Research Article

Designing Service Architectures for Distributed Geoprocessing: Challenges and Future Directions

Anders Friis-Christensen  
Institute for Environment & Sustainability  
European Commission Joint Research Centre  
Ispra, Italy

Michael Lutz  
Institute for Environment & Sustainability  
European Commission Joint Research Centre  
Ispra, Italy

Nicole Ostländer  
Institute for Environment & Sustainability  
European Commission Joint Research Centre  
Ispra, Italy

Lars Bernard  
Department of Geosciences  
Technische Universität Dresden

Abstract

In this paper we study the feasibility of using services offered by a Spatial Data Infrastructure as a basis for distributed service oriented geoprocessing. By developing a prototype we demonstrate that a Spatial Data Infrastructure facilitates rapid development of applications that solve typical problems for an existing risk management application. The prototype provides users with a distributed application that enables the assessment of fire damage areas based on land cover data in a given area. The services involved in the application include: Web Feature Services, Web Map Services, a Gazetteer Service, a Catalogue Service, and Geoprocessing Services. We present the architecture of the application and describe details about implementation-specific issues. We conclude that current OGC specifications provide a sound basis for developing service oriented architectures for geographic applications; however, in particular for geoprocessing applications, we question the
feasibility of the use of Web Feature Services as data sources for larger amounts of data and call for further research in this direction.

1 Introduction

Service Oriented Computing (SOC) represents a new generation Distributed Computing Platform (DCP) whose architectural model is called a Service Oriented Architecture (SOA) (Erl 2007). The OASIS Reference Model for SOA (OASIS 2006a) defines such an architecture as “a paradigm for organizing and utilizing distributed capabilities that may be under the control of different ownership domains” and sees services here as “the mechanism by which needs and capabilities are brought together”. A specific SOA thus provides the framework and rules for service description and discovery, interaction between service providers and consumers and the respective execution environment. Clearly, Web services and the protocols and mechanisms for their description (WSDL, W3C 2004), discovery (UDDI, OASIS 2004) and invocation (W3C 2003) are today’s most prominent SOA example.

The benefits are manifold. A SOA is an open and interoperable environment, which is based on reusability and standardized components. Application development in a SOA is focused on concrete applications (and thereby specific requirements and needs). In contrast to standard GIS applications, where normally only a small percentage of the functionalities in the software are used, applications based on SOA may provide users with just the functionality they need. As SOAs are loosely coupled service architectures, they provide data and processing capabilities required for a given processing activity not locally, but decentralized, i.e. close to the source of production. This means that inconsistency and outdateding in local copies and repositories are avoided and the integration of (distributed) real-time information becomes much easier. Furthermore, services providing algorithms can be reused by several different applications thus helping to avoid redundant implementations. In consequence, the SOA approach to system development can produce systems that can be flexibly adapted to changing requirements and technologies and are easier to maintain than standalone applications. However, the SOA paradigm has also downsides. First, changing from one paradigm to the other is a challenge as it requires new competencies and capacity building. Second, it requires an agreement on a common architecture and a standard development methodology in order to create applications based on a portfolio of services.

Also within the domain of geographic information (GI), emerging interoperability specifications – and here most prominently the work within the Open Geospatial Consortium (OGC) – follow the general SOA ideas (OGC 2004a, d; 2005c, d, e). This has created a technology evolution that moves from standalone GIS applications towards a more loosely coupled and distributed model based on self-contained, specialized, and interoperable GI services (ESA 2004, Nebert 2004). Through collections of technologies and organizational agreements, Spatial Data Infrastructures (SDIs) provide the framework for optimizing the creation, maintenance and distribution of GI services at different organization levels (e.g. regional, national, or global) and involve both public and private institutions (Nebert 2004). The political support given at high governmental levels to innovations like the U.S. National Spatial Data Infrastructure (White House 1994) or the INSPIRE (Infrastructure for Spatial Information in Europe) directive (European Commission 2007) have encouraged the development of SDIs.
At present, most SDI initiatives are still at an early stage in their development, just starting to offer geoportals that integrate on-line map viewers and search services for their data holdings (Bernard et al. 2005, European Commission 2005). In later development phases, they can also serve as frameworks for easy and flexible development of GI applications by providing standardized data access interfaces. In this paper, which is an extension of Friis-Christensen et al. (2006), we investigate how well the specifications, standards, and products that are available for SOA within the GI community today are suited to support developments of GI applications. In particular, we focus on how distributed geoprocessing can be achieved within a SOA.

