Generalization Services on the Web – A Classification and an Initial Prototype Implementation

Dirk Burghardt, Moritz Neun and Robert Weibel

GIS Division, Department of Geography, University of Zurich
Winterthurerstrasse 190, 8057 Zurich (Switzerland), Fax: +41-1-635 6848
Email: {burg,neun,weibel}@geo.unizh.ch

Abstract Much progress been made in the field of web based cartography through standards developed by the Open Geospatial Consortium (OGC). While automated access and presentation of cartographic data are defined, services for automated generalization are not yet standardized. This paper aims to show advantages of applying the service concept to generalization and suggests several classification schemas of Generalization Services at different levels of granularity. There follows a detailed explanation of a real implemented Generalization Service. It is shown how software developers can make their generalization functionality available as a service and how these services can be accessed dynamically. For the implementation, the open source Java Unified Mapping Platform (JUMP) was extended to work as a framework for generalization. Generalization Services could be used in different application scenarios, for instance as a middleware component between a Web Feature Service (WFS) and a Web Map Service (WMS), to allow on-the-fly adaptive zooming, or as an interactive Generalization Services for the production of topographic maps by national mapping agencies (NMA). They may also allow the development of a common research platform, where researchers would have access to a common generalization framework.

1 Motivation

In recent years cartography has evolved in a new direction with the application of web technologies and services. As a result of this process maps in the web context are generally are generated on-demand and on-the-fly, containing more specific and tailor-made information. This contrasts with the traditional way of map making which focused on the production of static maps that were designed in advance for general purpose usage (with the prototypical example of topographic maps). Web cartography can benefit from the standards developed by the Open Geospatial Consortium (OGC) to implement web services for feature access (WFS) and web mapping (WMS). Obviously, however, the demand for dynamic web map generation has also lead to increased requirements on automated map generalization procedures. Not surprisingly therefore, the OGC has also expressed interest into “feature generalization services” as part of their OGC Web Services (OWS) initiative (OGC and ISO, 2002). However, the specification process for such services has not experienced much progress so far. From another end, the map generalization research community has started to develop an interest into Generalization Services as well, driven by the desire to develop a common open research platform that would allow testing and sharing of generalization algorithms (Badard and Braun, 2003; Edwardes et. al, 2003). This has been evidenced

through discussions at the meetings of the ICA Commission on Map Generalization and Multiple Representation in Paris 2003 and Leicester 2004.

There are several advantages to using Generalization Services in a collaborative and distributed research environment as well as for on-demand map production. First of all, the platform independence makes the development independent from the operating system and the hardware used, which also offers application in a mobile context. Secondly, the service can be integrated in any software platforms, such as web browsers, GIS or map production software. Last, the service can be accessed over the internet or on the local machine, the latter however only if the code of the underlying generalization operations is made available.

The objectives of the proposed paper are two-fold: 1) to present a classification of Generalization Services, and 2) to report on a prototype implementations of sample Generalization Services, as well as on experiments that were carried out to assess the feasibility of the approach.

2 Web Services for Spatial Applications and Generalization

2.1 Web Services

In order to enable computers directly to access distributed data and to use services, the concept of Web Services has been introduced. Software programs – frankly computers – can read web pages (mostly in HTML), but they can't understand them. Human beings are able to read a web page and find the link or the button to click on. To enable computers to do this without the help of a human user, *Web Services* make use of machine-processable interface descriptors and of a standardized language:

> *A Web Service is a software system designed to support interoperable machine-to-machine interaction over a network. It has an interface described in a machine-processable format (specifically WSDL). Other systems interact with the Web Service in a manner prescribed by its description using SOAP messages, typically conveyed using HTTP with an XML serialization in conjunction with other Web-related standards. [W3C (2004)]*

The usage of such a Web Service can be either fully automatic within e.g. GIS applications or the Web Service could be called from the application upon the demand of the user.

2.2 Spatial Web Services

Human-computer Interaction Service
Most Spatial Web Services are understood as data delivery services. Usually they connect to a geographic database and retrieve information from a database upon request. Examples are catalog and gazetteer services for data search or Web Mapping Services (WMS) for the visualization of spatial information (Fitzke et. al, 2004). Common to this kind of services is that they are designed for end use by humans. Therefore this category is commonly called human-computer interaction services. The approach represents the view of the Open Geospatial Consortium who have a primary focus on the end user. The aim is to simplify the exploitation of spatial data and the access

to geo-information by the user at the end of a chain of spatial services. Very often Web-based Services that are connected to a database can also perform operations upon the data. Examples of such services include a Web Feature Service (WFS), Web Coverage Service (WCS) or a route planning service.

Computer-computer Interaction Service
A second main category is pure processing services which receive data and parameters from the requesting clients and perform operations on them. Possible reasons for this type of services include, for example, the availability of the algorithm only on the server due to licensing or platform incompatibilities, or also the need of faster calculation power on a super-computer. This service category reflects the original purpose of Web Services as given in the above definition, which includes the interoperable interaction between distributed applications, the prerequisite being a complete automation of processes.

2.3 Application of Service Concepts to Generalization

In general two concepts of Generalization Services seem to have a promising range of application. The one and potentially most widely used service would be the generalization or better the adaptive zooming in Web Map Services. Providing a zoom function is classically the domain of Multi-Resolution Databases (MRDB). A Generalization Service could be introduced as an add-on to a WMS or as a **middleware** between a WMS and the client which produces the desired resolution out of the available data. This kind of service belongs to the second category of computer-computer interaction service, which requires fully automated solutions.

