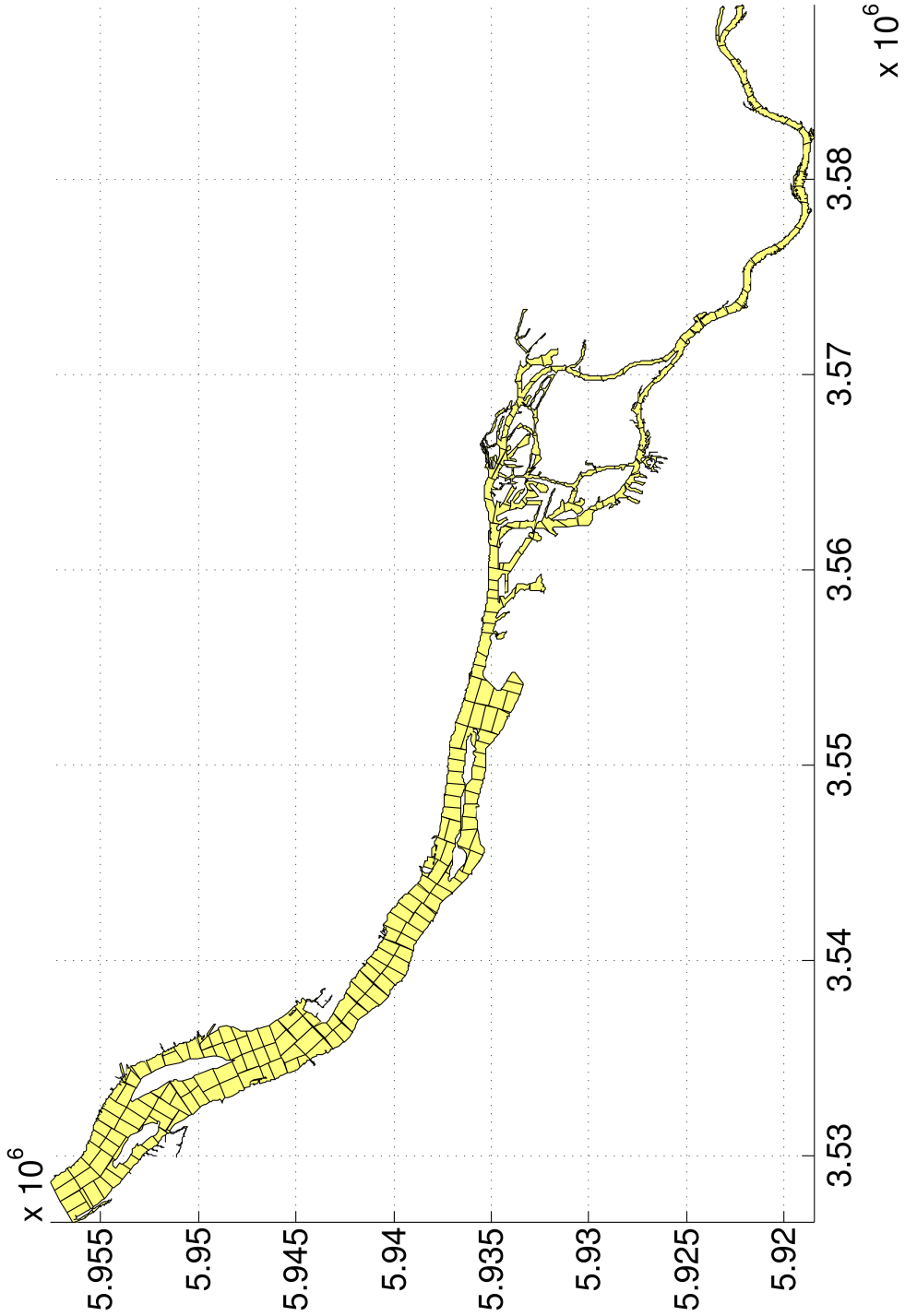



**Figure B.3:** Improved partition of the Elbe river water body boundaries (275 tiles, 1 $km^2$ maximum rotated tile area). A detailed view can be seen in Figure 6.6.