A forest fire application prototype serves as a use case and illustrates the problems and challenges that exist in distributed geoprocessing in a SOA. The goal of the prototype is to enable users to assess fire damage areas based on land cover data in a given area. It should also allow users to search catalogues for available land cover data and select those required for damage area assessment. Although this use case is very simple and can be easily implemented in any standalone GIS client, the aim and scope of this work is to create a prototype that runs in a distributed and interoperable environment. Thus, the main goal of the use case is to exemplify the scalability, flexibility and reusability of the components in an SDI (Bernard et al. 2005). We use the prototype to discuss the drawbacks and disadvantages of the architecture of the implementation, which is based on standard OGC service types. Further, we propose alternative designs and architectures which may solve some of the problems found.

The remaining paper is structured as follows. Section 2 describes the use case which is the basis for the architecture discussions. Section 3 describes an architecture for distributed geoprocessing and presents its implementation based on OGC standards. Additionally, the drawbacks of the current approaches are discussed. Section 4, investigates alternative design approaches in order to improve performance and flexibility. Finally, in Section 5 we offer some conclusions and briefly outline future research topics.

2 Use Case

The use case concerns an application that calculates forest fire statistics based on land cover entities. The use case is depicted in Figure 1, and as illustrated, several data sets are foreseen in the application. The order of the workflow is indicated by numbers. First, two data sets are required in order to assist the user in selecting the area of interest (steps 1 and 2 in Figure 1):

- Image 2000 (Nunes de Lima V 2005), which can provide backdrop satellite images (LANDSAT 7 Enhanced Thematic Mapper ETM+ imagery) to the application.
- Place names used for locating a specific geographic area based on geographic name input.

For the actual calculation of forest fire statistics the following data sets are used (step 3):

- Data describing the assessed boundaries of burnt areas for various years. This information is distributed via the European Forest Fire Information System (EFFIS; http://inforest.jrc.it/effis).
- Natura 2000 (European Commission 2001) to describe ecological networks of special areas of conservation across the European Union and used here to assess the amount of protected areas that are affected by fire.
• Corine Land Cover 2000 (Nunes de Lima V 2005), as a thematic reference data set, used here to identify which land cover classes are affected when assessing forest fire damage.

The last step in the workflow is the actual calculation of the burnt area statistics (step 4). It consists of several processes, as indicated by the boxes in Figure 1. First, if necessary, the coordinates are transformed into an area-true coordinate reference system in order to be able to provide correct area statistics. Then the data are clipped according to an area of interest (a bounding box) selected by the user based on backdrop satellite images. The clipping is required in order for the geometries to exactly match the bounding box and calculate correct statistics. Based on the area of interest, the thematic data set chosen by the user is intersected by the burnt areas and, finally, the statistics can be calculated by summarizing all intersected areas. These statistics can be calculated either for the thematic data set as a whole or for each class of a specific classification. For example, a user could be interested in how each of the Corine Land Cover classes (e.g. urban fabric or arable land) is affected by fire. For data that does not have several thematic classes (like Natura 2000) this is not necessary.

3 An Architecture for Distributed Geoprocessing

In this section we present a proposal for a specific architecture for the distributed application described in the previous section. It is based on current OGC service specifications.

3.1 Architecture

An overview of the components in the architecture is depicted in Figure 2.

