Improving the Reuse of Services in Geospatial Applications with XMDD Technology

Samih Al-Areqi
Institute of Computer Science
University of Potsdam
Potsdam, Germany
samih@cs.uni-potsdam.de

Anna-Lena Lamprecht
Institute of Computer Science
University of Potsdam
Potsdam, Germany
lamprecht@cs.uni-potsdam.de

Tiziana Margaria
Institute of Computer Science
University of Potsdam
Potsdam, Germany
CSIS, University of Limerick, and
Lero, the Irish Software Research Center
Limerick, Ireland
margaria@cs.uni-potsdam.de

Abstract— In recent years, the geospatial application domain has embraced component-based development and service orientation to support software reuse. However, due to the specific characteristics of geospatial applications, caused by complex and comprehensive analysis processes and heterogeneous data, the reuse of services faces particular barriers in this domain. Providing application experts without a strong programming or technical background with simple means to reuse these services is an important challenge. This paper describes how we followed the eXtreme Model-Driven Development (XMDD) paradigm to improve the reuse of geospatial services, namely by (1) performing rigorous service abstraction of geospatial tools to be reused in large scale applications, (2) using the java electronic tools integration (jETI) technology for enabling the remote execution and integration of services, and (3) supporting service composition at the user level by using the java application building center (jABC) process modeling framework. Concretely, we discuss how we improved the reuse of services for the assessment of the impacts of sea-level rise.

Keywords— Service reuse, geospatial applications, scientific workflows, agile methodologies, extreme model-driven design.

I. INTRODUCTION

Building applications based on the reuse of existing components or services has noticeably increased in many domains. In the geospatial application domain, big geographic data, lack of interoperability, and complex analysis processes constitute barriers to ensuring a successful and wide reuse of components and services. Service-oriented architecture (SOA) principles and Web Service technology have been embraced by the geospatial domain and many works quickly followed the trend of building geospatial applications by reusing components and services. Several works focused on the construction of domain-specific applications by assembling and reusing geospatial processes and data as services [1, 2, 3]. To facilitate the reuse of geospatial services, in the last decade many researchers followed the Open Geospatial Consortium’s (OGC) Web Service standards [4] to build geospatial applications by composing services (eg. [5, 6, 7, 8, 9]). Workflow technologies such as jOpera have been applied early to the geospatial domain [10], and also the Kepler scientific workflow system [11] has soon been applied to handle distributed geospatial data processing using Web Services [12] and to compose OGC services [13,14]. Other works used BPEL-based business workflow technology to orchestrate geospatial services [15]. Nevertheless, learning how to apply these technologies to build a system based on
services remains complex for application experts, in particular with the interoperability challenges of geospatial data. A result from embracing service orientation in the geospatial domain is that the scientific data has become increasingly remotely accessible in a distributed fashion through standardized geospatial Web Services [2]. Thus, scientific communities become more aware of the benefits of sharing their data and computational services, and are thus contributing to distributed data and services. However, researchers should also not be too occupied with exploring how to reuse and compose geospatial services in order to develop own software applications tailored to their specific needs.

Despite substantial efforts by the OGC to provide standards for geospatial Web Services, turning spatial data and processes into loosely coupled components and interoperable geospatial services is suffering from the technical complexity of using the standards. In addition, there is a lack of a framework for facilitating service execution, thus users face a great challenge when it comes to servification, that is, the process of turning arbitrary software components into proper services. Attempts were made to improve the reuse of service in geospatial applications for end users (application experts), and some works addressed technical complexities of workflow systems by enabling Web-based workflow composition and editing [16, 12], while others proposed a model-driven way of geospatial Web Service composition [17]. Lately, cloud technology has been used to support efficient resource allocation and execution for scientific workflows [18, 19]. However, more technical efforts are required to handle the lightweight geospatial service execution in the cloud as described in [20].