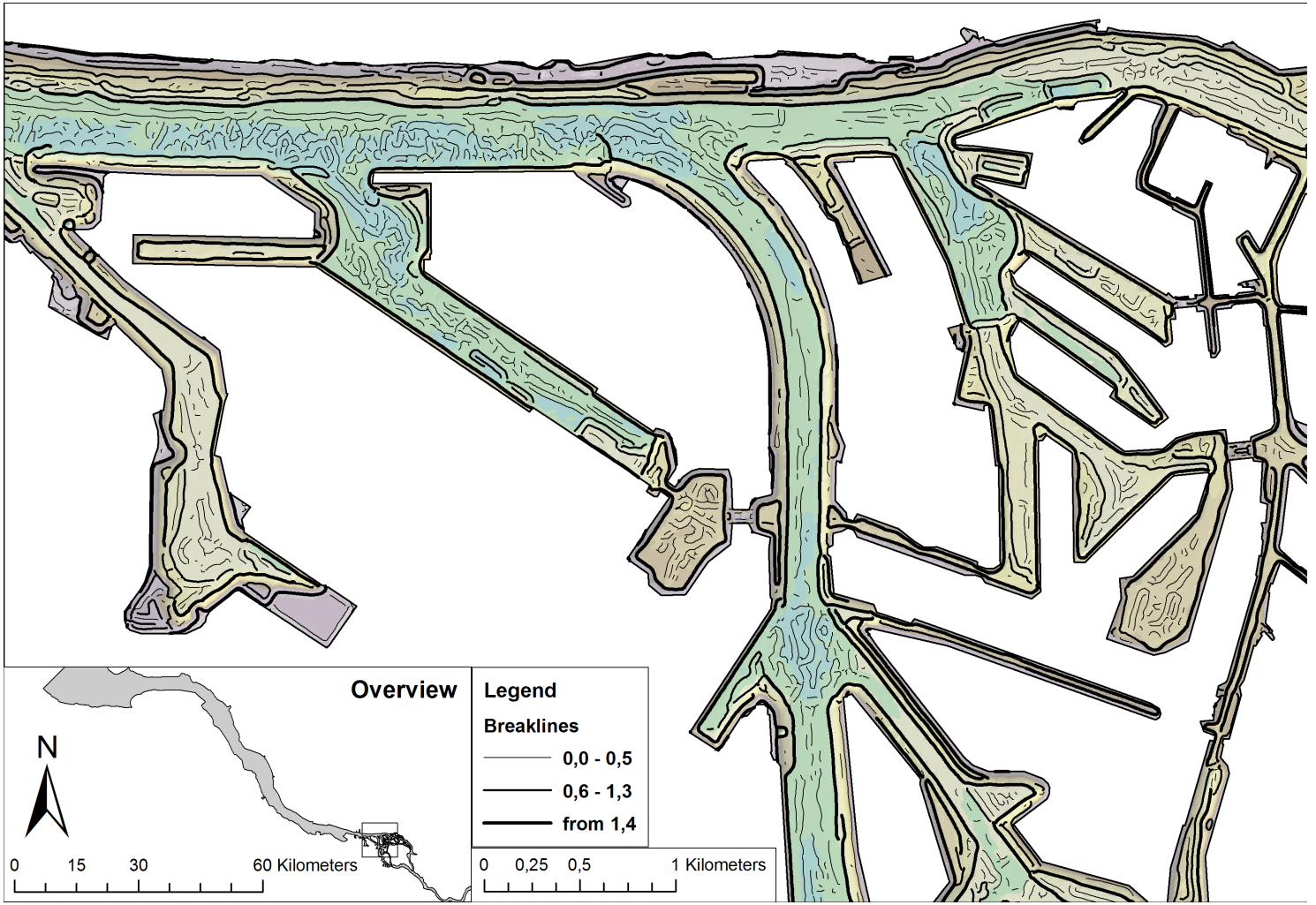



**Figure B.4.:** Breaklines for Elbe river detected in the DTM of Hamburg Metropolitan Region.

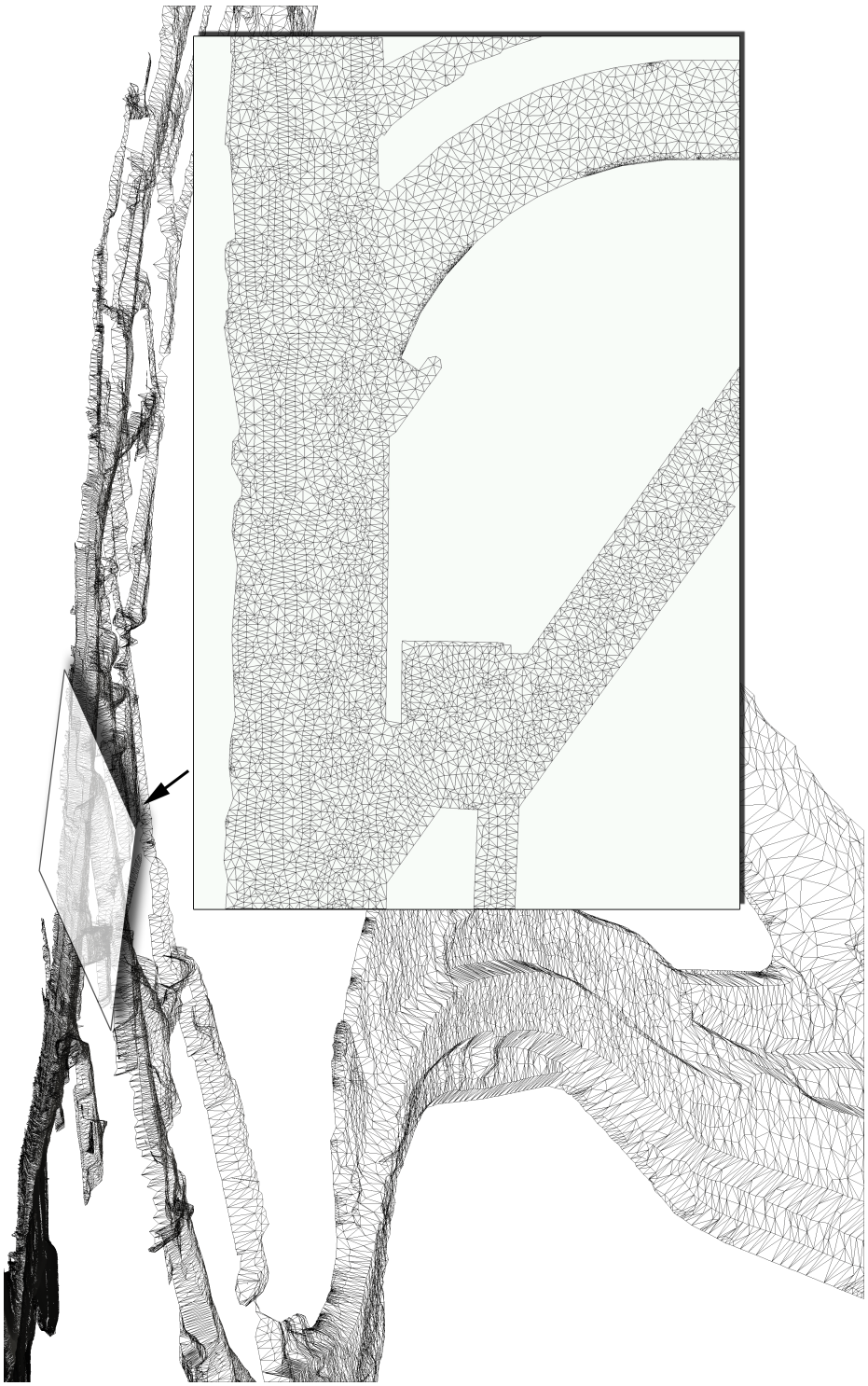


**Figure B.5:** Mesh of the Elbe river near Hamburg created using the flow model discretization service.

# C. Performance Measurements of RMA·Kalypso

Performance measurements were done for an implementation of the first level of the parallelization approach in RMA·Kalypso described in Section 7.3. The extension of this legacy code towards the MPI-parallel sparse solver library PETSc took about 1 month, which shows that limited parallel computing capabilities can be added to such a software with a relatively small effort. Nevertheless, this cannot truly be called a "gridification" due to the reasons explained in Chapter 2.

The mesh of the Elbe river created in this thesis has not yet been calibrated as calibration is a difficult and time-consuming manual process. Instead, an existing, calibrated model was used. Its mesh has about 390.000 triangles and simulates 10 hours of the 2007 storm surge event. Time is discretized in 5 minute intervals (120 time steps). Each time step, on average, requires 3-4 iterations, i. e. 3-4 linear systems have to be solved.

The execution environment for the performance measurement was a compute cluster at Hamburg University of Technology. All tests were executed three times and used the same type of computing resources with 24-96 GB of main memory and two 2.8 GHz quad-core CPUs. The test suite consisted of separate runs for the additive Schwarz preconditioned, iterative method using a direct solver on each subdomain (ASM / MUMPS LU) and the direct solution of the global system (MUMPS LU). The number of computing resources was varied in the range of 1 to 16. As per PETSc recommendations, only one MPI process per computing node was started so that memory bandwidth could not become a bottleneck. Separate tests — not shown here — confirmed that using two processes per node gives no further speedup. More than two processes per node even resulted in a degraded performance.

A major improvement that all tests demonstrated was the considerable reduction of the overall time to solution with an increasing number of resources for constant model size (also called weak scaling, see Figure C.1). The ASM method resulted in a slight advantage — a 10% reduction in time, at the most — over the direct solution. As the graphics show, using more than 16 resources is not likely to be efficient for this model size, as the parallel efficiency drops to below 33% for $\geq 16$ parallel processes.

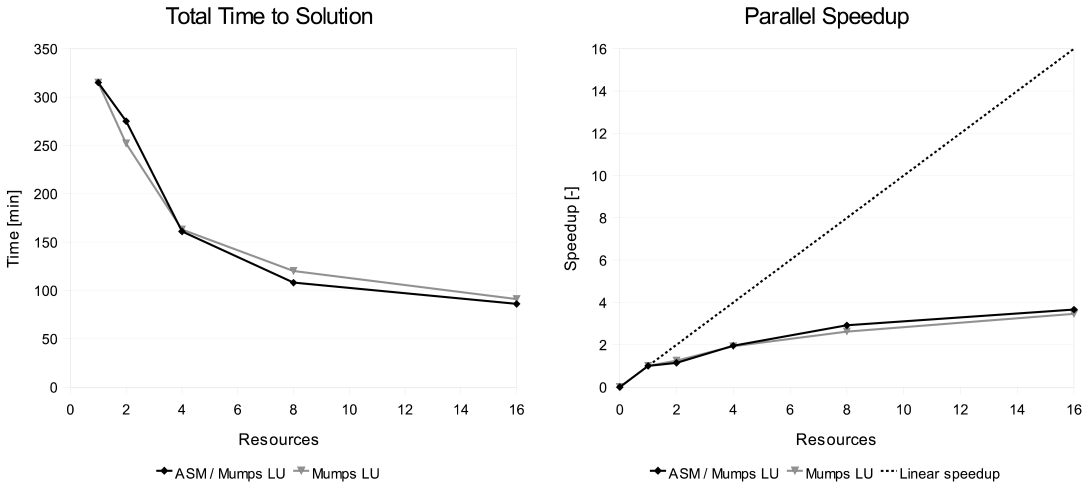



**Figure C.1.:** Performance comparison of the parallel execution of RMA·Kalypso using 1, 2, 4, 8, and 16 computing resources for the two solution methods presented in Section 7.3.

# Bibliography


[AA+05] M. Aktas, G. Aydin, A. Donnellan, R. Granat, G. Lyzenga, D. McLeod, S. Pallickara, J. Parker, M. Pierce, and A. Sayar. "Implementing Geographical Information System Grid Services to Support Computational Geophysics in a Service-Oriented Environment". In: *Proceedings of the Earth-Sun System Technology Conference (ESTC 2005)*. College Park, Maryland, USA, June 28-30. NASA (cit. on p. 66).

[AA+06] M. Aktas, G. Aydin, A. Donnellan, G. C. Fox, R. Granat, L. Grant, G. Lyzenga, D. McLeod, S. Pallickara, J. Parker, M. Pierce, J. Rundle, A. Sayar, and T. Tullis. "iSERVO: Implementing the International Solid Earth Research Virtual Observatory by Integrating Computational Grid and Geographical Information Web Services". In: *Pure and Applied Geophysics* 163 (11 2006), pp. 2281–2296 (cit. on p. 66).

[AB+05] A. Anjomshoaa, F. Brisard, M. Drescher, D. Fellows, A. Ly, S. McGough, and D. Pulsipher. *Job Submission Description Language (JSDL) Specification, Version 1.0. GFD-R.056 (obsoleted by GFD-R.136)*. Ed. by A. Savva. Global Grid Forum (GGF), 2005 (cit. on p. 69).

[AB+08] A. Anjomshoaa, F. Brisard, A. Ly, and D. Pulsipher. *Job Submission Description Language (JSDL) Specification, Version 1.0. GFD-R.136*. Open Grid Forum (OGF), 2008 (cit. on p. 69).

[ACE06] G. Aloisio, M. Caffaro, and I. Epicoco. "A Grid Software Process". In: *Grid Computing. Software Environments and Tools*. Ed. by J. C. Cunha and O. F. Rana. New York: Springer, 2006, pp. 75–98 (cit. on p. 14).

[AD+01] P. R. Amestoy, I. R. Duff, J.-Y. L'Excellent, and J. Koster. "A Fully Asynchronous Multifrontal Solver Using Distributed Dynamic Scheduling". In: *SIAM Journal on Matrix Analysis and Applications* 23 (1 2001), pp. 15–41 (cit. on p. 112).

[AG11] L. D. Angelo and L. Giaccari. "An Efficient Algorithm for the Nearest Neighbourhood Search for Point Clouds". In: *International Journal of Computer Science Issues* 8 (5 2011) (cit. on p. 88).

[AR96] M. B. Abbott and J. C. Refsgaard, eds. *Distributed Hydrological Modelling*. Vol. 22. Water Science and Technology Library. Dordrecht and Boston: Kluwer Academic, 1996 (cit. on p. 32).

[Bab57] I. Babuška. "Über Schwarzsche Algorithmen in partiellen Differentialgleichungen der mathematischen Physik". In: *ZAMM - Zeitschrift für Angewandte Mathematik und Mechanik* 37 (7-8 1957), pp. 243–245 (cit. on p. 112).

[Bar07] J. I. Barredo. "Major Flood Disasters in Europe: 1950–2005". In: *Natural Hazards* 42 (1 2007), pp. 125–148 (cit. on p. 1).

[Bar08a] B. Baranski. "52°North WPS-G, A Grid-enabled OGC Web Processing Service (WPS)". Presentation Slides. In: *OGC-OGF Collaboration Workshop at The Open Grid Forum (OGF-22)*. February 25-28, Cambridge, MA, USA. URL: `http://www.ogf.org/OGF22/materials/1075/bbaranski.WPS-G.pdf` (visited on May 2, 2012) (cit. on p. 67).

[Bar08b] B. Baranski. "Grid Computing Enabled Web Processing Service". In: *Proceedings of the 6th Geographic Information Days (GI-Days 2008)*. June 16-18, Münster, Germany, pp. 243–256 (cit. on p. 67).

[Bau10] P. Baumann. *OGC Web Coverage Service (WCS) 2.0 Interface Standard – Core. OGC 09-110r3*. Ed. by P. Baumann. Open Geospatial Consortium, Inc. (OGC), 2010 (cit. on pp. 45, 172).

[BB+11] S. Balay, J. Brown, K. Buschelman, V. Eijkhout, W. Gropp, D. Kaushik, M. G. Knepley, L. McInnes, B. F. Smith, and H. Zhang. *PETSc Users Manual Revision 3.2*. Mathematics and Computer Science Division, Argonne National Laboratory, 2011 (cit. on pp. 111, 113).

[Ber87] F. Bergholm. "Edge Focusing". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-9 (6 1987), pp. 726–741 (cit. on p. 52).

[BFH03] F. Berman, G. C. Fox, and A. J. G. Hey, eds. *Grid Computing – Making the Global Infrastructure a Reality*. Wiley Series in Communications Networking & Distributed Systems. Chichester: Wiley, 2003.

[BG+06] M. Babík, E. Gatial, O. Habala, L. Hluchý, M. Laclavíc, and M.Ališka. "Semantic Grid Services in K-Wf Grid". In: *Second International Conference on Semantics, Knowledge and Grid (SKG 2006)*. Guilin, Guangxi, China, November 1-3, p. 66 (cit. on p. 105).