In addition to the data access services, the application comprises a catalogue service, which is necessary in order for the user to discover which data could be used for the
area statistics assessment. Currently, only two data sets (Corine Land Cover and Natura 2000) are used for calculating the fire statistics, however, other data sets (e.g. administrative areas) could be used if required by the user. In situations where several data sets and sources are unknown to the user, the catalogue becomes pertinent. The different steps in the calculation of the area statistics (see Figure 1, step 4) are collected in one geoprocessing service because of practical reasons such as simpler application development and assumed higher performance. However, this limits flexibility as each processing step is not implemented as a separate operation (or service) and, thus, cannot be reused in another application independently. As a consequence, the service becomes highly specific towards one application (calculating area statistics) with a fixed workflow and without the possibility of, e.g. reusing the data clipping and intersection operations for other applications. We will discuss this aspect in section 4. Finally, there is a software client which provides the user interface and executes the workflow of the application.

To illustrate the interaction and communication flow a sequence diagram of an application scenario is depicted in Figure 3 (note that for simplicity the gazetteer service has been left out of this example).

The client requests the backdrop image, supplied by a fixed Web Map Service, for visualization of an initial view. From this initial view, the user selects the area and time of interest in order to get parameters for selecting appropriate data. The Image 2000 data showing the area of interest is visualized. In principle, the user could also select an arbitrary background image based on a catalogue search. However, for simplicity, we do not include this possibility here. Subsequently, the catalogue is used to search and select those data used as source data (or mask) and target data. In the scenario, the mask is a specific layer of burnt areas and the target could be Natura 2000 or Corine Land Cover data. After the selection of data the area statistics service is invoked. The service requests the selected data, transforms their coordinates into projected coordinates (if necessary) and calculates the area statistics. These are then returned and visualized in the client as a table.
3.2 Implementation

In this section the various components in the architecture are presented. More specifically the statistics service, the mapping and feature services, the catalogue service and client, and the forest fire client.

3.2.1 Statistics service

The interface of the statistics service follows the Web Processing Service (WPS) specification discussion paper (version 0.3.0) which is continuously evolving (OGC 2005e; at time of the publication of this paper, it has been approved as an official OpenGIS specification Web Processing Service, version 1.0.0 and the changes made to the specification do not affect the contents of this paper). It is implemented in Java 1.5 using the Geotools 2.1 API (Geotools 2005). A conceptual model of the service and its interface is shown in Figure 4. The model is simplified and does not show detailed implementation aspects. The OGC WPS specification specifies three operations as mandatory: getCapabilities, describeProcess, and execute. The getCapabilities operation (which is common for all OGC web services) simply allows clients to retrieve service metadata from the service. The describeProcess describes the process(ing) that is supported
by a specific WPS. This operation is not implemented in the current version of the statistics service. A process offered by a WPS can be called via the execute operation, which carries out the specific operation requested.

Two types of requests are supported by WPS: key valued pair (KVP – Get) and XML (Post) requests. The Get request is a plain html request with all parameters specified. The Post request is a submitted XML document including all parameters in XML tags. The execute operation takes both Get and Post requests; however, we only describe the Get request here. As seen in Figure 4 there are several parameters to the request, which need to be passed to the statistics service. First, the service and processname have to be specified. The service is a WPS and it only supports one process: AreaStatistics. The mask is the URL for the mask data and masktypename is the feature type name. The same parameters hold for target data. The bbox is the bounding box in which the statistics need to be calculated. The attribute parameter specifies the specific attribute to be used, if there is a need to give statistics per thematic class. The totalarea parameter specifies if the total burnt area should be given (default is true).

An example of a Get request is (line breaks are added to improve readability):

http://naturegis.h07.jrc.it:8090/StatisticsService/Process?
SERVICE=WPS&

Figure 4 A simplified model of the statistics service
When a request is made to the statistics service, it calls the execute operation in the StatisticLauncher, which uses the parameters from the request. In order to receive the features in the chosen area of interest (the bounding box) a getFeatures from a WFSReader is launched. Here it is determined from a getCapabilities request to the WFS which coordinate reference system is provided. At the moment our WFSs only distribute data in a geographic coordinate system and we transform the coordinates in order to get an area true coordinate reference system. Finally, the statistics are calculated and then an XML document including the statistics is returned from the service.