The aim of this paper is to show how we follow the eXtreme Model-Driven Development (XMDD) paradigm [21] to achieve an improvement of geospatial services reuse. We do this by (1) performing servification of sea-level rise impacts analysis tools and data, (2) using the jETI technology for the remote integration and execution of the services, and (3) enabling users to compose services into workflows using the jABC framework. The rest of the paper is organized as follows: Section II gives an overview of component and service reuse, geospatial services, scientific workflow technology and agile methodology, in particular Section II-A introduces the jABC framework and Section II-B gives a summary about the jETI framework. Section III describes the proposed approach to address service reuse, which comprises servification, service execution, and service reuse. Finally, Section 0 discusses conclusions and plans for future work.

II. BACKGROUND

Software reuse ranges from simple functions to complete applications and is often considered the most effective means for improvement of productivity and maintainability in software development projects. The emergence of paradigms such as component-based software engineering (CBSE) and service-oriented software engineering (SOSE) has leveraged the development of applications based on reuse of existing components and services. It significantly increased the possibilities of building systems and applications from reusable components [22]. CBSE aims at encouraging reuse of software applications, where systems are built by assembling components already developed and prepared for integration. In addition, it leverages the emergence of middleware technologies, such as object standards, to make software reuse a reality [23]. Although CBSE has proven to be successful for software reuse and maintainability, software developers are facing today more complexities, such as varying platforms, varying protocols, various devices, etc. [24].

Services are a natural further development of software components. They can be defined as loosely coupled reusable software components that encapsulate discrete functionality [25]. The paradigm of service-oriented software engineering overcomes the issues of heterogeneity and interoperability challenges of CBSE by defining standards to support easy service reuse and composition for system developers. Web Service standards are defined to represent computational or information resources that can be used by other applications. Service-oriented architectures (SOA) support distributed systems development based on service reuse. The major benefit from SOA standards (such as WSDL to describe services) is to enable interoperability across applications over different platforms.

The interoperability challenge in the geospatial domain and the advancements in general Web Service technologies and in GIS service standards such as Spatial Data Infrastructures (SDI) and OGC Web Service standards encouraged the migration from the traditional form of stand-alone geospatial applications to loosely coupled components, interoperable geospatial services, and grid computing. The OGC standards that are based on the service-oriented architecture have been designed to ease the reuse and integration of geospatial Web Services. However, they do not comply with the Web Service standards as defined by the W3C and OASIS. Therefore, developing geospatial services and composing them based on OGC standards requires additional technical efforts from both developers and users.

Scientific workflow technologies aim to facilitate and support the composition and execution of complex analysis processes in a flexible fashion [26]. In contrast to the communication- and document-oriented workflows in the business domain, scientific workflows are data- and computation-oriented. Despite their promise to simplify the service composition process, scientific workflow management systems are often inherently complex and challenging in use and design, especially where the managed resources are heterogeneous. Furthermore, many current workflow technologies are designed to support service composition at a lower, technical level, and not at a level where average users.
can handle the composition and execution tasks. Composing 
services of geospatial applications in such workflows has a 
great focus on the data flow, and the underlying computation 
infrastructure has a major impact on the execution of 
the workflows. While clusters and grids are traditionally used to 
run large-scale scientific workflows, lately the trend is to 
execute the scientific workflows in hosting platforms such as 
clouds. Cloud computing "enables small and medium sized 
companies to deploy their Web-based applications in an 
instant scalable fashion without the need to invest in large 
computational infrastructures for storing large amounts of data 
and/or performing complex processes" [27]. Further, the use 
of VM images in the cloud to store computational 
environments and on-demand provisioning capabilities will 
improve reproducibility, which is significantly important for 
scientific workflows [28]. Users need however programming 
environments that support an easy design and execution of the 
scientific workflows.