[BG+97] S. Balay, W. Gropp, L. C. McInnes, and B. F. Smith. "Efficient Management of Parallelism in Object Oriented Numerical Software Libraries". In: *Modern Software Tools in Scientific Computing*, pp. 163–202 (cit. on p. 111).

[BH+07] M. Babík, O. Habala, L. Hluchý, and M. Laclavíc. "Semantic Services Grid in Flood-Forecasting Simulations". In: *Computing and Informatics* 26 (4 2007), pp. 447–464 (cit. on p. 105).

[BH+08] A. Brzank, C. Heipke, J. Goepfert, and U. Soergel. "Aspects of Generating Precise Digital Terrain Models in the Wadden Sea from Lidar – Water Classification and Structure Line Extraction". Remote Sensing of the Coastal Ecosystems. In: *ISPRS Journal of Photogrammetry and Remote Sensing* 63 (5 2008), pp. 510–528 (cit. on p. 52).

[BMH03] P. D. Bates, K. J. Marks, and M. S. Horritt. "Optimal Use of High-Resolution Topographic Data in Flood Inundation Models". In: *Hydrological Processes* 17 (3 2003), pp. 537–557 (cit. on p. 52).

[BRL07] J. I. Barredo, A. d. Roo, and C. Lavalle. "Flood Risk Mapping at European Scale". In: *Water Science & Technology* 56 (4 2007), pp. 11–17 (cit. on p. 1).

[Bro05] A. Bronstert, ed. *Abflussbildung. Prozessbeschreibung und Fallbeispiele*. Vol. 13. Forum für Hydrologie. Hennef: Deutsche Vereinigung für Wasserwirtschaft, Abwasser und Abfall e.V. (DWA), 2005 (cit. on p. 32).

[BS+09] B. Baranski, A. Shaon, A. Woolf, and S. Kurzbach. *OWS-6 WPS Grid Processing Profile Engineering Report. OGC 09-041r1*. Open Geospatial Consortium, Inc. (OGC), 2009 (cit. on p. 4).

[BSE12] A. Bröring, C. Stasch, and J. Echterhoff. *OGC Sensor Observation Service Interface Standard, Version 2.0. OGC 12-006*. Open Geospatial Consortium, Inc. (OGC), 2012 (cit. on pp. 45, 171).

[BSR11] B. Baranski, B. Schäffer, and R. Redweik. "Geoprocessing in the Clouds". In: *OSGeo Journal* (8 2011), pp. 17–22 (cit. on p. 64).

[BU06] M. Bubak and S. Unger, eds. *K-Wf Grid – The Knowledge-Based Workflow System for Grid Applications (CGW 2006)*. Proceedings of the Cracow '06 Grid Workshop. Cracow, Poland, October 15-18 (cit. on p. 105).

[Bun10] Bund/Länder-Arbeitsgemeinschaft Wasser. *Empfehlungen zur Aufstellung von Hochwassergefahrenkarten und Hochwasserrisikokarten. Beschlossen auf der 139. LAWA-VV am 25./26. März 2010 in Dresden*. Ed. by Sächsisches Staatsministerium für Umwelt und Landwirtschaft. Dresden: Ständiger Ausschuss der LAWA "Hochwasserschutz und Hydrologie (AH)", 2010 (cit. on p. 30).

[Can86] J. F. Canny. "A Computational Approach To Edge Detection". In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 8 (6 1986), pp. 679–698 (cit. on pp. 52, 53).

[Cas99] V. Casulli. "A Semi-Implicit Finite Difference Method for Non-Hydrostatic, Free-Surface Flows". In: *International Journal for Numerical Methods in Fluids* 30 (4 1999), pp. 425–440 (cit. on p. 109).

[Cat09] S. J. Caton. "On-Demand Distributed Image Processing Over An Adaptive Campus-Grid". School of Computer Science. Dissertation. Cardiff University, 2009 (cit. on p. 11).

[CC+06] D. Caromel, V. Cavé, A. Di Costanzo, C. Brignolles, B. Grawitz, and Y. Viala. "Executing Hydrodynamic Simulation on Desktop Grid with ObjectWeb Proactive". Nice, France, September 4-8. In: *Proceedings of the 7th International Conference on Hydroinformatics (HIC 2006)*. Nice, France, September 4-8 (cit. on p. 16).

[CC+10] N. Chrisochoides, A. Chernikov, A. Fedorov, A. Kot, L. Linardakis, and P. Foteinos. "Towards Exascale Parallel Delaunay Mesh Generation". In: *Proceedings of the 18th International Meshing Roundtable (IMR 2009)*. October 25-28, Salt Lake City, UT, USA, pp. 319–336 (cit. on p. 49).

[CF+01] K. Czajkowski, S. Fitzgerald, I. Foster, and C. Kesselman. "Grid Information Services for Distributed Resource Sharing". In: *Proceedings of the 10th IEEE International Symposium on High-Performance Distributed Computing (HPDC-10)*. San Francisco, CA, USA, August 7-9, pp. 181–194 (cit. on p. 63).

[CH08] I. Cooper and Y. Huang. "The Design and Evaluation of MPI-Style Web Services". In: *Computational Science – ICCS 2008*. 8th International Conference, Kraków, Poland, June 23-25, Proceedings, Part I. Ed. by M. Bubak, G. D. v. Albada, J. J. Dongarra, and P. M. A. Sloot. Vol. 5101. 3 vols. Lecture Notes in Computer Science 5101. Berlin and Heidelberg: Springer, 2008, pp. 184–193 (cit. on p. 18).

[Cha04] H. Chanson. *Environmental Hydraulics for Open Channel Flows*. Oxford: Elsevier Science & Technology, 2004 (cit. on p. 35).

[CM+03] D. M. Cobby, D. C. Mason, M. S. Horritt, and P. D. Bates. "Two-Dimensional Hydraulic Flood Modelling Using a Finite-Element Mesh Decomposed According to Vegetation and Topographic Features Derived from Airborne Scanning Laser Altimetry". In: *Hydrological Processes* 17 (10 2003), pp. 1979–2000 (cit. on p. 52).

[Cog08] D. W. Coggin. "LiDAR in Coastal Storm Surge Modeling: Modeling Linear Raised Features". Department of Civil, Environmental and Construction Engineering. Master Thesis. University of Central Florida, 2008 (cit. on p. 52).

[CW00] V. Casulli and R. A. Walters. "An Unstructured Grid, Three-Dimensional Model Based on the Shallow Water Equations". In: *International Journal for Numerical Methods in Fluids* 32 (3 2000), pp. 331–348 (cit. on p. 109).

[CZ02] V. Casulli and P. Zanolli. "Semi-Implicit Numerical Modeling of Nonhydrostatic Free-Surface Flows for Environmental Problems". In: *Mathematical and Computer Modelling* 36 (9-10 2002), pp. 1131–1149 (cit. on p. 109).

[CZ05] V. Casulli and P. Zanolli. "High Resolution Methods for Multidimensional Advection-Diffusion Problems in Free-Surface Hydrodynamics". In: *Ocean Modelling* 10 (1-2 2005), pp. 137–151 (cit. on p. 109).

[DC+03] L. Di, A. Chen, W. Yang, and P. Zhao. "The Integration of Grid Technology with OGC Web Services (OWS). NWGISS for NASA EOS Data". In: *Proceedings of the 12th IEEE International Symposium on High-Performance Distributed Computing and the 8th Global Grid Forum (HPDC-12 & GGF-8)*. Seattle, WA, USA, June 22-24, pp. 24–27 (cit. on p. 62).

[DC+08] L. Di, A. Chen, W. Yang, Y. Liu, Y. Wei, P. Mehrotra, C. Hu, and D. Williams. "The Development of a Geospatial Data Grid by Integrating OGC Web Services with Globus-Based Grid Technology". In: *Concurrency and Computation: Practice & Experience* 20 (14 2008), pp. 1617–1635 (cit. on p. 63).

[Der87] R. Deriche. "Using Canny's Criteria to Derive a Recursively Implemented Optimal Edge Detector". In: *International Journal of Computer Vision* 1 (2 1987), pp. 167–187 (cit. on p. 52).

[DES82] R. S. Dembo, S. C. Eisenstat, and T. Steihaug. "Inexact Newton Methods". In: *SIAM Journal on Numerical Analysis* 19 (2 1982), pp. 400–408 (cit. on p. 113).

[DGG08] L. Díaz, C. Granell, and M. Gould. "Case Study: Geospatial Processing Services for Web-Based Hydrological Applications". In: *Geospatial Services and Applications for the Internet*. Ed. by J. T. Sample, K. Shaw, S. Tu, and M. Abdelguerfi. New York: Springer, 2008, pp. 31–47 (cit. on p. 64).

[DH+07] B. Dillaway, M. Humphrey, C. Smith, M. Theimer, and G. Wasson, eds. *HPC Basic Profile, Version 1.0. OGF GFD-R-P.114*. Open Grid Forum (cit. on p. 69).

[Din08] S. L. Dingman. *Fluvial Hydraulics*. New York: Oxford University, 2008 (cit. on p. 35).

[DK04] S. Dong and G. E. Karniadakes. "Multilevel Parallelization Models in CFD". In: *Journal of Aerospace Computing, Information, and Communication* 1 (6 2004), pp. 256–268 (cit. on p. 17).

[DKK05] S. Dong, G. E. Karniadakes, and N. T. Karonis. "Cross-Site Computations on the TeraGrid". In: *Computing in Science & Engineering* 7 (5 2005), pp. 14–23 (cit. on p. 16).

[Dor09] T. Dorka. *Geographische Berechnungsdienste im Grid mit dem Web-Services-Resource-Framework*. Diplomarbeit. 2009 (cit. on pp. 72, 96, 99).

[DSF08] T. Dörnemann, M. Smith, and B. Freisleben. "Composition and Execution of Secure Workflows in WSRF-Grids". In: *8th IEEE International Symposium on Cluster Computing and the Grid (CCGRID 2008)*. 19-22 May, Lyon, France, pp. 122–129 (cit. on p. 20).

[DW87] M. Dryja and O. B. Widlund. *An Additive Variant of the Schwarz Alternating Method for the Case of Many Subregions*. Ultracomputer Note #131. New York and USA: Courant Institute of Mathematical Sciences, New York University, 1987 (cit. on p. 112).