3.2.2 Mapping and feature services

As mapping and feature services we use standard implementations of OGC WMS 1.3 and WFS 1.0 specifications. We use the open source GeoServer version 1.3.0 (http://www.geoserver.org) as a WFS and ArcIMS 9.1 with a WMS connector, which provides the backdrop and satellite images. For the gazetteer Ionic Software RedSpider Studio provides a simple mechanism to build a gazetteer service on top of an existing WFS service. The WFS used is available at the Ionic Software website (http://webservices.ionicsoft.com/gazetteer/wfs/GNS_GAZ).

3.2.3 Catalogue

As Catalogue service we use con terra (http://www.conterra.de) terraCatalog, which is an implementation of the OGC Web Catalogue Service (OGC 2004a) specification and makes it possible to store and retrieve information about spatial data and services. In particular, this implementation supports the ISO 19115/19119 profile for CSW 2.0 catalogue services (OGC 2005a). In order to access the catalogue from the forest fire client we used the standard catalogue interface to access metadata stored in the catalogue. A simplified model of the client is shown in Figure 5. What the client basically offers is a search operation which takes title, bounding box, and time of interest (year) as parameters. The client supports two different protocol bindings using HTTP as transport mechanism; the Z39.50 protocol binding and the Catalogue Services for the Web (CSW). For the communication with the terraCatalog we use the CSWCatalog-Client, which implements the CSW protocol binding. The title of the data set and a service URL is returned for the catalogue and then the preferred data set can be selected in the forest fire client. The service URL is used as parameter for a request to the statistics server.
3.2.4 Forest fire client application

The client was built using Dynamic HTML (DHTML) and the RedSpider Studio 3 which is a geospatial portal development solution for distributed OGC web services. More specifically, the JSP ‘geotag’ library allows for easy access to remote services which implement OGC specifications. Response times for user requests were reduced by utilizing the Asynchronous JavaScript and XML (AJAX) scripting technique. This allows users to call the catalogue and statistics services and work with the results without requiring a full application refresh. As depicted on the screenshot in Figure 6 users can zoom and pan or locate an area via a gazetteer service. Then, after selecting a...
year and keywords for searching burnt areas (only this is shown) and target data, a
damage area statistics report is generated (the bottom part in Figure 6).

3.3 Issues in the Presented Architecture

The example described above illustrates that the standards of OGC provide a possible
basis for implementing distributed geoprocessing within an SDI. With standardized
interfaces, it becomes possible to easily combine several services for providing,
processing and visualizing data. However, while this makes geoprocessing in SDIs flex-
ible, there are a number of fundamental performance issues stemming from technical
limitations and the architectural design that constrain the usability and scalability of the
presented architecture. In particular an architectural design based on distributed data
sources is problematic.

The main problem we encountered in our implementation is the transport of data
within a service chain, which becomes necessary if the data and the processing facilities
are provided on different nodes of the SDI. The ISO specification 19119 (ISO 2005),
defines a service chain as a sequence of processing services where, for each adjacent pair
of services, occurrence of the first action is necessary for the occurrence of the second
action. In current OGC-standards-based SDIs, the data being transported in a service
chain is usually encoded as GML or as a binary coverage (e.g. a GeoTIFF). A typical
GML file for CORINE Land Cover data for Portugal and Spain is > 1 GB uncompressed
(approximately 1/3 size when compressed), the corresponding Grid Coverage in, e.g.
GeoTiff format is > 350 MB, with 1 band representing one attribute using 8 bit
encoding and 100 \times 100 m spatial resolution. Obviously, the retrieval of such amounts
of data from a WFS or WCS instance and its transport over the SDI network could
easily take longer than the client may be willing to wait.