A. jABC

Agile methods in the spirit of [29] have become 
increasingly popular in software development. Their core 
principle is to open software development to customers and 
users, in order to improve productivity, quality and 
stakeholder collaboration and satisfaction. The eXtreme 
Model-Driven Design (XMDD) paradigm [21] is an extremely 
rigorous way of model-driven development that supports a 
very agile and cooperative development of service-oriented 
systems by turning system development into user-centric 
orchestration of intuitive service functionality [30]. The multi-
purpose process modeling and execution framework jABC 
[31] inherits the power of XMDD to enable end users to easily 
use and compose services into agile workflows. Its way of 
handling the collaborative design of complex software systems 
has proven to be effective and adequate for the cooperation of 
non-programmers and technical people. It enhances other 
modeling practices like the UML-based RUP (Rational 
Unified Process) and by leveraging plugin technology 
supports most activities needed along the development 
lifecycle like animation, rapid prototyping, formal verification, 
debugging, code generation, and evolution. In fact, compared 
with other workflow systems, the jABC offers a number of 
advantages that play a particular role when integrating off-the-
shelf, possibly remote functionalities [32]:

• Simplicity: Focusing on application experts, who are 
typically non-programmers. The basic ideas of the 
modeling process have been explained in past projects 
to new participants in less than one hour.

• Agility: Models, and artifacts change over time based 
on expected requirements, therefore the process 
supports evolution as a normal process phase.

• Customizability: The building blocks which form the 
model can be freely renamed or restructured to fit the 
habits of the application experts.

• Consistency: The same modelling paradigm underlies 
the whole process, from the very first steps of 
prototyping up to the final execution, guaranteeing 
traceability and semantic consistency.

• Verification: With the model checking plugin, the 
jABC supports users to consistently modify their 
models. The basic idea is to define local or global 
properties that the model must satisfy and to provide 
automatic checking mechanisms.

• Service orientation: Existing or external features, 
applications, or services can be easily integrated into a 
model by wrapping the existing functionality into 
building blocks that can be used inside the models.

• Executability: The model can have different kinds of 
execution code. These can be as abstract as textual 
descriptions (for example in the first animations during 
requirement capture), and as concrete as the final 
runtime implementation.

• Universality: Based on Java as largely platform-

independent, object-oriented implementation language, 
jABC can be easily adopted in a large variety of 
technical contexts and of application domains.

The service concept of jABC is very close to an intuitive 
understanding of service that is required to be ubiquitously 
accessible (location-agnostic) and mechanically configurable 
[33]. The term service is used to denote functional building 
blocks (SIBs), which are viewed as independent from their 
location, the program entity, and hardware-platform which 
provides them. The SIBs are orchestrated with their 
operational or behavioral semantics in mind. Concretely, this 
means that each SIB, once activated, executes its logic and 
upon termination triggers subsequent SIBs according to the 
outcome of this execution. This methodology of composition 
has been termed lightweight process coordination [9], 
focusing on operational aspects of the application rather than 
structural properties of the software. The notion of service in 
jABC is therefore fundamentally different from the Web 
Service notion. The ties to Web-communication protocols are 
not an essential part of jABC, but provided by the jETI 
technology [34]. The jABC process modeling and execution 
framework [31] has been applied to support agile workflows 
in different scientific applications domains in the last years, 
predominantly in the field of bioinformatics and for geospatial 
applications (cf. [35, 36, 37]). The framework has furthermore 
been extended by functionality for semantics-based semi-
automatic service composition, which has been shown to be 
beneficial especially for dealing with variant-rich scientific 
workflows [35].

B. jETI

The Java-based jETI [34] is a redesigned version of the 
Electronic Tool Integration (ETI) [38] platform, an open
platform for the interactive experimentation with and the coordination of heterogeneous software tools via the internet. It was designed to provide:

- tool users with an instant hands-on experience with the tools, without need to download and install the software - which too often costs a considerable amount of effort and time, and
- tool providers with an environment where they may publish and promote their tools, making experimentation available to end-users without the burden and legal issues of direct distribution, and where they may receive valuable feedback.

Although the ETI platform offered a good solution to integrate software tools remotely, its servers were too complicated for both the tool providers and users. To follow the rapid development methodologies, the jETI framework overcomes these problems by applying newer technologies and standards that internally base on Web services and Java technology. It replaces the requirement of physical tool integration of the original ETI approach by very simple registration and publishing platform. Corresponding to the Web services functionality and service description standards such as WSDL, jETI uses an HTML tool configurator to create service descriptions. This allows providers to register a new tool functionality just by uploading the tool to the server and filling the description information (interface definition, input and output parameters, etc.) into a simple template form. All this information is internally maintained in an XML file and available for further use. For example, SIBs for use in the jETI framework can be generated automatically from the specifications, so that the services can easily be used within the jABC. Thus, with the lightweight remote service technology of jETI, users are able to

1. considerably simplify the integration process, and at the same time
2. flexibilize the distribution, version management and use of integrated tools,
3. broaden the scope of potential user profiles and roles from different application domains to solve complex problems and
4. solve the scalability problem connected with tool maintenance and evolution.