[Euro7a] European Parliament. *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 Directive. Version 25.04.2007. 2007 (cit. on pp. 2, 90).

[Euro7b] European Parliament. *Directive 2007/60/EC of the European Parliament and of the Council of 23 October 2007 on the assessment and management of flood risks*. Floods Directive. Version 06.11.2007. 2007 (cit. on pp. 1, 27, 29, 30).

[Exc07] The European Exchange Circle on Flood Mapping (EXCIMAP). *Handbook on Good Practices for Flood Mapping in Europe*. Endorsed by Water Directors, 29-30 November 2007. 2007 (cit. on p. 30).

[FB+10] T. Foerster, B. Baranski, B. Schäffer, and K. Lange. "Geoprocessing in Hybrid Clouds". In: *Geoinformatik 2010. Die Welt im Netz*. March 17-19, Kiel, Germany. Ed. by A. Zipf, K. Behncke, F. Hillen, and J. Schaefermeyer. Heidelberg: Akademische Verlagsgesellschaft AKA, 2010, pp. 13–19 (cit. on p. 65).

[FC04] E. Floros and Y. Cotronis. "Exposing MPI Applications as Grid Services". In: *Euro-Par 2004 Parallel Processing*. 10th International Euro-Par Conference, Pisa, Italy, August/September 2004, Proceedings. Ed. by M. Danelutto, D. Laforenza, and M. Vanneschi. Vol. 3149. Lecture Notes in Computer Science 3149. Berlin and Heidelberg: Springer, 2004, pp. 436–443 (cit. on p. 106).

[FC+05] I. Foster, K. Czajkowski, D. E. Ferguson, J. Frey, S. Graham, T. Maguire, D. Snelling, and S. Tuecke. "Modeling and Managing State in Distributed Systems: The Role of OGSI and WSRF". In: *Proceedings of the IEEE* 93 (3 2005), pp. 604–612 (cit. on p. 23).

[FC06] E. Floros and Y. Cotronis. "ServOSims: A Service Oriented Framework for Composing and Executing Multidisciplinary Simulations". In: *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing (e-Science 2006)*. Amsterdam, Netherlands, 4-6 December, p. 66 (cit. on pp. 106, 125).

[FG+07] I. Foster, A. Grimshaw, P. Lane, W. Lee, M. Morgan, S. Newhouse, S. Pickles, D. Pulsipher, C. Smith, and M. Theimer, eds. *OGSA Basic Execution Service Version 1.0. GFD-R.108*. Open Grid Forum (OGF) (cit. on p. 69).

[FH+07] T. Foerster, C. Heier, S. Keens, C. Kiehle, R. O'Neil, N. Ostländer, J. M. Pau, P. Schut, and A. Whiteside. *OGC Web Processing Service (WPS) Specification, Version 1.0. OGC 05-007r7*. Ed. by P. Schut and A. Whiteside. Open Geospatial Consortium, Inc. (OGC), 2007 (cit. on pp. 45, 172).

[FH+09] T. Foerster, C. Heier, S. Keens, C. Kiehle, R. O'Neil, N. Ostländer, J. M. Pau, P. Schut, and A. Whiteside. *Corrigendum for OpenGIS Implementation Standard Web Processing Service (WPS) 1.0.0, Version 0.0.8. OGC 08-091r6*. Ed. by P. Schut. Open Geospatial Consortium, Inc. (OGC), 2009 (cit. on p. 46).

[Fie00] R. T. Fielding. "Architectural Styles and the Design of Network-Based Software Architectures". Department of Computer Science. Dissertation. University of California, 2000 (cit. on p. 43).

[FK+02] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. "Grid Services for Distributed System Integration". In: *Computer* 35 (6 2002), pp. 37–46 (cit. on pp. 19, 61).

[FK+03] I. Foster, C. Kesselman, J. M. Nick, and S. Tuecke. "The Physiology of the Grid". In: *Grid Computing – Making the Global Infrastructure a Reality*. Ed. by F. Berman, G. C. Fox, and A. J. G. Hey. Wiley Series in Communications Networking & Distributed Systems. Chichester: Wiley, 2003, pp. 217–250 (cit. on pp. 19, 21).

[FK97] I. Foster and C. Kesselman. "Globus: A Metacomputing Infrastructure Toolkit". In: *International Journal of High Performance Computing Applications* 11 (2 1997), p. 115 (cit. on p. 170).

[FK+98] I. Foster, C. Kesselman, G. Tsudik, and S. Tuecke. "A Security Architecture for Computational Grids". In: *Proceedings of the 5th Conference on Computer and Communications Security (5CCS 1998)*. San Francisco, CA, USA, November 2-5, pp. 83–92 (cit. on p. 24).

[FK99a] I. Foster and C. Kesselman. "Computational Grids". In: *The Grid: Blueprint for a New Computing Infrastructure*. Ed. by I. Foster and C. Kesselman. Morgan Kaufmann, 1999, pp. 15–51 (cit. on p. 8).

[FK99b] I. Foster and C. Kesselman. "The Globus Toolkit". In: *The Grid: Blueprint for a New Computing Infrastructure*. Ed. by I. Foster and C. Kesselman. Morgan Kaufmann, 1999, pp. 259–278 (cit. on p. 170).

[FK99c] I. Foster and C. Kesselman, eds. *The Grid: Blueprint for a New Computing Infrastructure*. Morgan Kaufmann, 1999.

[FKT03] I. Foster, C. Kesselman, and S. Tuecke. "The Anatomy of the Grid. Enabling Scalable Virtual Organizations". In: *Grid Computing – Making the Global Infrastructure a Reality*. Ed. by F. Berman, G. C. Fox, and A. J. G. Hey. Wiley Series in Communications Networking & Distributed Systems. Chichester: Wiley, 2003, pp. 171–198 (cit. on pp. 10, 13).

[Fle02] G. Fleming. *Flood Risk Management. Learning to Live with Rivers*. London: Thomas Telford, 2002 (cit. on p. 34).

[FM08] T. Fleuren and P. Müller. "BPEL Workflows Combining Standard OGC Web Services and Grid-enabled OGC Web Services". In: *Proceedings of the 34th EUROMICRO Conference on Software Engineering and Advanced Applications (SEAA 2008)*. Parma, Italy, September 3-5, pp. 337–344 (cit. on p. 20).

[FMS06] I. Foster, T. Maguire, and D. Snelling. *OGSA WSRF Basic Profile 1.0. GFD-R-P.072*. Open Grid Forum (OGF), 2006 (cit. on p. 69).

[FO+07] A. Friis-Christensen, N. Ostländer, M. Lutz, and L. Bernard. "Designing Service Architectures for Distributed Geoprocessing: Challenges and Future Directions". In: *Transactions in GIS* 11 (6 2007), pp. 799–818 (cit. on pp. 45, 64, 65, 71).

[Fos02] I. Foster. "What is the Grid? – A Three Point Checklist". In: *GRIDtoday* 1 (6 2002) (cit. on p. 9).

[Fos05] I. Foster. "Globus Toolkit Version 4: Software for Service-oriented Systems". In: *Network and Parallel Computing. IFIP International Conference, NPC 2005, Beijing, China, November/December 2005, Proceedings*. Ed. by H. Jin, D. Reed, and W. Jiang. Vol. 3779. Lecture Notes in Computer Science 3779. Berlin and Heidelberg: Springer, 2005, pp. 2–13 (cit. on p. 23).

[FPC08] A. Friis-Christensen, H. Pundt, and I. Compte, eds. *Proceedings of the 11th AGILE International Conference on Geographic Information Science. Taking Geoinformation Science One Step Further (AGILE 2008)*. May 5-8, Girona, Spain.

[FS07] T. Foerster and B. Schäffer. "A Client for Distributed Geo-Processing on the Web". In: *Proceedings of the 7th International Symposium on Web and Wireless Geographical Information Systems (W2GIS 2007)*. November 28-29, Cardiff, UK, pp. 252–263 (cit. on p. 47).

[FS+11] T. Foerster, B. Schäffer, B. Baranski, and J. Brauner. "Geospatial Web Services for Distributed Processing – Applications and Scenarios". In: *Geospatial Web Services: Advances in Information Interoperability*. Ed. by P. Zhao and L. Di. IGI Global, 2011, pp. 245–286 (cit. on p. 67).

[FT05] I. Foster and S. Tuecke. "Describing the Elephant: The Different Faces of IT as Service". In: *Queue* 3 (6 2005), pp. 26–34 (cit. on p. 7).

[FW11] S. Freitag and P. Wieder. "The German Grid Initiative D-Grid: Current State and Future Perspectives". In: *Guide to E-Science: Next Generation Scientific Research and Discovery*. Ed. by X. Yang, L. Wang, and W. Jie. New York: Springer, 2011, pp. 29–52 (cit. on p. 24).

[FZ+08] I. Foster, Y. Zhao, I. Raicu, and S. Lu. "Cloud Computing and Grid Computing 360-Degree Compared". In: *Grid Computing Environments Workshop (GCE 2008)*. 12-16 November, Austin, TX, USA (cit. on p. 23).

[Gau67] P. Gauckler. "Etudes Théoriques et Pratiques sur l'Ecoulement et le Mouvement des Eaux". In: *Comptes Rendues de l'Académie des Sciences* (64 1867), pp. 818–822 (cit. on p. 55).

[GD+12] C. Granell, L. Díaz, A. Tamayo, and J. Huerta. "Assessment of OGC Web Processing Services for REST Principles". In: *International Journal of Data Mining, Modelling and Management* (Accepted for Publication 2012). URL: `http://arxiv.org/abs/1202.0723v2` (visited on May 15, 2012) (cit. on p. 43).

[Gel99] P. H. A. J. M. v. Gelder. "Risks and Safety of Flood Protection Structures in the Netherlands". In: *Proceedings of the Participation of Young Scientists in the FORUM ENGELBERG 1999 on Risk and Safety of Technical Systems – in View of Profound Changes. PSI-PROCEEDINGS 99-03*. Engelberg, Switzerland, March 23-26. Paul Scherrer Institut, pp. 55–60 (cit. on p. 29).

[GGW07] J. B. Gregersen, P. J. Gijsbers, and S. J. Westen. "OpenMI: Open Modelling Interface". In: *Journal of Hydroinformatics* 9 (3 2007), pp. 175–191 (cit. on p. 126).