Another issue is related to the flexibility of a service-oriented geoprocessing
architecture. In the presented architecture we combine several distinct processes in one
going processing service operation in order to make our implementation easier and
improve performance as only one request to a geoprocessing service has to be made
instead of several. The problem of an explicit combination is that the service becomes
application specific and its reusability decreases. A more flexible solution would be to
implement a distinct service operation for each distinct process that is needed.

This leads to some aspects concerning the architectural design that needs further
investigation:

• In the presented architecture, we have assumed a synchronous communication pattern
for the calls to the WPS. Should the geoprocessing operations, that can take considerable
time to complete, instead be offered via asynchronous calls? What does asynchronous
communication imply for the architecture – what additional components and interaction
steps are required? These questions will be addressed in Section 4.1.

• How can geoprocessing in SDIs be made more flexible and efficient? In Section 4.2,
we present approaches to increase flexibility and efficiency. First, we discuss the
coupling of several processing operations within a single processing service instance.
This is done in order to keep processes as distinct operations and still achieve the
performance advantages of combining several processes into one. Second, we present
an approach to tightly couple data and geoprocessing operations, which can improve
performance.
4 Alternative Design Approaches

As described in the previous section we encountered several problems with the proposed architecture. In this section we present alternative design approaches which address some of the problems identified.

4.1 Using Asynchronous Communication

Depending on the file size, processing of spatial data may take a considerable amount of time. Users of desktop GIS are aware of this and cope with it. The resulting actions are batch and overnight processing, and coffee breaks. When large processing tasks are realised over a network, data transportation and the parsing of the spatial data used for the statistics calculation can dramatically increase the response time. This contradicts expectations and the ‘interest-curve’ of Internet users who, other than Desktop GIS users, might expect a result within a few seconds (Nah 2004). But even if the Internet user is patient, one of the web servers involved in a given request might time out because a response takes longer than it is configured to wait.

A solution to this is the use of asynchronous messaging for time-consuming requests. In asynchronous messaging the process response does not return immediately but some time later in a different communication session. This means that the user need not wait at the browser until the service is finished processing a request, rather the preparation of the response is undertaken offline, and the user can retrieve the results once the process has finished or failed. This creates the following requirements:

- Each service call has to be equipped with a unique identifier in order to allow the user to later retrieve the results that were generated for this call.
- The service requester should be able to find out about the process status and, in case the process has completed, where to pick up the result.
- The service requester should be given the opportunity to abort the process.

The latter two points cross the border between asynchronous messaging and asynchronous services. The interface of an asynchronous service allows the service requester to interact with the service while processing, asking about status and aborting or modifying an operation (OASIS 2005).

The strategy of asynchronous messaging in SDIs is discussed in several papers and standardization approaches. For the existing approaches, we can in principle point to a pull mechanism, requiring the service requester to check the status of the process, and a push mechanism, requiring the service provider to give updates about the status of the process. The asynchronous messaging as foreseen in the OGC Web Processing Service Specification is an example of the pull mechanism; an alternative interaction pattern proposed by the OGC Web Notification Service Specification (OGC 2006) follows the push mechanism. In the following we illustrate the usage of these two approaches within the given use case, starting with the pull mechanism.

The OGC Web Processing Service (OGC WPS) Specification, in version 0.4.0 (OGC 2005f), has a built-in mechanism for asynchronous messaging. It foresees that the user of a WPS instance is informed about the possibilities of asynchronous messaging through the statusSupported attribute in the process description. If statusSupported is “true”, the Execute operation request for this specific process may be called asynchronously by the service requester.
Figure 7 shows how the processing workflow of the forest fire use case would change if the processing was implemented through an asynchronous WPS call using a pull approach. The WPS responds immediately to the execute request with an acknowledgement. The returned XML document contains a URL pointing to a constantly updated Execute response document. As long as the process is not completed, the Execute response document contains the status and a measure of the amount of processing time remaining. Thus, if the client tries to retrieve the URL while the statistics calculation is still running, the response document only contains the status. Once the process is completed the status URL contains the final Execute response including the results. This approach is more convenient for the user as the status can be monitored.