The other promising service would be more interactive Generalization Services for GIS users and for map production in organizations such as national mapping agencies (NMA). In this case the Generalization Service would provide its functionality and its calculation power to the service subscribers. It also fulfills the requirements of a **common research platform** (Edwardes and Burghardt, 2005), where the researchers want to have access to a common generalization framework. Advantages of such a **standalone service** would include, for example, the ability to provide specialized or novel algorithms to the research community without everybody having to adapt their systems to the specific needs. Furthermore, the possibility exists to write specific algorithms for special computer architectures e.g. clusters, grids or other parallel processing systems and offering this service to the subscribers.

Figure 1 illustrates the two types of Generalization Services. In the case of zooming only configuration parameters such as the resolution and the bounding box are sent from the client to the Generalization Service. In the case of a specialized Generalization Service the data would also have to be sent. In the first case the service knows exactly the type and structure of the data (Lehto and Sarjakoski, 2004; Illert and Afflerbach, 2004). In the second case this is a major problem to handle. Only standardized and valid data can be handled by the Generalization Service.

Additionally, the kind of interaction with such services differs. While the Generalization Service runs completely automatically in the first instance, the user has (i.e. wants to have) more control in the second, interactive scenario. These different usage scenarios will be outlined in detail in section 4.

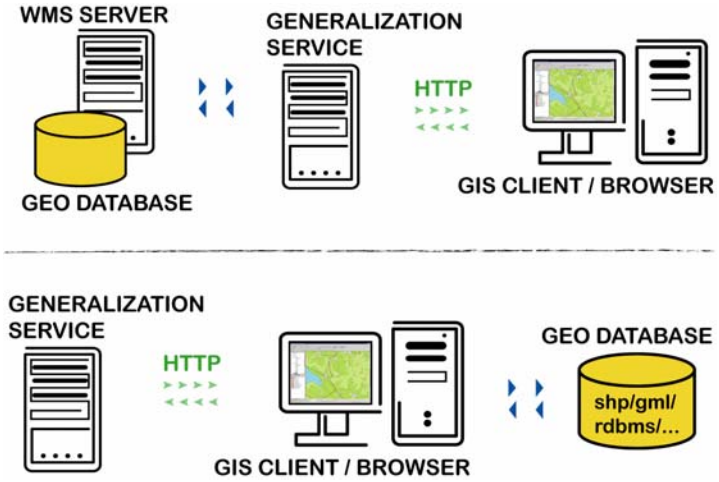

Figure 1: Two types of Generalization Service: As a middleware (above) or as research platform (below).

3 Characteristics for Generalization Services on the Web

There are several characteristics for Generalization Services using general concepts of Web Services. First, the conceptual and implementation characteristics will be outlined. It is shown what a service oriented architecture is and which techniques are available to offer Generalization Services to service consumers, such as human users or cartographic software applications.

3.1 Service-Concept and Component Architecture

A **service** is an abstract **resource** that represents a capability of performing various tasks. This abstract service has to be realized by a concrete agent. Different services can be connected with a request based communication. This **service oriented architecture** approach offers the resources to other users in a network as independent services. Service access is achieved through a standardized approach. These loosely coupled services offer more flexibility than other system architectures. In essence, service-oriented architectures represent collections of services communicating with each other. Communication can be either simple data passing or it could also be two or more services jointly coordinating some activity. A service represents the endpoint of a connection. A service has an underlying computer system that manages the client-server connection. Service invocation is done by a **service request** to the invokable interface of the service. Upon successful operation, the service provider returns a **service response**. Errors have to be handled with **service exceptions**. These interactions are independent and interfaces and protocols yield the underlying infrastructure. The **Web Services** technology offers these capabilities for service communication and interfacing.

Services can be used by higher level services and can themselves use the functionality of others. This is achieved through the ***n*-tier distribution** capabilities of Web Services. This structure implies the client/server program model. The processing of a specific application occurs over *n* machines across a network. These tiers usually include a data tier, business logic tier, and a presentation tier. A given machine will perform the specific tasks of a tier. Sometimes multiple tiers can also reside on one machine but this usually is only used for testing purposes. In that case the remote architecture remains unchanged. *N*-tier applications have not only the advantages of distrib-

uting computing but additionally any one tier can run on an appropriate processor or operating system platform and every tier can be updated independently.

The **component architecture** emerges from object-oriented and internet technologies. The systems components in a component-based architecture have generic interfaces through which they advertise their functionalities. This enables the dynamic loading of the components. Components, software objects encapsulating a set of functionalities, interact with other components. Every component has an interface which conforms to a defined architecture.

3.2 Conceptual and Technical Characteristics for Web Services in General

Every Web Service is a resource. The Web Service architecture implements a service oriented architecture using web technologies. The following main characteristics can be defined for all Web Services:

- platform independence
- service registry
- Web API - interface
- loosely coupled communication

The **platform independence** of a Web Service is a major advantage. This feature, also referred to as interoperability, enables a Web Service to be really distributed over many different platforms and distinguishes Web Services from other technologies such as CORBA or Java RMI. When browsing the web and accessing services with a browser users expect to be able to see and use a web page on any platform or operating system. They don't want to care whether they are using a Windows, Macintosh or UNIX/Linux machine. Web pages use a common set of standardized protocols and languages. To achieve this interoperability equally for Web Services already existing standards have been chosen. Web protocols ensure easy integration of heterogeneous environments. The **protocol HTTP** and the **language XML** are available for every major platform.

In order to announce the availability of a Web Service and to find Web Services a registry is needed. The **"publish-find-bind"** principle describes this functionality. Web Services are registered (*publish*) and can be located through a Web Service **registry**. Service consumers can *find* suitable Web Services through a registry. These service consumers may be humans or other applications. The registry offers a single point access which then gives the exact service endpoint of the service's implementation to the service consumer (*bind*).