[Ghi05] D. R. Ghimire. "Design of a Grid-based Geo-service Architecture". International Institute for Geo-Information Science and Earth Observation. Master Thesis. University of Twente, 2005 (cit. on p. 64).

[Glo05] The Globus Security Team. *Globus Toolkit Version 4 Grid Security Infrastructure: A Standards Perspective*. Argonne National Laboratory, 2005 (cit. on p. 25).

[GM+09] A. Grimshaw, M. Morgan, D. Merrill, H. Kishimoto, A. Savva, D. Snelling, C. Smith, and D. Berry. "An Open Grid Services Architecture Primer". In: *Computer* 42 (2 2009), pp. 27–34 (cit. on p. 69).

[GMR05] L. Giraud, A. Marrocco, and J.-C. Rioual. "Iterative Versus Direct Parallel Substructuring Methods in Semiconductor Device Modelling". In: *Numerical Linear Algebra with Applications* 12 (1 2005), pp. 33–53 (cit. on p. 119).

[GSW05] D. R. Ghimire, I. Simonis, and A. Wytzisk. "Integration of Grid Approaches Into the Geographic Web Service Domain". In: *From Pharaohs to Geoinformatics – The Role of SDIs in an Information Society. Proceedings of the FIG Working Week 2005 and GSDI-8*. 16-21 April, Cairo, Egypt. Global Spatial Data Infrastructure Association (cit. on p. 64).

[GW02] D. Gesch and R. Wilson. "Development of a Seamless Multisource Topographic/Bathymetric Elevation Model of Tampa Bay". In: *Marine Technology Society Journal* 35 (4 2002), pp. 58–64 (cit. on p. 51).

[GZ10] M. Griebel and P. Zaspel. "A Multi-GPU Accelerated Solver for the Three-Dimensional Two-Phase Incompressible Navier-Stokes Equations". In: *Computer Science-Research and Development* 25 (1 2010), pp. 65–73 (cit. on pp. 16, 17).

[HA+04] L. Hluchý, J. Astaloš, M. Dobrucký, O. Habala, B. Šimo, and V. Tran. "Flood Forecasting in a Grid Computing Environment". In: *Parallel Processing and Applied Mathematics*. 5th International Conference, PPAM 2003, Częstochowa, Poland, September 7-10, 2003, Revised Papers. Ed. by R. Wyrzykowski, J. J. Dongarra, M. Paprzycki, and J. Waśniewski. Vol. 3019. Lecture Notes in Computer Science 3019. Berlin and Heidelberg: Springer, 2004, pp. 831–839 (cit. on p. 104).

[Has10] W. Hasselbring, ed. *Betriebliche Informationssysteme: Grid-basierte Integration und Orchestrierung*. Berlin: GITO mbH Verlag, 2010 (cit. on pp. 21, 65).

[Hei11] T. Heister. "A Massively Parallel Finite Element Framework with Application to Incompressible Flows". Mathematisch-naturwissenschaftliche Fakultäten. Dissertation. Georg-August-Universität Göttingen, 2011 (cit. on p. 112).

[HF+10] G. Hobona, D. Fairbairn, H. Hide, and P. James. "Orchestration of Grid-enabled Geospatial Web Services in Geoscientific Workflows". In: *IEEE Transactions on Automation Science and Engineering* 7 (2 2010), pp. 407–411 (cit. on p. 65).

[HFJ07] G. Hobona, D. Fairbairn, and P. James. "Workflow Enactment of Grid-enabled Geospatial Web Services". In: *Proceedings of the 2007 UK e-Science All Hands Meeting* (cit. on pp. 65, 67).

[HH+05] L. Hluchý, O. Habala, V. Tran, E. Gatial, M. Mališka, B. Šimo, and P. Slížik. "Collaborative Environment for Grid-Based Flood Prediction". In: *Computing and Informatics* 24 (1 2005), pp. 87–108 (cit. on p. 105).

[HH+06] L. Hluchý, O. Habala, M. Mališka, B. Šimo, V. Tran, J. Astaloš, and M. Babík. "Grid-Based Flood Prediction Virtual Organization". In: *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing (e-Science 2006)*. Amsterdam, Netherlands, 4-6 December, p. 4 (cit. on p. 105).

[HK05] C. Heier and C. Kiehle. "Geodatenverarbeitung im Internet – der OGC Web Processing Service". In: *GIS* 6 (2005), pp. 39–43 (cit. on p. 46).

[HMH06] O. Habala, M. Mališka, and L. Hluchý. "Service-Based Flood Forecasting Simulation Cascade in K-Wf Grid". In: *K-Wf Grid – The Knowledge-Based Workflow System for Grid Applications (CGW 2006)*. Proceedings of the Cracow 2006 Grid Workshop. Cracow, Poland, October 15-18, pp. 138–145 (cit. on p. 105).

[Hoe01] J. Hoeflinger. "Producing Scalable Performance with OpenMP: Experiments with Two CFD Applications". In: *Parallel Computing* 27 (4 2001), pp. 391–413 (cit. on p. 16).

[Hor00] M. S. Horritt. "Development of Physically Based Meshes for Two-Dimensional Models of Meandering Channel Flow". In: *International Journal for Numerical Methods in Engineering* 47 (12 2000), pp. 2019–2037 (cit. on p. 52).

[HOS08] J. Horak, A. Orlik, and J. Stromsky. "Web Services for Distributed and Interoperable Hydro-Information Systems". In: *Hydrology and Earth System Sciences* 12 (2 2008), pp. 635–644 (cit. on pp. 65, 126).

[HS+07] M. Humphrey, C. Smith, M. Theimer, and G. Wasson, eds. *JSDL HPC Profile Application Extension, Version 1.0. OGF GFD-R.111*. Open Grid Forum (cit. on p. 69).

[HS52] M. R. Hestenes and E. Stiefel. "Methods of Conjugate Gradients for Solving Linear Systems". In: *Journal of Research of the National Bureau of Standards* 49 (6 1952), pp. 409–436 (cit. on p. 39).

[HT+02] L. Hluchý, V. Tran, J. Astaloš, M. Dobrucký, and G. T. Nguyen. "Parallel Flood Modeling Systems". In: *Computational Science – ICCS 2002*. International Conference Amsterdam, The Netherlands, April 21–24, 2002 Proceedings, Part I. Ed. by P. M. A. Sloot, A. G. Hoekstra, C. J. Kenneth Tan, and J. J. Dongarra. Vol. 2. Lecture Notes in Computer Science 2329. Berlin and Heidelberg: Springer, 2002, pp. 543–551 (cit. on p. 104).

[HT+03] L. Hluchý, V. Tran, D. Froehlich, and W. Castaings. "Methods and Experiences of Parallelizing Flood Models". In: *Recent Advances in Parallel Virtual Machine and Message Passing Interface* (2 2003), pp. 677–680 (cit. on p. 104).

[HT+04] L. Hluchý, V. Tran, O. Habala, B. Šimo, E. Gatial, J. Astaloš, and M. Dobrucký. "Flood Forecasting in CrossGrid Project". In: *Grid Computing*. Second European AcrossGrids Conference, AxGrids 2004, Nicosia, Cyprus, January 28-30, 2004. Revised Papers. Ed. by M. D. Dikaiakos. Vol. 3165. Lecture Notes in Computer Science 3165. Berlin and Heidelberg: Springer, 2004, pp. 51–60 (cit. on p. 104).

[Hug83] T. P. Hughes. *Networks of Power. Electrification in Western Society, 1880-1930*. Baltimore and London: Johns Hopkins University, 1983 (cit. on p. 8).

[Jan07] J. A. Jankowski. "Further Developments of UnTRIM: Parallel Implementation and its Verification". In: *Proceedings of the 5th IAHR International Symposium on Environmental Hydraulics (ISEH V)*. Tempe, AZ, USA, December 4-7 (cit. on p. 109).

[Jan09] J. A. Jankowski. "Parallel Implementation of a Non-Hydrostatic Model for Free Surface Flows with Semi-Lagrangian Advection Treatment". In: *International Journal for Numerical Methods in Fluids* 59 (10 2009), pp. 1157–1179 (cit. on p. 109).

[JC+10] S. Jha, M. Cole, D. S. Katz, M. Parashar, O. F. Rana, and J. Weissman. *Abstractions for Large-Scale Distributed Applications and Systems*. Unpublished. 2010. URL: `http://www.cct.lsu.edu/~sjha/dpa_publications/dpa_surveypaper.pdf` (visited on Apr. 12, 2012) (cit. on p. 13).

[JK12] S. Jha and D. S. Katz. *Abstractions for Distributed Applications and Systems: A Computational Science Perspective*. Vol. 79. Wiley Series on Parallel and Distributed Computing. Wiley, 2012 (to be published) (cit. on p. 13).

[Jul02] P. Y. Julien. *River Mechanics*. 1st ed. Cambridge: Cambridge University Press, 2002 (cit. on p. 35).

[KB+09] S. Kurzbach, S. Braune, C. Grimm, and E. Pasche. "Hochwasser-Modellierung im Geodateninfrastruktur-Grid". In: *Angewandte Geoinformatik 2009. Beiträge zum 21. AGIT-Symposium Salzburg*. Ed. by J. Strobl, T. Blaschke, and G. Griesebner. Wichmann, 2009, pp. 372–377 (cit. on p. 4).

[KB+10] S. Kurzbach, S. Braune, E. Pasche, and M. Smith. "Operative Hochwasservorhersage-Dienste im Geodateninfrastruktur-Grid". In: *Angewandte Geoinformatik 2010. Beiträge zum 22. AGIT-Symposium Salzburg*. Ed. by J. Strobl, T. Blaschke, and G. Griesebner. Wichmann, 2010, pp. 881–886 (cit. on p. 5).

[KC+08] V. Kravtsov, D. Carmeli, W. Dubitzky, A. Orda, A. Schuster, M. Silberstein, and B. Yoshpa. "Quasi-Opportunistic Supercomputing in Grid Environments". In: *Algorithms and Architectures for Parallel Processing. 8th International Conference, ICA3PP 2008*. Cyprus, June 9-11, 2008, Proceedings, pp. 233–244 (cit. on p. 11).

[Kee06] S. Keens. *OWS-4 WPS IPR: Discussions, Findings, and Use of WPS in OWS-4. 06-182r1*. Open Geospatial Consortium, Inc. (OGC), 2006 (cit. on p. 68).