The push approach to asynchronous operation calls, which uses notification, is illustrated in Figure 8. It has been inspired by the Web Notification Service (WNS) best practices paper (OGC 2006) which describes a notification mechanism for sensor webs. The WNS is an asynchronous and general purpose messaging service. It allows a user to notify a client of the occurrence of an event. The WNS requires a user registration, which includes user information (i.e. the notifying user) and notification target (i.e. the notified client). A notifying user can be either a client application or another web service. In principle, the WNS could be used as a standalone service in a notification scenario. In contrast, we present a simpler solution here, where its core functionalities (user registration and notification) are integrated into the processing service and which therefore does not require additional messages to be passed between the processing service and the WNS in order to do the notification. The workflow for this scenario is shown in...
Figure 8. First, the client registers with the StatisticsService providing, among other details, the preferred notification channel. The StatisticsService returns a registration ID that the client uses for subsequent asynchronous calls to the StatisticsService Execute operation. Once the process is completed the StatisticsService sends a notification to the client including a URL with a reference to the result data. Note that both the Execute call and the notification are one-way, and that no status request is foreseen.

Comparing the two approaches, a number of issues should be discussed. Firstly, a general messaging policy has to be defined within an architectural framework that specifies whether a pull or a push approach or a combination thereof is used for asynchronous communication. So far the pull model as described in the first example is more suited for machine-machine communication as there are no notifications to the user. Rather, a service requester could automatically request a status with a given interval in order to monitor the status. However, the approach would benefit from a more sophisticated policy on communicating the process status to avoid the need for the service requester to frequently check the status. The push model as described with the WNS mechanism is more useful for alerting (human) users by supporting emails or Short Message Services (SMS). However, this approach does not yet allow aborting a process.

Second, someone must determine if the messaging of a specific service operation call is to be asynchronous. The two approaches presented in Figures 7 and 8 show two different ways of handling this. In the case of the WPS, the service provider in the first instance decides whether the operation may be called asynchronously. However, it is the service requester in the end who (that) decides about the type of messaging. In the second approach, the service provider may omit this and only allow asynchronous calls.
A third issue is closely related to this, namely the inclusion of asynchronous messaging in service chains. A number of questions need to be answered: Does one asynchronous call in a chain make the whole chain asynchronous? How is an asynchronous service call integrated in Ad-Hoc chaining? Which strategies exist? These questions will be addressed in future research.

A fourth issue is about the way results of asynchronous calls are handled. Besides the policies for web services notification (OASIS 2006b) and web services coordination (OASIS 2007b), no stringent policy exists on how to deal with data references as the URL pointing to the result data set(s) of an asynchronous service call. An enriched policy needs to define, for example, the time this reference remains valid as well as the way it deals with access rights or property rights. Clearly this last issue links again with the way service chaining could be enabled and does not only refer to asynchronous messaging but, in general, to the way data is shipped between various processing steps, as discussed in the next section.

4.2 Increasing Efficiency of Processing Services

It is a common practice in GIS development to combine elementary operations into more complex tools in order to address specific user requirements, see for example the ArcGIS ModelBuilder approach (ESRI 2007). This decreases development time and increases flexibility when facing changing user requirements. When applying this approach to the given use case, the calculateStatistics operation can be split into the four different elementary operations described in Figure 1: coordinate transformation, clip, intersect, and calculate statistics. Here we first focus on the three latter operations. We provide all three operations as independent WPS processes and combine these into a service chain to meet the use case requirements. Three types of service chaining are defined in ISO (2005):

- Transparent or user defined chaining, where the human user manages the workflow.
- Translucent or workflow-managed chaining, where the human user invokes a Workflow Management service that controls the chain, while the user is aware of the individual steps.
- Opaque chaining, where the user invokes an aggregated service that carries out the chain. The user has no awareness of the individual steps.