The **Web API** is the **interface** of a service which can be called from other programs. This interface is a standards based application-to-application programming interface which can be invoked from nearly any type of program. In order to enable programs to bind an interface automatically, the capabilities of an interface are shown through self-description. This usually is achieved through an interface description language (i.e. WSDL). The binding of a Web Service specifies the protocol and data format used for transmitting messages to the invoked interface.

Web Services are **loosely coupled**. The systems pass XML messages, usually over the HTTP protocol, to each other via their Web API. The Web API interface is the abstraction layer which yields the real communication and makes the connections stable and flexible. The protocol HTTP and the language XML can transport nearly any type of data. For instance binary arrays can be converted into ASCII and then packed into an XML document. This approach makes the Web Service concept very open and usable for many different purposes.

3.3 Advanced Characteristics for Generalization Services

Depending on the usage concept, either as a standalone service (e.g. research platform) or as middleware, different requirements of service registry and service invocation exist. The middleware concept often needs no registry because mostly it forms a combined system together with the data source (e.g. WMS). So, in this case the middleware would form some kind of service endpoint for the data source. In such a case the Generalization Service is bundled with the data access. The success of Generalization Services as a common research platform or standalone Generalization Service, however, is very much dependent on the existence of some kind of **service registry** (cf. section 5.1). The service registry has to be the single point access to all available services. These "Yellow Pages" for Generalization Services must know where the interface description and the service endpoints of the desired service can be found. The concrete architecture of such a registry will be described later in this paper.

Like the service registry, the **service invocation** also depends on the usage concept. A middleware concept for the access to generalized data would only need to transfer parameters to the service while a research platform or a standalone Generalization Service also needs functionality to upload data. All these concepts can also have different ways of interaction. This can be the classical human-computer interaction which is based on forms (in web pages) or on application plug-ins. The computer-computer interaction could also be fully transparent to the user or be hidden in a cartographic system without any control or interaction of the user.

In the case of a Generalization Service which supports the upload and treatment of the individual users' data there are special requirements for the **interface** and the **encoding of geo-features** with their geometry. In a form-based web page environment with full human interaction transfer formats such as Shapefiles or GML would be suitable. But they have to respect the format needed by the service. When using a plug-in in a cartographic application, the communication logic of the plug-in has to translate the application's internal concept of geo-features to a common format which can be understood by the service (cf. section 5.2).

Generalization of more than one feature and complex Generalization Services need two additional features which are not built into Web Services. These features are the capability to maintain the program's **execution state** and the support for **atomic transactions**. Maintaining the state of the execution of an algorithm is very important in the case when client interaction with the service at run time is desired. Once the client has uploaded the initial parameters (and possibly data) the service starts the execution of the algorithm. If the algorithm needs additional data, parameters or user input, a response with the corresponding request is sent back to the client. While this communication takes place the service has to maintain the already entered information, data and the already calculated changes. As the Web Services are based on loosely coupled state-less communication this has to be accomplished additionally. Technologies for this purpose such as session management via session IDs or cookies are usually already built into many Web Servers. The notion of transactions is fully compatible with the transaction concept in database systems. While executing a complex algorithm on multiple features in one data layer the data set is only changed if no error occurs. Otherwise a roll-back is done and no changes are committed. This feature is important to maintain the original data set's consistency in generalization.

The maintenance of state and the concept of transactions become extremely important when advanced **service control** and especially the **chaining** of several services is desired. The chaining of different services with their algorithms is one important step towards a "generalizeMap()" opera-

tion which generalizes an entire map and delivers a more or less ready-to-use product. For the use in middleware systems with the objective of fast display of generalized map objects the steps of the processing chain would be fully opaque. The quality in this case is less important than speed and stability. In the case of map production and research service chaining should also include interaction with the user in order to obtain high quality results. In this case usually the usage concept would be implemented with a plug-in because a form-based solution (e.g. in a browser) does not offer the flexibility and the support for long-time connections.

4 Categories for Generalization Services

4.1 Categorization Based on OGC/ISO Architecture Model (02-025)

As discussed above the distinction of spatial services according to the involvement of humans has lead to the categories of *human-computer* and *computer-computer interaction services*. The OGC/ISO architecture model refines these two main service categories with the following subdivision:

- Processing services
- Model/information management services
- Workflow/task management services
- Human interaction services
- Communication services
- System management services

Services belonging to these different categories can be applied usefully to the generalization process, as we will now show. *Processing services* encompass services that perform large-scale computations as they are needed when a generalization process is carried out fully automatically, for instance, on a complete map image or on generalization sub-tasks which can be completely separated from each other, such as text placement. Typically processing services do not include capabilities for providing persistent storage of data. With regard to our application scenarios processing services are needed to implement on-the-fly generalization, such as adaptive zooming. Between data access through a Web Feature Service (WFS) and map presentation with the help of a Web Mapping Service (WMS), the Generalization Service runs completely automatically as a middleware to adapt information to the screen size of an output device, used symbolization and map content. In this context performance is very important, so lower quality of the Generalization Service will be accepted. Lower quality of Generalization Service means only simple selection and simplification operations are carried out in real time, while more time consuming, context dependent generalization operations will be omitted. Lower quality maps usually are sufficient for display on mobile devices where the image loading time is very important for the user's look-and-feel and too many details can not be displayed due to limited screen resolution.