[Kel06] P. Kellenberger, ed. *Proceedings of the Second IEEE International Conference on e-Science and Grid Computing (e-Science 2006)*. Amsterdam, Netherlands, 4-6 December. Washington DC and USA: Wiley.

[Kel87] C. T. Kelley. *Solving Nonlinear Equations with Newton's Method*. Vol. 1. Fundamentals of Algorithms. 1987 (cit. on p. 38).

[Key81] R. Keys. "Cubic Convolution Interpolation for Digital Image Processing". In: *IEEE Transactions on Acoustics, Speech and Signal Processing* 29 (6 1981), pp. 1153–1160 (cit. on p. 88).

[KGH06] C. Kiehle, K. Greve, and C. Heier. "Standardized Geoprocessing – Taking Spatial Data Infrastructures One Step Further". In: *Proceedings of the 9th AGILE International Conference on Geographic Information Science (AGILE 2006). Shaping the Future of Geographic Information Science in Europe*. Visegrád, Hungary, April 20-22 (cit. on p. 46).

[KGH07] C. Kiehle, K. Greve, and C. Heier. "Requirements for Next Generation Spatial Data Infrastructures – Standardized Web Based Geoprocessing and Web Service Orchestration". In: *Transactions in GIS* 11 (6 2007), pp. 819–834 (cit. on p. 46).

[KH08] M. Koutroumpas and C. Higgins. "A Pipeline Processing Approach To GIS". In: *Proceedings of the 11th AGILE International Conference on Geographic Information Science. Taking Geoinformation Science One Step Further. Girona, Spain, May 5-8*. May 5-8, Girona, Spain (cit. on p. 63).

[Kha09] D. B. Kharat. "Practical Aspects of Integrated 1D2D Flood Modelling of Urban Floodplains using LiDAR Topography Data". Doctoral Thesis. Heriot-Watt University, 2009 (cit. on p. 52).

[Kin88] I. P. King. *A Finite Element Model for Three Dimensional Flow*. RMA report. 1988 (cit. on p. 111).

[Kin93] I. P. King. *RMA-10 – A Finite Element Model for Three-Dimensional Density Stratified Flow*. 1993 (cit. on pp. 56, 111).

[KJ+04] R. Khatibi, D. Jackson, J. Curtin, C. Whitlow, and A. Verwey. "Vision Statement on Open Architecture for Hydraulic Modelling Software Tools". In: *Journal of Hydroinformatics* 6 (1 2004), pp. 57–74 (cit. on p. 126).

[KK08] A. Krüger and T. H. Kolbe. "Mapping Spatial Data Infrastructures to a Grid Environment for Optimized Processing of Large Amounts of Spatial Data". In: *Advances in Photogrammetry, Remote Sensing and Spatial Information Sciences. 2008 ISPRS Congress Book*. XXIst Congress of the International Society for Photogrammetry and Remote Sensing, Beijing, China, July 3-11. International Society for Photogrammetry and Remote Sensing Congress, pp. 1559–1566 (cit. on pp. 65, 66, 68, 69).

[KM+99] F. Keil, W. Mackens, H. Voß, and J. Werther, eds. *Scientific Computing in Chemical Engineering II: Simulation, Image Processing, Optimization, and Control*. Hamburg-Harburg, Germany, 26-28 May. Berlin and Heidelberg: Springer, 1999.

[KN+10] A. Kron, F. Nestmann, I. Schlüter, G. Schädler, C. Kottmeier, M. Helms, R. Mikovec, J. Ihringer, M. Musall, P. Oberle, U. Saucke, A. Bieberstein, J. Daňhelka, and J. Krejčí. "Operational Flood Management Under Large-Scale Extreme Conditions, Using the Example of the Middle Elbe". In: *Natural Hazards and Earth System Science* 10 (6 2010), pp. 1171–1181 (cit. on p. 28).

[KP+09] S. Kurzbach, E. Pasche, S. Lanig, and A. Zipf. "Benefits of Grid Computing for Flood Modeling in Service-Oriented Spatial Data Infrastructures". In: *GIS.Science* (3 2009), pp. 89–97 (cit. on p. 4).

[KP09] S. Kurzbach and E. Pasche. "A 3d Terrain Discretization Grid Service for Hydrodynamic Modeling". In: *Proceedings of the 8th International Conference on Hydroinformatics (HEIC 2009)*. Concepción, Chile, January 12-16 (cit. on pp. 4, 68).

[Kri51] D. G. Krige. "A Statistical Approach to Some Mine Valuations and Allied Problems at the Witwatersrand". Master Thesis. University of Witwatersrand, 1951 (cit. on p. 88).

[KS12] S. Kurzbach and N. Schrage. "Automatic Mesh Generation for 2D Hydrodynamic Flood Models from High-Resolution Digital Elevation Data". In: *Proceedings of the 10th Conference on Hydroinformatics (HIC 2012). Understanding Changing Climate and Environment and Finding Solutions*. July 14-18, Hamburg, Germany (cit. on p. 5).

[KTF03] N. T. Karonis, B. Toonen, and I. Foster. "MPICH-G2: A Grid-enabled Implementation of the Message Passing Interface". In: *Journal of Parallel and Distributed Computing* 63 (5 2003), pp. 551–563 (cit. on p. 15).

[LC+04] W.-J. Li, D. Chen, Y.-j. Li, Z.-w. Liang, and G.-f. Ji. "GIS Grid Services Load Balancing Based on Dynamic Resource Discovery". In: *International Symposium on Future Software Technology (ISFST 2004)*. Oktober 20-22, Xian, China. Ed. by K. Hao and Z. Jin. Xian and China, 2004 (cit. on p. 63).

[Lee08] C. Lee. *Grid Technologies for GEOSS. OGC TC Plenary Workshop on Perspectives on GEOSS Architecture: Principles and Implementation*. Presentation. December 3-4, Palacio de Congresos, Valencia, Spain. 2008 (cit. on p. 68).

[LK+09] S. Lanig, S. Kurzbach, E. Pasche, and A. Zipf. "Standards-Based Processing of Digital Elevation Models in Grid Computing Environments". In: *Workshop on Grid Technologies for Geospatial Applications at 12th AGILE International Conference on Geographic Information Science. Advances in GIScience*. June 2-5, Hannover, Germany (cit. on pp. 4, 68).

[LP08] C. Lee and G. Percivall. "Standards-Based Computing Capabilities for Distributed Geospatial Applications". In: *Computer* 41 (11 2008), pp. 50–57 (cit. on pp. 67, 71).

[LS+08a] S. Lanig, A. Schilling, B. Stollberg, and A. Zipf. "Erste Schritte auf dem Weg zur Verarbeitung digitaler Geländemodelle mittels Grid Computing". In: *Angewandte Geoinformatik 2008. Beiträge zum 20. AGIT-Symposium Salzburg*. Ed. by J. Strobl, T. Blaschke, and G. Griesebner. Wichmann, 2008 (cit. on p. 68).

[LS+08b] S. Lanig, A. Schilling, B. Stollberg, and A. Zipf. "Towards Standards-Based Processing of Digital Elevation Models for Grid Computing through Web Processing Service (WPS)". In: *International Conference on Computational Science and Its Applications (ICCSA 2008)*. Perugia, Italy, June/July 2008, pp. 191–203 (cit. on p. 68).

[LZ09] S. Lanig and A. Zipf. "Towards Generalization Processes of LiDAR Data Based on GRID and OGC Web Processing Services (WPS)". In: *Geoinformatik 2009*. Osnabrück, Germany, March 31 - April 2 (cit. on p. 68).

[MAA09] H. d. Moel, J. v. Alphen, and J. C. J. H. Aerts. "Flood Maps in Europe – Methods, Availability and Use". In: *Natural Hazards and Earth System Science* 9 (2 2009), pp. 289–301 (cit. on pp. 1, 28, 31).

[Mal01] A. Malcherek. *Hydromechanik der Fließgewässer*. Habilitation. Vol. 61. 2001 (cit. on p. 35).

[Mal10] A. Malcherek. *Gezeiten und Wellen. Die Hydromechanik der Küstengewässer*. 1st ed. Wiesbaden: Vieweg + Teubner, 2010. 301 pp. (cit. on p. 35).

[Man91] R. Manning. "On the Flow of Water in Open Channels and Pipes". In: *Transactions of the Institution of Civil Engineers of Ireland* (20 1891), pp. 161–207 (cit. on p. 55).

[Man95] R. Manning. "On the Flow of Water in Open Channels and Pipes". In: *Transactions of the Institution of Civil Engineers of Ireland* (24 1895), pp. 179–207 (cit. on p. 55).

[MC+03] D. C. Mason, D. M. Cobby, M. S. Horritt, and P. D. Bates. "Floodplain Friction Parameterization in Two-Dimensional River Food Models Using Vegetation Heights Derived from Airborne Scanning Laser Altimetry". In: *Hydrological Processes* 17 (2003), pp. 1711–1732 (cit. on p. 55).

[Mel08] I. Melzer. *Service-orientierte Architekturen mit Web Services. Konzepte – Standards – Praxis*. In collab. with S. Eberhard, A. v. Hilliger Thiele, M. Flehmig, P. Sauter, B. Zengler, P. Tröger, W. Dostal, B. Stumm, M. Lipp, J. Vajda, and M. Jeckle. 3rd ed. Heidelberg: Springer, 2008 (cit. on p. 18).

[Men99] J. Menck. "Work Control for Newton Type Coupling". In: *Scientific Computing in Chemical Engineering II: Simulation, Image Processing, Optimization, and Control. Proceedings of the 2nd Workshop on Scientific Computing in Chemical Engineering*. Hamburg-Harburg, Germany, 26-28 May. Ed. by F. Keil, W. Mackens, H. Voß, and J. Werther. Berlin and Heidelberg: Springer, 1999, pp. 192–199 (cit. on pp. 118, 119).

[Mic09] P. Micikevicius. "3D Finite Difference Computation on GPUs Using CUDA". In: *Proceedings of 2nd Workshop on General Purpose Processing on Graphics Processing Units (GPGPU 2009)*, pp. 79–84 (cit. on p. 17).