For the discussion on increasing processing efficiency we focus on transparent and translucent chaining. Transparent chaining can be implemented using the WPS Specification in its current form: Figure 9 shows an application for the use case at hand by means of transparent chaining.

One WPS instance provides each of the required elementary operations. In the given example, the client is a thick client that interacts with the WPS instance, i.e. it prepares the requests and manages the execution of the operations in the correct order to achieve the intended result. Though this approach offers the desired high flexibility of operation combination, it requires continuous interaction with the client application. Furthermore, it results in a repeated sending of input data to the same WPS to execute several related operations, which increases the time required for achieving the final result. In case this approach is combined with the asynchronous messaging as described in Figures 7 and 8, the workflow becomes even more complex and lengthy.
Some of these issues can be eliminated through the introduction of a *translucent* chaining approach for the WPS. We suggest allowing the user to define a workflow that describes a chain of two or more processes of the same WPS Instance. This workflow is sent to the WPS instance in a single execute request. As in the case of the transparent chaining, the output of one process shall be usable as an input for the successive processes. The WPS execute-request concept in its current form allows the specification of only one process per request. Thus, the following adaptations of the WPS specification would be required in order to allow the translucent chaining:

- Extending the WPS execute concept and syntax in order to allow the ordered execution of more than one process.
• Allowing the outputs of one process as inputs to one or more succeeding processes.

A strategy has to be developed in order to define outputs as intermediate or final to
the workflow.

If these adaptations of the WPS interface were to be made, the workflow could change
as shown in Figure 10. Only one execute request is included, which specifies the processes
to be executed and their order of execution. The steps for retrieving data and sending
execute responses after every operation are omitted. This approach could be combined
with any of the two asynchronous messaging approaches described earlier without
increasing the complexity.

In this workflow, the user defines the order of the processing steps by defining the
output of one operation as the input of the next. Clearly, it could also be envisioned
that the processing service instance tries to optimize the way the requested operations
are processed such that, for example, an operation that reduces large data amounts
should be executed first in a service chain in order to reduce data to be processed by
subsequent operations. This concept of optimization is similar to a query optimization
in a database (Jarke and Koch 1984).

Another approach to be investigated is the processing at the source of data, which
we also term tightly coupled geoprocessing. It could increase the performance of the
geoprocessing, and be combined with the approach presented earlier in this section. The
WFS Filter allows for simple geoprocessing and is, thus a first step towards tightly
coupled geoprocessing. According to the specification (OGC 2005d), it is possible to
query a WFS using topological relations and/or to apply arithmetic operators (addition,

![Figure 10](image-url)  
**Figure 10** Chaining several atomic WPS operations in one WPS instance (using translucent chaining)
subtraction, multiplication, division) or even arbitrary functions that can accept zero or more arguments as input and generate a single result (OGC 2005b). Figure 11 illustrates an example where tightly coupled geoprocessing and data access might benefit our use case. In the example a new service type instance, a processing WFS (here called a WFS-P), is responsible for coordinate transformations and the clipping of data.

For the coordinate transformation the `getFeatures` request specifies which coordinate reference system is required. The coordinate transformation is already possible using the existing OGC WFS specification without changing the interface. As a further possibility in the request, the example in Figure 11 allows one to specify that a bounding box (or even irregular polygon) clipping should occur. As described in a previous section, the normal WFS bounding box parameter would just create an intersection and, thus, retrieve all polygons overlapping the bounding box without a clipping.

Tightly coupled geoprocessing is expected to decrease the size of the requested data and additionally the processing load for an eventual later processing accomplished by other processing services. An example of a processing at source which would substantially reduce the data being transported is server side generalization of data as described by Lehto and Sarjakoski (2005). The disadvantage of the approach is that changes may occur in an already standardized interface such as the described clipping possibility, which would be an extension of a standard WFS. This is a problem since a SOA should have common standardized interfaces in order to have well-known service types. Well-known service types are essential when building distributed service applications.