The other application scenarios use the Generalization Service as support for interactive map production and user controlled semi-automated derivation of multiple representations at smaller scales. The *model and information management services* category from the OGC/ISO architecture model can be seen as collection of these kinds of services, which allow the development, manipulation, and storage of metadata, conceptual schemas, and datasets. For this type of application the main control lies on the user side. *Workflow services* can help to define, invoke, status and control service chaining. *Human interaction services* allow interaction during the generaliza-

tion process, for instance, to decide which object classes have to be generalized or selected and to parameterize Generalization Service operators. The OGC/ISO architecture model suggests two additional categories, one for *communication services* to encode and transfer data across networks and one for *system management services* which include, for instance, management of user access privileges. All these kinds of services can be advantageously combined in a generalization research platform.

4.2 Default and Advanced Generalization Service Category Based on GML Specification of Open Geospatial Consortium

The second way of categorizing Generalization Services is based on the OpenGIS® Geography Markup Language (GML) Implementation Specification, which differentiates between GML core schema elements and GML application schema elements. Default Generalization Services as one category deal with GML core schema elements and could be used with any GML dataset. In many cases, for example when simplifying single area objects (e.g. the geometry of a building) or line objects (e.g. the geometry of a river or street) application requirements may be simple and the default generalization behaviors could suffice to meet those requirements. Advanced Generalization Services as the second category encompass services for GML application schema elements. Application schema elements allow additionally the modeling of objects by means of attributes and object compositions.

Default Generalization Services contain generalization functionality which operates on the micro level. This means that generalization operations are carried out on single objects, while context dependency will not be considered. According to Ruas (2000) micro objects generalize themselves or react to orders for contextual operations given by (superordinate) meso objects. Using the typology of McMaster and Shea (1992) the following operators can be applied on single objects: simplification, smoothing, geometry type change, collapse, enhancement and selection based on geometry. Default Generalization Services allow the reduction of resolution by means of a single geometrical operation (e.g. line simplification) and simplified modeling (e.g. centerline representation of roads and streets).

Advanced Generalization Services consider also attributes, symbolization and spatial context through neighboring objects. Hence, generalization operations have to deal with groups of objects, so called meso objects. Meso objects (or meso agents) generalize themselves when they perform contextual operations (Ruas 2000). Advanced Generalization Services allow to implement the remaining, contextual generalization operators of the typology by McMaster and Shea (1992), including context-dependent selection, aggregation, typification, exaggeration and displacement of map objects, taking into account both the geometry as well as semantics and attributes of the objects involved.

4.3 Hierarchical Categorization

Extending the idea of default and advanced Generalization Services, a hierarchical breakdown can be made which distinguishes between the following categories

1. Generalization support service (e.g. services for buffering or for creating a topological data structure, a skeleton or a constrained Delaunay triangulation)
2. Generalization operator services (e.g. services for line simplification, line displacement, area aggregation)

3. Generalization process services (e.g. services for automated orchestration, services for evaluation of generalization results)

The first category contains services that should support the generalization process and the generalization operators. Therefore this category represents the lowest level of this hierarchical categorization. Examples are services for creating a topological data structure or services for creating a constrained Delaunay triangulation. The results of such a service is additional cartographic information which can be optionally stored also in a Multi-Resolution Database (MRDB). These kinds of services can be seen as enriching data with respect to the automated generalization process. The main goal of such services is to make information explicit, representing common structural properties such as neighborhood or proximity relations and alignments which can be usefully exploited by generalization operations (Neun et al. 2004).

The second service category delivers the functionality of standalone generalization operators. Several typologies of such generalization operators are suggested for instance by McMaster and Shea (1992). Examples are services for simplification, smoothing, aggregation, amalgamation, merging, collapse, refinement, exaggeration, enhancement and displacement. These generalization operator services can be further subdivided for point, line and area objects and specialized depending on object classes. It is obvious that rivers, borders and railway lines have to be generalized in a different way, despite the fact that all are line objects. The generalization operators of this second service category are offered in an interactive mode, with the user selecting appropriate generalization operators/algorithms as well as setting the control parameters of the algorithms.

The third hierarchy level of generalization process services uses services from lower categories and encompasses services which allow the control and orchestration of generalization operators. Examples are the meso agents as described for the advanced Generalization Service category (Ruas 2000). Automated control of the generalization process presently receives ample attention as a research topic. Besides agent-based modeling also combinatorial and continuous optimization approaches are proposed in the literature. Simulated annealing (Ware et al., 2003) as a combinatorial optimization approach allows the selection of generalization operations controlled by assigning costs to each operation. Continuous optimization approaches include the finite element method (Højholt, 2000), snakes or elastic beams (Burghardt and Meier, 1997; Bader, 2001; Galanda and Weibel, 2003) and least-squares adjustment (Harrie, 1999; Sester 2000). The latter methods are primarily used to control generalization operations of continuous nature, such as displacement or smoothing. All approaches mentioned so far, however, are still quite a way from a perfect modeling of the overall map generalization process. In the meantime, an interim mechanism for controlling the generalization process is provided by the three architecture patterns for service chaining in the OpenGIS® Web Services Architecture (OGC, 03-025) which can be used depending on purpose and map complexity:

- User defined (transparent) chaining: the human user entirely manages the workflow. For complex generalization processes for which no adequate process modeling exists yet.
- Workflow-managed (translucent) chaining: the human user invokes a Workflow Management service that controls the chain and the user is aware of the individual services. For medium-complexity generalization processes that can be specified as workflows.
- Aggregate service (opaque): the human user invokes a service that carries out the chain, with the user having no awareness of the individual services. For relatively simple, sequential generalization processes.