[MM+08] S. Manos, M. Mazzeo, O. Kenway, P. V. Coveney, N. T. Karonis, and B. Toonen. "Distributed MPI Cross-site Run Performance Using MPIg". In: *Proceedings of the 17th International Symposium on High Performance Distributed Computing. HPDC 2008*. Boston, MA, USA, June 23-27, pp. 229–230 (cit. on pp. 15, 16).

[MMV99] W. Mackens, J. Menck, and H. Voß. "Coupling Iterative Subsystem Solvers". In: *Scientific Computing in Chemical Engineering II: Simulation, Image Processing, Optimization, and Control. Proceedings of the 2nd Workshop on Scientific Computing in Chemical Engineering*. Hamburg-Harburg, Germany, 26-28 May. Ed. by F. Keil, W. Mackens, H. Voß, and J. Werther. Berlin and Heidelberg: Springer, 1999, pp. 184–191 (cit. on p. 118).

[MRS07] W. S. Meyer, M. R. Ramirez, and J. M. Souza. "Spatial Query Broker in a Grid Environment". In: *Advances in Geoinformatics. VIII Brazilian Symposium on Geoinformatics (GeoInfo 2006)*. November 19-22, Campos do Jordão, Brazil, pp. 107–126 (cit. on p. 63).

[MT04] B. Merz and A. H. Thieken. "Flood Risk Analysis: Concepts and Challenges". In: *Österreichische Wasser-und Abfallwirtschaft* 56 (3-4 2004), pp. 27–34 (cit. on pp. 29, 34).

[Neb04] D. Nebert. *Developing Spatial Data Infrastructures: The SDI Cookbook, Version 2.0*. GSDI, 2004. URL: `http://www.gsdi.org/docs2004/Cookbook/cookbookV2.0.pdf` (visited on Jan. 27, 2011) (cit. on pp. 2, 42).

[NF+10] J. C. Neal, T. J. Fewtrell, P. D. Bates, and N. G. Wright. "A Comparison of Three Parallelisation Methods for 2D Flood Inundation Models". In: *Environmental Modelling & Software* 25 (4 2010), pp. 398–411 (cit. on p. 107).

[NG05] P. Niblett and S. Graham. "Events and Service-Oriented Architecture: The OASIS Web Services Notification Specification". In: *IBM Systems Journal* 44 (4 2005), pp. 869–886 (cit. on p. 71).

[NKB07] E. Nash, P. Korduan, and R. Bill. "Optimising Data Flows in Precision Agriculture Using Open Geospatial Web Services". In: *Precision Agriculture* 7 (2007), pp. 753–759 (cit. on p. 46).

[Nov10] P. Novák. *Hydraulic Modelling – An Introduction. Principles, Methods and Applications*. 1st ed. London: Spon Press, 2010 (cit. on p. 35).

[OS83] J. O'Rourke and K. J. Supowit. "Some NP-Hard Polygon Decomposition Problems". In: *IEEE Transactions on Information Theory* 29 (2 1983), pp. 181–190 (cit. on p. 87).

[Pas07] E. Pasche. "Flood Modelling in Urban Rivers – The State-of-the-Art and Where to Go". In: *Advances in Urban Flood Management*. Ed. by R. Ashley, S. Garvin, E. Pasche, A. Vassilopoulos, and C. Zevenbergen. Leiden: CRC Press / Balkema, 2007, pp. 59–89 (cit. on p. 35).

[PJ04] A. Pant and H. Jafri. "Communicating Efficiently on Cluster Based Grids with MPICH-VMI". In: *Proceedings of the 6th IEEE International Conference on Cluster Computing*. San Diego, CA, 20-23 September, pp. 23–33 (cit. on pp. 15, 16).

[PK09a] A. Padberg and C. Kiehle. "Geodateninfrastrukturen & Grid-Computing: Aktuelle Herausforderungen und Anwendungsbeispiele". In: *Geoinformatik 2009*. Osnabrück, Germany, March 31 - April 2, pp. 129–138 (cit. on pp. 67, 68).

[PK09b] A. Padberg and C. Kiehle. "Towards a Grid-Enabled SDI: Matching the Paradigms of OGC Web Services & Grid-Computing". In: *Spatial Data Infrastructure Convergence: Building SDI Bridges to Address Global Challenges. Proceedings of the GSDI 11 World Conference*. June 15-19, Rotterdam, Netherlands. 2009 (cit. on pp. 67, 68).

[PL02] A. Panatkool and S. Laoveerakul. "Decentralized GIS Web Services on Grid". In: *Proceedings of the Open Source Free Software GIS - GRASS Users Conference 2002*. Trento, Italy, 11-13 September. Dipartimento di Ingegneria Civile e Ambientale (cit. on p. 62).

[PM08] J. M. Park and Y. L. Murphey. "Edge Detection in Grayscale, Color, and Range Images". In: *Wiley Encyclopedia of Computer Science and Engineering*. Ed. by B. W. Wah. Wiley, 2008 (cit. on p. 52).

[Por07] C. Portele. *OpenGIS Geography Markup Language (GML) Encoding Standard, Version 3.2.1. OGC 07-036*. Ed. by C. Portele. Open Geospatial Consortium, Inc. (OGC), 2007 (cit. on pp. 44, 170).

[PZS09] O. Parra, J. Zambrano, and A. Stehr, eds. *Proceedings of the 8th International Conference on Hydroinformatics (HEIC 2009)*. Concepción, Chile, January 12-16.

[Rat07] S. Rath. "Model discretization in 2D hydroinformatics based on high resolution remote sensing data and the feasibility of automated model parameterisation". Dissertation. Hamburg University of Technology, 2007 (cit. on pp. 50–53, 55, 78, 79, 133).

[RJS05] D. d. Roure, N. R. Jennings, and N. R. Shadbolt. "The Semantic Grid: Past, Present, and Future". In: *Proceedings of the IEEE* 93 (3 2005), pp. 669–681 (cit. on p. 128).

[RKE09] W. Reinhardt, A. Krüger, and M. Ehlers, eds. *Geoinformatik 2009*. Osnabrück, Germany, March 31 - April 2.

[SA+09] N. Schrage, D. Antanaskovic, T. Jung, and E. Pasche. "KALYPSO – An Open Source Software Tool for Flood Studies in Rivers". In: *Proceedings of the 8th International Conference on Hydroinformatics (HEIC 2009)*. Concepción, Chile, January 12-16 (cit. on p. 121).

[Saa03] Y. Saad. *Iterative Methods for Sparse Linear Systems*. 2nd ed. Philadelphia: Society for Industrial and Applied Mathematics, 2003 (cit. on pp. 39, 115).

[SB+10] U. Schwiegelshohn, R. M. Badia, M. Bubak, M. Danelutto, S. Dustdar, F. Gagliardi, A. Geiger, L. Hluchý, D. Kranzlmüller, E. Laure, T. Priol, A. Reinefeld, M. Resch, A. Reuter, O. Rienhoff, T. Rüter, P. Sloot, D. Talia, K. Ullmann, R. Yahyapour, and G. v. Voigt. "Perspectives on Grid Computing". In: *Future Generation Computer Systems* 26 (8 2010), pp. 1104–1115 (cit. on p. 129).

[SB97] S. M. Smith and J. M. Bradey. "SUSAN – A New Approach to Low Level Image Processing". In: *International Journal of Computer Vision* 23 (1 1997), pp. 45–78 (cit. on p. 52).

[SBG96] B. F. Smith, P. E. Bjørstad, and W. Gropp. *Domain Decomposition. Parallel Multilevel Methods for Elliptic Partial Differential Equations*. Cambridge: Cambridge University Press, 1996 (cit. on p. 112).

[Sch70] H. A. Schwarz. "Über einen Grenzübergang durch alternierendes Verfahren". In: *Vierteljahrsschrift der Naturforschenden Gesellschaft in Zürich* (15 1870), pp. 272–286 (cit. on p. 112).

[SF93] G. L. G. Sleijpen and D. R. Fokkema. "BiCGstab(l) for Linear Equations Involving Unsymmetric Matrices with Complex Spectrum". In: *Electronic Transactions on Numerical Analysis* (1 1993), pp. 11–32 (cit. on pp. 39, 112).

[SG04] O. Schenk and K. Gärtner. "Solving Unsymmetric Sparse Systems of Linear Equations with PARDISO". In: *Journal of Future Generation Computer Systems* 20 (3 2004), pp. 475–487 (cit. on p. 110).

[SG+05] P. Schut, X. Geng, M. Newby, S. Fellah, S. Keens, W. Li, M. Kyle, C. Kiehle, C. Heier, M. Adair, N. Ostländer, H. Borsutzky, and A. Whiteside. *OGC Web Processing Service (WPS) Specification, Version 0.4. OGC 05-007r4*. Ed. by P. Schut and A. Whiteside. Open Geospatial Consortium, Inc. (OGC), 2005 (cit. on pp. 45, 69).

[SG06] O. Schenk and K. Gärtner. "On Fast Factorization Pivoting Methods for Symmetric Indefinite Systems". In: *Electronic Transactions on Numerical Analysis* (23 2006), pp. 158–179 (cit. on p. 110).

[SH+08] B. Šimo, O. Habala, E. Gatial, and L. Hluchý. "Leveraging Interactivity and MPI for Environmental Applications". In: *Computing and Informatics* 27 (2 2008), pp. 271–284 (cit. on pp. 106, 107).

[She02] J. R. Shewchuk. "Delaunay Refinement Algorithms for Triangular Mesh Generation". In: *Computational Geometry: Theory and Applications* 22 (1-3 2002), pp. 21–74 (cit. on pp. 47–49).

[She68] D. Shepard. "A Two-Dimensional Interpolation Function for Irregularly-Spaced Data". In: *Proceedings of the 23rd ACM National Conference (ACM 1968)*, pp. 517–524 (cit. on p. 88).

[She96] J. R. Shewchuk. "Triangle: Engineering a 2D Quality Mesh Generator and Delaunay Triangulator". In: *Applied Computational Geometry: Towards Geometric Engineering. FCRC'96 Workshop, WACG'96*. Philadelphia, PA, May 27-28. Ed. by M. C. Lin and D. Manocha. Vol. 1148. Lecture Notes in Computer Science 1148. Berlin and Heidelberg: Springer, 1996, pp. 203–222 (cit. on pp. 47, 133).

[Sib81] R. Sibson. "A Brief Description of Natural Neighbor Interpolation". In: *Interpolating Multivariate Data*. Ed. by V. Barnett. Chichester: Wiley, 1981, pp. 21–36 (cit. on p. 88).