III. MAIN APPROACH

In this section we discuss how we used the jABC and jETI technologies to improve the reuse of geospatial services. As shown in Figure 1, the methodology involves three phases: First, the scientific tools (in this case tools for sea-level rise impacts analysis) which are used for geospatial applications, are servified (turned into services). Second, these services are reused to construct geospatial applications in the form of workflows, and finally the workflows (WF) are executed, accessing the remote services. A concrete description of each phase is given in the following sections.

A. Servification

Several tools and applications have been developed to analyze the risk index of climate impacts, such as data creation, conversion, and visualization tools. The scientific tools that we used for our application address the analysis of the impacts of sea-level rise. These tools are used in the ci:grasp1 climate information platform. They are based on scripts in the GNU R language that comprises several tools for spatial analysis. The srmtools-package [39] used for the data analysis provides the methods required to produce results as presented on ci:grasp. It combines various tools that are based on different packages. For instance, a raster package tool2 for data reading, writing, manipulating, analyzing and modeling of gridded spatial data, the Gdal tool3 for data conversion, and other packages for data visualization such as Png4 and plotGoogleMaps [40].

According to the service orientation paradigm, which postulates that any kind of computational resource should be seen and handled as a service – that is, a well-defined unit of functionality with a well-defined interface – to provide a high level of abstraction and reusability (cf., e.g., [41]), we use the term servification to refer to the process of turning arbitrary software components into proper services that are adequate, for example, for (re-) use in workflow management systems. Concretely, in the servification phase, the analysis processes of sea-level rise impacts implemented for ci:grasp and coded in R scripts have been decomposed into loosely coupled services. The decomposition handled service reuse by determining the most frequently used process steps in various applications of climate impact assessment and perform rigorous abstraction to ensure a great level of reuse for the services. Through jETI, a description for each service, equipped with well-defined inputs and outputs, is configured on the server and connected with the corresponding script file. After that, services are generated automatically into SIBs, so that they can easily be consumed by the jABC.

---

1 http://www.cigrasp.org
2 http://cran.r-project.org/web/packages/raster/
4 http://www.rforge.net/png
So far, 17 services for different data creation, computation, and data output tasks have been created (see Table II). Concretely, its three subclasses of SLR services concern: data creation (comprising 6 services), computation (6), and output (5). With regard to working with the jABC, this is the domain modeling phase, which enables us to model the domain of the sea-level rise example by integrating such created services and organize them in domain-specific taxonomies, so that they are ready for use in the actual workflow design phase. Figure 4 shows how the SLR services can be taxonomically classified and categorized into three groups:

- Data creation (loading, clipping, masking and converting data)
- Computation (of flooded areas, yield loss, caloric energy loss and land loss classes)
- Output generation (creation of PNG, PDF, TXT, GeoTiff/ASCII output files and result visualization in an interactive map)

Table I. APPLICATION OBJECTIVES TO ASSESS SLR IMPACTS

<table>
<thead>
<tr>
<th>Application</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>compute rural and urban GDP at risk</td>
<td>focuses on potential economic damage in coastal communities</td>
</tr>
<tr>
<td>compute population at risk of migration</td>
<td>focuses on the number of people that would be affected</td>
</tr>
<tr>
<td>compute potential yield loss</td>
<td>compute potential production value affected in USD</td>
</tr>
<tr>
<td>compute potential land loss (ha)</td>
<td>determine the area that will be potentially inundated</td>
</tr>
<tr>
<td>compute potential production affected ($)</td>
<td>focuses on the economic value of the agricultural loss</td>
</tr>
<tr>
<td>compute potential caloric energy loss</td>
<td>focuses on the potential number of peoples annual diets lost</td>
</tr>
</tbody>
</table>