5 Implementation

The prototype implementations of Generalization Services reported here include examples of rather simple services (e.g. Douglas-Peucker line simplification, building simplification) where spatial context is not considered. The objective is to show the feasibility of the service-based approach and to describe the minimum set of components needed to run the Generalization Services over the internet. For the implementation of the Generalization Services we use an open source framework for mapping application called JUMP (http://www.jump-project.org) that delivers standard functionality for reading and writing files (Shape, GML), as well as modifying and visualizing cartographic data. JUMP is written in Java, which allows easy application service provision (ASP) over the internet. The usage of the framework's functions is enabled by including the JUMP libraries in Java servlets which are running in a servlet engine such as TOMCAT. These servlets contain algorithms with the generalization logic or controls for external generalization libraries, respectively.

5.1 Registry for Generalization Services

The central point for accessing and publishing Generalization Services is the registry. Comparable to "yellow pages" the registry has to be used when looking for available generalization functionality. It would make sense for some large independent organization such as the ICA Commission on Map Generalization and Multiple Representation to host such a registry database.

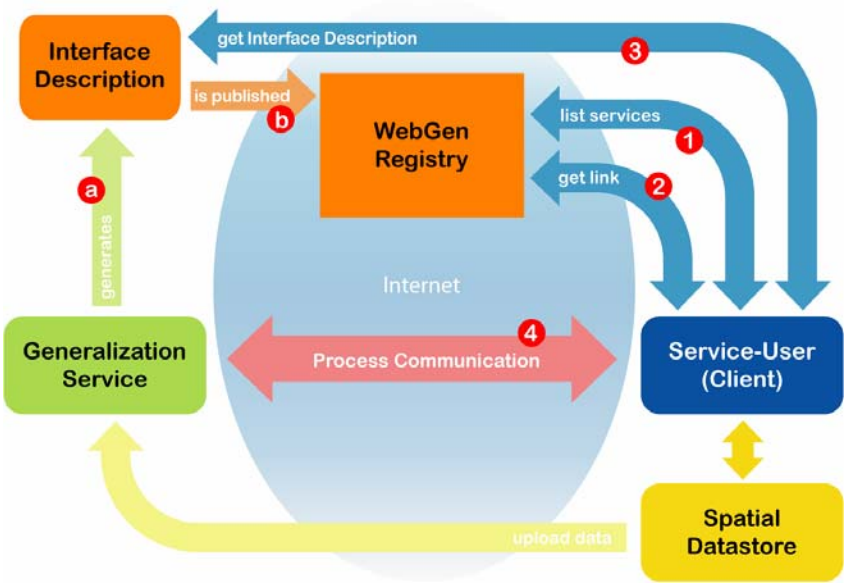

Figure 2: Registry for Generalization Services

To offer a Generalization Service the following steps have to be performed by the service provider, for instance by a research group (cf. Figure 2). The first task is to create (a) an interface description containing the parameters of the Generalization Service and the service endpoint, which consists of a URL address where the service can be accessed. Once the interface description is published (b) in the registry database the community can access the service.

Clients looking for Generalization Services can use the registry to find (1) available services. Selecting a desired Generalization Service the client obtains the link (2) to the interface descrip-

tion of the service. Accessing the interface description (3) the user knows the endpoint of the service, as well as the number, names and types of the parameters. With this information the client plug-in can create automatically a user interface for the selected service, allowing the specification of generalization parameters. Having a central access point is one advantage of a registry. Additionally, the registry automatically identifies service endpoints, so there is no active change needed by the service users if the service provider changes the location of the software on the server. Finally, the process communication (4) is responsible for the transfer of parameters and data between user and service provider.

5.2 Process Communication and Feature Metadata (Encoding and Upload)

The two main service concepts (cf. section 2.3) require different ways of communication for the data input (upload) and the output (download). Figure 3 illustrates three different ways to implement process communication. The concept of a research platform is either implemented as a form-based web page in a browser or as a plug-in for mapping software. The browser upload offers only the file upload via HTTP. The selected file is encoded automatically by the browser and has to be decoded on the server. The browser example offers the possibility to upload and process Shapefiles. As output in the browser the user can download a newly created Shapefile with the result of the generalization algorithm.

The implementation as a plug-in on the client GIS offers the possibility to encode the data directly out of the application. This gives the possibility to choose a better suited format for the transfer. In the example plug-in, the geo-features are encoded in a GML compatible format directly into a SOAP message (Simple Object Access Protocol), then transferred to the server. The use of another format (e.g. a binary format) would also be possible, with the possibilities of SOAP envelopes and e.g. MIME encoding. The output of the server is in the same format as the request and can again be decoded on the client and displayed there.

The concept of a middleware layer between the data source and the user (mostly in conjunction with a browser) introduces a third possibility for getting the data to the Generalization Service. In this case the client would simply send the URL of the data source (e.g. a WFS) to the Generalization Service as a parameter in the call. The server itself would then access the data source, process the data and send it back to the user (Sester et. al, 2004).