[Sim08] I. Simonis. *OGC Sensor Web Enablement Architecture, Version 0.4. OGC 06-021r4*. Ed. by I. Simonis. Open Geospatial Consortium, Inc. (OGC), 2008 (cit. on p. 45).

[SS+09] M. Smith, M. Schmidt, N. Fallenbeck, T. Dörnemann, C. Schridde, and B. Freisleben. "Secure On-Demand Grid Computing". In: *Future Generation Computer Systems* 25 (3 2009), pp. 315–325 (cit. on p. 20).

[SS09] B. Schäffer and S. Schade. *OWS-6 Geoprocessing Workflow Architecture Engineering Report. OGC 07-143r1*. Open Geospatial Consortium, Inc. (OGC), 2009 (cit. on p. 46).

[SS86] Y. Saad and M. H. Schultz. "GMRES: A Generalized Minimal Residual Algorithm for Solving Nonsymmetric Linear Systems". In: *SIAM Journal on Scientific and Statistical Computing* 7 (3 1986), pp. 856–869 (cit. on p. 39).

[ST+07] D. A. Spielman, S.-H. Teng, A. Üngör, and M. Goodrich. "Parallel Delaunay refinement: Algorithms and analyses". In: *International Journal of Computational Geometry and Applications* 17 (1 2007), pp. 1–30 (cit. on p. 49).

[STA95] A. Schweikard, R. Tombropoulos, and J. R. Adler. "Robotic Radiosurgery with Beams of Adaptable Shapes". In: *Computer Vision, Virtual Reality and Robotics in Medicine*. First International Conference, CVRMed '95, Nice, France, April 3–6, 1995 Proceedings, pp. 138–149 (cit. on p. 87).

[Sv95] G. L. G. Sleijpen and H. A. van der Vorst. "An Overview of Approaches for the Stable Computation of Hybrid BiCG Methods". In: *Applied Numerical Mathematics* 19 (3 1995), pp. 235–254 (cit. on pp. 39, 112).

[SZ07] B. Stollberg and A. Zipf. "OGC Web Processing Service Interface for Web Service Orchestration – Aggregating Geo-processing Services in a Bomb Threat Scenario". In: *Proceedings of the 7th International Symposium on Web and Wireless Geographical Information Systems (W2GIS 2007)*. November 28-29, Cardiff, UK, pp. 239–251 (cit. on p. 46).

[SZ08] B. Stollberg and A. Zipf. "Geoprocessing Services for Spatial Decision Support in the Domain of Housing Market Analyses-Experiences from Applying the OGC Web Processing Service Interface in Practice". In: *Proceedings of the 11th AGILE International Conference on Geographic Information Science. Taking Geoinformation Science One Step Further. Girona, Spain, May 5-8*. May 5-8, Girona, Spain (cit. on p. 46).

[SZZ06] Y. Shu, J. F. Zhang, and X. Zhou. "A Grid-enabled Architecture for Geospatial Data Sharing". In: *Proceedings of the 2006 IEEE Asia-Pacific Conference on Services Computing (APSCC 2006)*. December 12-15, Guangzhou, Guangdong, China, pp. 369–375 (cit. on p. 63).

[Ter10] A. R. Terrel. "Finite Element Method Automation For Non-Newtonian Fluid Models". Dissertation. University of Chicago, 2010 (cit. on p. 38).

[TH04] V. Tran and L. Hluchý. "Parallelizing Flood Models with MPI. Approaches and Experiences". In: *Computational Science – ICCS 2004*. 4th International Conference, Kraków, Poland, June 6-9, Proceedings, Part I. 4 vols., pp. 425–428 (cit. on p. 104).

[TW05] A. Toselli and O. B. Widlund. *Domain Decomposition Methods. Algorithms and Theory*. Berlin and Heidelberg: Springer, 2005 (cit. on pp. 112, 118).

[Ver06] A. Verwey. "Trends in the Numerical Modelling of Floods". In: *Proceedings of the 4th Annual Mekong Flood Forum. Improving Flood Forecasting and Warning Systems for Flood Management and Mitigation in the Lower Mekong Basin*. 18-19 May 2006, Siem Reap, Cambodia (cit. on p. 2).

[Ver07] A. Verwey. "Hydroinformatics Support to Flood Forecasting and Flood Management". In: *Water in Celtic Countries. Quantity, Quality and Climate Variability*. Proceedings of the Fourth InterCeltic Colloquium on Hydrology and Management of Water Resources, Guimarães, Portugal, July 2005, pp. 23–36 (cit. on p. 2).

[Vre10] P. A. Vretanos. *OpenGIS Web Feature Service (WFS) 2.0 Interface Standard. OGC 09-025r1 and ISO/DIS 19142*. Ed. by P. A. Vretanos. Open Geospatial Consortium, Inc. (OGC), 2010 (cit. on pp. 44, 172).

[WH08] G. Wasson and M. Humphrey, eds. *HPC File Staging Profile, Version 1.0. OGF GFD-R-P.135*. Open Grid Forum (cit. on p. 69).

[WT07] J. M. Ware and G. E. Taylor, eds. *Proceedings of the 7th International Symposium on Web and Wireless Geographical Information Systems (W2GIS 2007)*. November 28-29, Cardiff, UK. Berlin and Heidelberg: Springer.

[ZA+10] C. Zevenbergen, R. Ashley, S. Garvin, E. Pasche, N. Evelpidou, and A. Cashman, eds. *Urban Flood Management*. Leiden: CRC Press / Balkema, 2010 (cit. on p. 102).

[ZT05] T. Zhang and M.-H. Tsou. "The Integration of Grid-enabled Internet GIServices and Geographic Semantic Web Technologies". In: *Geographic Information Sciences* 11 (1 2005), pp. 15–23 (cit. on p. 64).

# Glossary

**Bathymetry**
The shape of the terrain below the water surface, e. g. in oceans and river beds. See also the entry for topography.

**BPEL**
The Business Process Execution Language (BPEL) is a workflow description language.

**CORINE**
CORINE (Coordinated Information on the European Environment) is a project for the acquisition of a digital land cover classification obtained from satellite imagery.

**DEM**
A digital elevation model, here a common term for DSM and DTM, is a representation of the earth's surface. If a distinction is necessary, the more specific term will be used.

**DSM**
Digital surface model, a DEM capturing the surface elevation of the earth as measured by remote sensing methods, e. g. LiDAR. It includes measurements that lie on the top of trees or buildings.

**DTM**
Digital terrain model, a DEM representing the bare ground surface of the earth. A DTM is created from a DSM by a filtering algorithm. Elevated objects, such as vegetation or buildings, have been filtered out.

**GIS**
A geographic information system (GIS) has the capabilities of storing, managing, manipulating, analyzing, and visualizing geographical data, i. e. data related to a location on earth (also called geodata or spatial data).

**gLite**
LHC Computing Grid (LCG) / gLite is a grid middleware distribution.

**Globus Toolkit**
An open source grid middleware distribution provided by the Globus Alliance beginning in 1995 [FK97; FK99b], currently at version 5. Grid services in this thesis are founded upon Globus Toolkit 4.

**GML**
Geography Markup Language (GML) is a modeling language and exchange format for geographic data [Por07], see Section 4.1.

**GRAM**
Grid Resource Allocation and Management (GRAM) provides an interface for remote job submission to compute resources and execution management in a Globus Toolkit grid. GRAM is able to interact with different batch systems.

**GridFTP**
A file transfer protocol based on the well-known FTP protocol, which has been tailored to high-performance networks and grid security.

**GSI**
Grid Security Infrastructure (GSI), formerly known as Globus Security Infrastructure, is a specification for secret, tamper-proof, delegatable communication between software in a grid computing environment.

**HPC**
High-performance computing (HPC) here refers to the use of applications designed for tightly coupled parallel execution on a supercomputer or computing cluster.

**MPI**
The Message Passing Interface (MPI) is the de-facto standard for developing HPC applications in a computing cluster.

**OASIS**
The Organization for the Advancement of Structured Information Standards (OASIS) is a global, non-profit consortium driving the development of e-business and web service standards.

**OGC**
The Open Geospatial Consortium, Inc.® (OGC) is the leading international voluntary consensus organization for standardization of geospatial information and services.

**OGF**
The Open Grid Forum (OGF) is an organization for the standardization of grid technologies.

**OGSA**
The OGF has issued the Open Grid Services Architecture (OGSA) as a common, standard, and open SOA for grid-based applications.

**OGSA-DAI**
Open Grid Services Architecture for Data Access and Integration (OGSA-DAI) is an open source framework for accessing and integrating distributed data resources based on WSRF grid services.

**OpenMP**
Open Multi-Processing (OpenMP) is a programming interface for the development of shared-memory parallel applications on multi-core processors.

**SDI**
A spatial data infrastructure (SDI) enables the provision and use of geographic data across organizational boundaries using well-known standards.

**Shapefile**
Shapefile is the de-facto standard for storing spatial vector data, which emerged from the commercial GIS software packages ArcView and ArcGIS by Esri Inc. (Environmental Systems Research Institute).

**SOA**
Service-oriented architecture (SOA) is an abstract software architecture focusing on offering, searching, and using services as basic building blocks.

**SOAP**
Simple Object Access Protocol (SOAP) is a protocol specification for communication between web services in a SOA.

**SOS**
OGC Sensor Observation Service [BSE12], see Section 4.1.

**Topography**
The shape of the terrain surface, typically referring to features above the water surface (see also the entry for bathymetry).

**UNICORE**
UNiform Interface to COmputing REsources (UNICORE) is a grid middleware distribution.

**Virtual Organization**
A virtual organization (VO) is a collection of real organizations or individuals sharing a common goal. For the use of this term in grid computing see Section 2.3.1.

**W3C**
The World Wide Web Consortium (W3C) is an international community that develops standards to ensure the long-term growth of the web (`www.w3.org`).

**WCS**
OGC Web Coverage Service [Bau10], see Section 4.1.

**WFS**
OGC Web Feature Service [Vre10], see Section 4.1.

**WPS**
OGC Web Processing Service [FH+07], see Section 4.1.

**WSDL**
The Web Services Description Language (WSDL) describes the functional and technical details of a web service interface required for discovering and calling service operations in a SOA.

**WSRF**
The Web Services Resource Framework (WSRF) is a concrete implementation specification for the OGSA issued by the OGF.