B. Service Reuse

To increase service reuse, a significant aspect is to facilitate service consumption and to make the composition easy and flexible for a wide range of communities and people, so that the scientific community (e.g. geospatial application experts) can use and understand the service principles and build applications by means of service compositions. This section demonstrates how the agile methodologies supported by the jABC framework make an essential contribution to increasing geospatial service reuse. Concretely, we will show how based on the newly created domain-specific services and the large library of SIBs for common functionality that comes with the jABC framework, we easily construct different workflows for SLR impact assessment in an agile workflow-based way.

Figure 2 shows a simple workflow for assessing the impact of sea-level rise on the agricultural yield loss for a region to be selected by the user. From top to bottom, the services belong to three different groups of functionalities (data creation, application-specific computation services, and output generation). Starting in the upper left corner (the SIB with the underlined name denotes the starting point), the workflow performs (1) definition of the investigated area by coordinates of name; (2) downloading the digital elevation model of the selected area; (3) entering the magnitude of sea level rise; (4) computation of the flooded area; (5) load raster data from yield dataset; (6) resample two different data sets (in this example land loss data with yield data); (7) computation of the yield loss cause by the flooding; and (8) generation of an output file with results in an interactive Google map.

Table II. FREQUENCY OF SERVICE REUSE

<table>
<thead>
<tr>
<th>Service S</th>
<th>No. Reuses of S</th>
<th>No. Workflows Using S</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data creation services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Load raster data</td>
<td>28</td>
<td>11</td>
</tr>
<tr>
<td>Load SRTM data</td>
<td>11</td>
<td>11</td>
</tr>
<tr>
<td>Resampling</td>
<td>23</td>
<td>10</td>
</tr>
<tr>
<td>Clipping</td>
<td>6</td>
<td>6</td>
</tr>
<tr>
<td>Masking</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>ConvertKgTokcal</td>
<td>4</td>
<td>4</td>
</tr>
<tr>
<td>Computation services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Compute flooded area</td>
<td>10</td>
<td>10</td>
</tr>
<tr>
<td>Compute land loss classes</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>Compute population at risk</td>
<td>2</td>
<td>2</td>
</tr>
<tr>
<td>Compute yield loss</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Identify agriculture area</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Identify flooded agriculture area</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>Output services</td>
<td></td>
<td></td>
</tr>
<tr>
<td>Produce Pdf file</td>
<td>55</td>
<td>11</td>
</tr>
<tr>
<td>Produce image file</td>
<td>55</td>
<td>11</td>
</tr>
<tr>
<td>Produce GeoTiff file</td>
<td>55</td>
<td>11</td>
</tr>
<tr>
<td>Produce text file</td>
<td>55</td>
<td>11</td>
</tr>
<tr>
<td>Generate interactive map</td>
<td>55</td>
<td>11</td>
</tr>
</tbody>
</table>

In order to support the reuse of workflows, multiple abstraction levels have been introduced by making use of the hierarchical modeling capabilities of the jABC. Some of the
SIBs in the figure are marked by a green circle, which indicates that the functionality represented by this building block is actually more complex and defined by a separate (sub-) model. For example, SIB (7) encapsulates a (sub-) model for the computation of the yield loss (shown in Figure 3. Note that it again makes use of other (sub-) models, as the SIB to select potential yield data is a composite service that allows for the computation of several types of yield loss for different climate scenarios. This hierarchical modeling style allows to organize workflow applications at different levels of abstraction, from coarse-granular and more conceptual views at the higher levels, down to fine-granular and more technical views at the lower levels. The current SLR workflow scenario comprises six different computations (applications), as summarized in Table I.