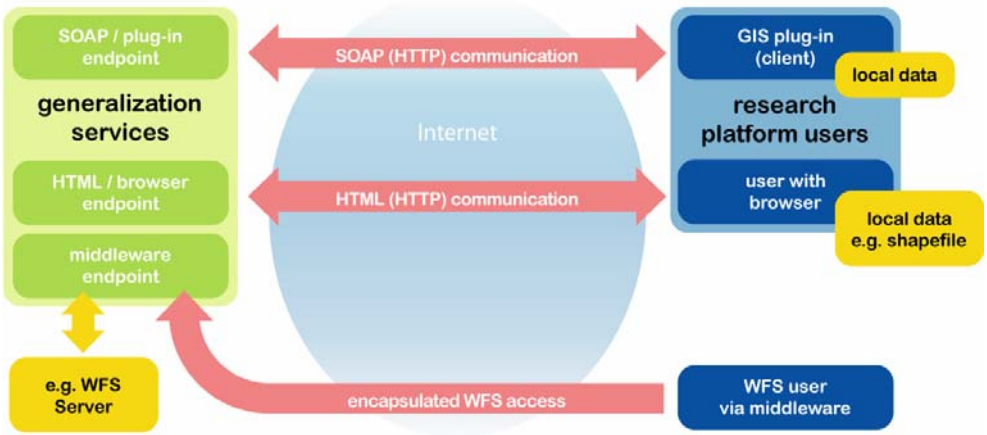

Figure 3: Three ways to implement process communication

The metadata specification for the geo-features is treated in the following way. Every feature can have an arbitrary number of attributes. The number, name and type (spatial or non-spatial) of the attributes needed by the algorithm are specified in the interface description (cf. section 5.3). Generalization data is always geometrically dominated. This means that every feature has at least one geometry plus possibly other attributes. The interface of a road generalization algorithm could for example require a geometry for each road and the road class (highway, major road, etc.). Listing 1 shows an example for such a schema specification. The users of the service would then in their client select the columns with the geometry and the road class in their local data set. For the data transfer the client then automatically assigns the name from the schema to the selected column. So, the server can identify which attributes it can use. Other attributes in a local dataset will simply be ignored not deleted.

```
<schema>
  <attribute>
   <name>geometry</name>
   <type>GEOMETRY</type>
  </attribute>
  <attribute>
   <name>class</name>
   <type>STRING</type>
  </attribute>
</schema>
```

Listing 1: Example feature schema

For the browser upload and the plug-in example the necessary encoder and decoder methods have been implemented in Java. These classes form the communication framework which can be used for the client-server communication.

5.3 Service Invocation and Client Implementation

On the client side both an example for the browser based Generalization Service and a plug-in for the JUMP Unified Mapping Platform have been implemented. Other plug-ins for platforms such as ArcView® are also planned. The client implements an easy to use GUI (Graphical User Interface) for the service user. The browser example works with standard HTML pages in every major browser platform. The user accesses such a service through a start page (user authentication with password can be activated). This start page, containing all available services, is dynamically created with information from the service registry (cf. section 5.1). After selecting a particular Generalization Service the user is presented a new, dynamically created page which allows him/her to enter the parameters for the algorithm and upload a Shapefile from his/her local system.

The JUMP plug-in does mostly the same as the browser solution but it integrates seamlessly into the software so that the user does not have to quit the application and even does not necessarily notice a big difference between using a local or remote algorithm. The plug-in integrates into the JUMP menu bar. The first time after the installation of the plug-in the user has to enter the URL of the registry (cf. section 5.1). With this URL the plug-in automatically looks every time it is started for all available Generalization Services. The result of this query is displayed in a selection list to the user (see Figure 5).

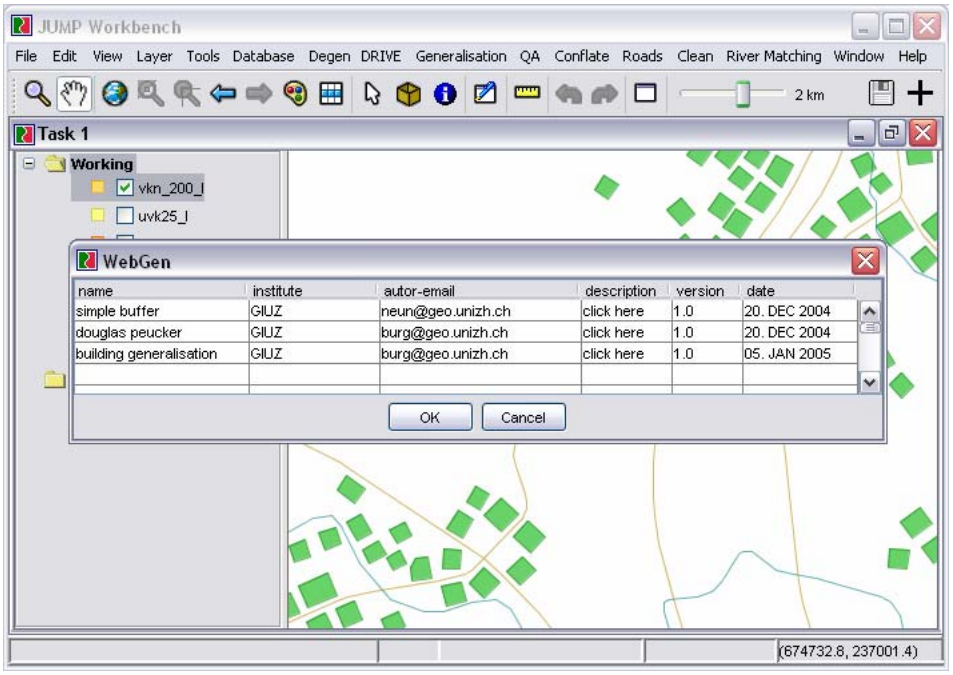

Figure 5: List of available services (from the Generalization Service registry)