5 Conclusions and Future Work

In this paper we have reported on the development of an application that enables the assessment of fire damage areas based on land cover data in a given area. For the application, we have used and implemented important components in an SDI based on existing standards and specifications from OGC. As stated in the introduction, most SDI initiatives are still in an initial state and just starting to offer geoportals that integrate...
on-line map viewers and search services for their data. In this paper, we have demonstrated that beyond this initial step, SDIs can also be used to develop applications, which solve problems in a more flexible manner than standalone applications. Additionally, by building the prototype using software from various vendors, we have illustrated how interoperability can be achieved.

Our work showed that the standards of OGC provide a functional base for interoperability among services within an SDI. The discussion paper of the Web Processing Service provides a draft specification that supports the implementation of distributed, interoperable geoprocessing services. We have shown that catalogues providing metadata for data and services are a backbone for an application involving distributed data sources and geoprocessing services – not only for discovery but also for invocation.

In addition, we presented the considerable benefits of a distributed geoprocessing environment. For example, the data can possibly be retrieved directly from the data creation source and do not have to be replicated, e.g. to a local machine/server. Furthermore, the developed geoprocessing service can be reused for other applications. However, as also shown by the implementation of the use case, there are fundamental performance issues stemming from technical limitations and the architectural design. Mainly, these issues originate from an architectural design based on distributed data access and passing data among services. The implemented architecture, which is based on synchronous communication, is not suited for applications that require transport and calculations of large amounts of data. This underlines the necessity to clearly identify the intended use and the requirements in the planning of a distributed geoprocessing architecture. We presented several alternative approaches, each of which may improve the application’s performance and flexibility.

Further research should cover additional architectural approaches: e.g. the distribution of algorithms instead of data, assuming the data sources are capable of processing these algorithms. Another example is the separation of geometry and attribute data: Geometry information, though not required for a large number of processing operations (like classification and attribute normalisation) is dragged along as information ballast slowing down the performance of applications. Examples for specifications looking into this issue are the related OGC discussion papers on the Geolinking Service (OGC 2004c) and the Geolinked Data Access Service (OGC 2004b).

Finally, future research should try to synchronize and streamline today’s GI specifications with the achievements of other SOA initiatives, and here most prominently the general Web service standards WSDL, SOAP, UDDI and WS-BPEL (OASIS 2007a). WSDL could be used as an alternative to the WPS’s describeProcess operation for describing geoprocessing operations. Using these descriptions, the services could then be published in a service registry like UDDI and invoked using SOAP. Using orchestration languages like WS-BPEL and corresponding workflow engines (rather than hard-coding the service interaction in the client application) would facilitate the opaque chaining defined by ISO (2005).

References

© 2007 The Authors. Journal compilation © 2007 Blackwell Publishing Ltd
Transactions in GIS, 2007, 11(6)
Nah F 2004 A study on tolerable waiting time: How long are Web users willing to wait? Behavior and Information Technology 23: 153–63
OGC 2004a Catalogue Services Specification v2.0. Wayland, MA, Open Geospatial Consortium Report No 04-021r2
OGC 2004b Geolinked Data Access Service v0.9.1. Wayland, MA, Open Geospatial Consortium Report No 04-010r1
OGC 2004c Geolinking Service v0.9.1. Wayland, MA, Open Geospatial Consortium Report No 04-011r1
OGC 2004d Web Map Service Specification v1.3. Wayland, MA, Open Geospatial Consortium Report No 04-024
OGC 2005e Web Processing Service v0.3.0. Wayland, MA, Open Geospatial Consortium Report No 05-007r3
OGC 2005f Web Processing Service v0.4.0. Wayland, MA, Open Geospatial Consortium Report No 05-007r4
OGC 2006 Web Notification Service v0.0.9. Wayland, MA, Open Geospatial Consortium Report No 06-095