According to different objectives to assess SLR impacts, each workflow application has several variations of workflow instances. For evaluating the reuse of geospatial services in the workflow variations, we used the jABCstats framework [42] to calculate the frequently of services reuse. Table II shows that the 17 created services as described and classified in section III-A have been reused 392 times in total, and within 11 workflow variations for sea level rise impact analysis. This also reflects that the services have contributed to a significant number of reuses in the different workflow applications. Not surprisingly, that data creation and output generation services are reused for all SLR applications. Figure 4 depicts the taxonomic classification and reuse levels of all services. Note that these services could also be reused in other analyses of climate change drivers included in ci:grasp, such as changes in temperature and precipitation and increased drought risk, and to other risk analyses related to climate impacts. Furthermore, the services of data creation, resampling and output generation and visualization are more likely to be reused in the geospatial application domain in general.

C. Service execution

We believe that performing rigorous servification and providing an easy and flexible way to consume services in geospatial applications significantly improves their reuse. However, geospatial services deal with large data sets and need comprehensive computing resources.

In this section we show how jETI handles the remote execution of geospatial services. As mentioned in section III-A, the created services are based on several packages and use a diversity of data sets (e.g., elevation, land-use, population density or yield data). Consequently, these packages and data and the pre-configuration corresponding to the operating system platform are required to perform the execution of services. The jETI platform offers a lightweight remote component (tool) to further simplify integration and execution of software tools, it can be seen as a tool that enhances other tools and frameworks by the integration, organization and execution of remote functionalities, so that users do not have to deal with the required configuration to execute the services.
In our case, we use the jETI server to support a convenient and flexible platform that enables users to execute geospatial services without dealing with the related configurations. On the jETI server, script files for created services are installed and wrapped to enable convenient automated invocation. The required configuration includes the installation of the GNU R language and packages such as Raster, Rgdal, ClassInt, Png and plotGoogleMapall. The jETI server itself runs in a virtual machine image based on a Debian Linux operating system. Managing the underlying infrastructure can be an issue as well, thus we follow the recent trend of using cloud technology to host our server and services. Thus, in our solution, the users design their workflow applications with the jABC, which during workflow execution submits jobs to the cloud where the jETI services for risk analysis of SLR impacts are hosted, as shown in Figure 5. This allows us to benefit from the advantages of cloud, such as resource scalability and data availability for other users to run their own workflows.

![Interaction of users, the jABC and the jETI environment.](image)

**IV. CONCLUSION**

Due to the increase of using GIS in a wide range of domains, software reuse and data sharing become more important. The service orientation paradigm has been developed to support software reuse. However, the Geospatial services have their own characteristics, such as complex processes and big data sets, that hamper the service reuse. In this light, the approach presented in this paper aims to improve the reuse of geospatial services by applying XMDD-based technologies such as jABC and jETI, and it focuses on the reuse challenge from three perspectives:

1. Performing rigorous servification by turning basic components as well as their compositions into flexibly reusable pieces of functionality,
2. enabling flexible and easy service consumption to reuse and compose services in an agile workflows which free end users from the burdens of learning programming/scripting languages and other required technologies to design and adapt workflows,
3. offering a suitable environment to handle comprehensive geospatial processing by supporting remote execution and integration of services.

In the example presented in this work, we discussed how the reuse of services used in the analysis of sea-level rise impacts is improved. The next step may be to perform a similar servification process for other (scientific) tools, extending the library with additional and alternative general and geospatial services. Moreover, a more flexible inclusion of various, heterogeneous data sources could be achieved with additional SIBs. However, to support easy and correct reuse of services also in large-scale applications, the core of our future work is going to address the semantically aware reuse of geospatial services by designing domain-specific ontologies. Once a semantics-based workflow design framework is available, the reuse of services in geospatial applications by a larger audience will become possible.

**ACKNOWLEDGMENT**

The authors would like to thank Steffen Kriewald, Dominik Reusser, and Markus Wrobel from the climate change and development group at the Potsdam Institute for Climate Impact Research (PIK), who provided us with the tools and data sets required for the scenario. Special thanks go to Sven Lehmann for his support with the server configuration.

This work was supported, in part, by Science Foundation Ireland grant 13/RC/2094 to Lero - the Irish Software Research Centre (www.lero.ie). Also supported, in part, by German Academic Exchange Service (DAAD) grant 2014/15 (57076385).

**REFERENCES**