From the list of available services the user selects the desired Generalization Service. The user then is presented an entry form for the corresponding algorithm's parameters. The configuration of all those entry forms is dynamically created from the interface description (cf. section 5.1). An example of such a simple interface description for the Building Simplification algorithm is shown in Listing 2. The data format for the interface description is XML. The important parts of the interface description (Listing 2) are highlighted.

```
<?xml version="1.0" encoding="UTF-8"?>
<webgen>
<name>building generalization</name>
<method>buildingGenNew</method>
<endpoint>http://www.geo.unizh.ch:8080/neun/servlet/GenHandlerXML</endpoint>
<description>This algorithm simplifies all buildings in a layer and returns a layer
containing the resulting buildings!</description>
<config>
 <layer>
  <schema>
   <attribute>
    <name>GEOMETRY</name>
    <type>GEOMETRY</type>
   </attribute>
 </schema>
 </layer>
 <param>
  <name>min edge length</name>
  <type>DOUBLE</type>
  <description>Minimum Edge Length!</description>
 </param>
</config>
</webgen>
```

Listing 2: The interface description in XML

The entire description is in one XML container. The tag <method> specifies the generalization algorithm which has to be used on the server and has to be passed to the server proxy. The tag <endpoint> specifies the URL of the server proxy. The tag <layer> specifies the minimum schema (metadata) the algorithm needs. In the example (Building Simplification algorithm) only geometries (<name> and <type>) are needed. Other attributes can also be contained in the layer but they will be ignored. For another algorithm like road re-classification e.g. a second attribute indicating the class of the road would be needed. The other parameters which are needed by the algorithm are indicated by the <param> tag. There can be as many parameters as needed. In the example there is only one parameter, the minimal edge length (type DOUBLE), needed.

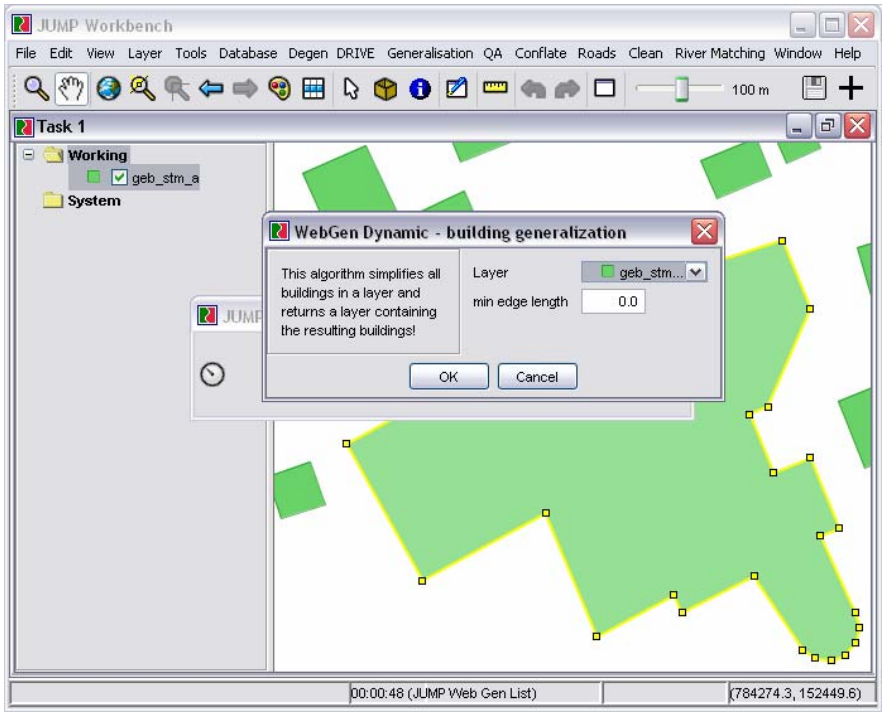

Figure 6: Algorithm interface generated dynamically from an interface description

Figure 6 shows the automatically generated entry form for the Building Simplification algorithm. After selecting the layer with the geometries and entering the tolerance, the data is transferred to the server and processed. The resulting geometries are sent back to the client and displayed in a new layer in JUMP.

The implementation example assumes a near real-time execution of the Generalization Service. The client is kept in a blocked wait mode. In this case the generalized data can be delivered immediately back to the client. Execution time is limited by client and server timeout settings and users' patience. For more time consuming operations, a system which informs the user when the service is finished would be possible and preferable. So, during the processing of a request the client would not be blocked and could perform other tasks. This has not been implemented so far, however. For the middleware concept only the real-time execution is of interest, because the user usually wants to see the map very quickly. The (simple) algorithms implemented so far have all performed very fast, so the bottleneck was not the computing time but the network transfer time. In the middleware concept, assuming a fast connection between data source and Generalization Service, this is negligible.

6 Conclusion

In an attempt to stimulate the discussion about Generalization Services in the generalization research community this paper explained the minimal requirements for the client and server side implementation of Generalization Services. It was shown how developers and researchers can make their generalization functionality available by means of an interface description and how possible users of these services can find them through a registry database. As an example simple Generalization Services for line and building simplification were implemented and accessed dynamically from the open source Java Unified Mapping Platform. To show the advantages of Generalization Services two different application scenarios were proposed. The first scenario implements the processing service as a middleware solution in order to realize on-the-fly generalization for tasks such as adaptive zooming. The second application scenario focuses on interactive generalization as support for interactive map production and user controlled semi-automated derivation of multiple representations at smaller scales.

The limitations of our solution are that no symbolization was considered so far and only context independent Generalization Services were implemented. Also, there is no user or session management up to now. This would be needed for longer computations. Further research has to investigate ways of interacting and chaining of Generalization Services, which is related to the yet largely unsolved problem of automated orchestration of generalization operators. To take full advantage of distributed services and grid computation ways have to be found for separating and distributing generalization tasks. The partitioning of a generalization task based on the identification of independent problems (e.g. the generalization of building blocks surrounded by streets) or the subdivision of the map area with solving boundary problems is imaginable.

Acknowledgements

Research reported in this paper was partially funded by the Swiss NSF through grant no. 20-101798, project DEGEN. We are grateful to Alistair Edwardes for helpful comments on the paper.

References

Badard, T., and A. Braun. 2003. OXYGENE: An open framework for the deployment of geographic web services. In: *Proc of the 21st Int. Cartographic Conf.,Durban, South Africa.* (CD-ROM).

Bader, M. 2001. *Energy Minimization Methods for Feature Displacement in Map Generalization*. Doctoral Thesis, Department of Geography, University of Zurich, Switzerland.

Burghardt, D. and S. Meier. 1997. Cartographic Displacement Using the Snakes Concept. In: Förstner, W., and L. Plümer (eds), *Semantic Modeling for the Acquisition of Topografic Information from Images and Maps,* Birkhäuser Verlag, 59-71.

Edwardes, A., D. Burghardt, M. Bobzien, L. Harrie, L. Lehto, T. Reichenbacher, M. Sester and R. Weibel. 2003. Map Generalisation Technology: Addressing the need for a common research platform. In: *Proceedings of 21st International Cartographic Conference. Durban, South Africa.* (CD-ROM)

Edwardes, A. and D. Burghardt. 2005. Experiments to build an open generalisation system. In: W. Mackaness, A. Ruas and T. Sarjakoski (eds), *Challenges in the Portrayal of Geographic Information: Issues of Generalisation and Multi Scale Representation.*.

Fitzke, J., K. Greve, M. Müller, A. Poth. 2004. Building SDIs with Free Software – the deegree project. In: *Proc. 7th Conf. Global Spatial Data Infrastructure (Bangalore, India).*

Galanda, M. and R. Weibel. 2003. Using an Energy Minimization Technique for Polygon Generalization. *Cartography and Geographic Information Science,* **30**(3): 259-275.

GML. 2005. OpenGIS® Geography Markup Language (GML) Implementation Specification http://www.opengis.org/techno/implementation.htm (accessed 01/2005).

Harrie, L. 1999. The Constraint Method for Solving Spatial Conflicts in Cartographic Generalization. *Cartography and Geographic Information Science,* **26**(1): 55-69.

He, Hao. 2003. What is Service-Oriented Architecture? http://webservices.xml.com/pub/a/ws/2003/09/30/soa.html

Højholt, P. 2000. Solving Space Conflicts in Map Generalization: Using a Finite Element Method. *Cartography and Geographic Information Science,* **27**(1): 65-73.

Illert, A. and S. Afflerbach. 2004. Global schema specification. GiMoDig-project, IST-2000-30090, Deliverable D5.3.1, Public EC report, 35 pgs., http://gimodig.fgi.fi/deliverables.php (accessed 01/2005)

JUMP. 2005. JUMP Pilot Project – Fostering the Development of the JUMP GIS Platform. http:// www.jump-project.org; http://jump-pilot.sourceforge.net/index.php (accessed 01/2005).

Lehto, L., and T. Sarjakoski. 2004. An Open Service Architecture For Mobile Cartographic Applications. In: G. Gartner (ed.), Location Based Services & TeleCartography, *Proceedings of the Symposium 2004*, Vienna University of Technology, January 28-29, 2004, Vienna, 141-145.

McMaster, R., and S. Shea. 1992. *Generalization in Digital Cartography*. Washington D.C: Association of American Geographers.

Neun, M., R. Weibel and D. Burghardt. 2004. Data Enrichment for Adaptive Generalisation. In: *ICA Workshop on Generalisation and Multiple Representation* (Leicester), available from http://ica.ign.fr/

OGC and ISO. 2002. The OpenGIS™ Abstract Specification, Topic 12: OpenGIS Service Architecture, Version 4.3. Abstract Specification OGC 02-112. Also available as ISO/DIS 19119 – Geographic Information Services.

Ruas, A. 2000. The Roles of Meso Objects for Generalisation. In: *Proceedings 9th Symposium on Spatial Data Handling (Beijing, China)*: 3b50-3b63

Sester, M. 2000. Generalization Based on Least-squares Adjustment. In: *International Archives of Photogrammetry and Remote Sensing*, Vol. XXXIII, Part B4, Amsterdam, 931-938.

Sester, M., L. T. Sarjakoski, L. Harrie, M. Hampe, T. Koivula, T. Sarjakoski, L. Lehto, B. Elias, A.-M. Nivala, and H. Stigmar. 2004. Real-time Generalisation and Multiple Representation in the GiMoDig Mobile Service. GiMoDig-project, IST-2000-30090, Deliverables D7.1.1*, D7.2.1* and D7.3.1, Public EC report, 151 pgs. http://gimodig.fgi.fi/deliverables.php (accessed 01/2005)

Vivid Solutions. 2004. JTS Topology Suite and JTS Conflation Suite, http://www.vividsolutions.com/ (accessed 01/2005)

Ware, J.M., C.B. Jones and N. Thomas. 2003. Automated Map Generalization with Multiple Operators: A Simulated Annealing Approach. *International Journal of Geographical Information Science*, **17**(8): 743-769.

W3C. 2005. Web Services Architecture. http://www.w3.org/TR/2004/NOTE-ws-arch-20040211/ (accessed 01/2005)

WFS. 2005. OpenGIS® Web Feature Service Interfaces Implementation Specification. http://www.opengis.org/techno/implementation.htm (accessed 01/2005)

WMS. 2005. OpenGIS® Web Map Service Interfaces Implementation Specification. http://www.opengis.org/techno/implementation.htm (accessed 01/2005)