

Web Services for an Open Generalisation Research Platform

Moritz Neun and Dirk Burghardt

Department of Geography, University of Zurich, {neun,burg}@geo.unizh.ch

Abstract While automated access and presentation of cartographic data over the internet are defined, services for automated generalisation are not yet standardised. This paper aims to show advantages of applying the service concept to generalisation for the development of a common research platform, where researchers would have access to a common generalisation framework. There follows a detailed explanation of a real implemented Generalisation Service. It is shown, how software developers can make their generalisation 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 generalisation.

1. Introduction

The map generalisation research community has started to develop an interest into Generalisation Services as well, driven by the desire to develop a common open research platform that would allow testing and sharing of generalisation algorithms (Badard and Braun, 2003; Edwardes et al., 2003). This has been evidenced through discussions at the meetings of the ICA Commission on Map Generalisation and Multiple Representation in Paris 2003 and Leicester 2004 (ICA, 2004). Such an open generalisation system supports co-operation by sharing of techniques within the cartographic research community, for example new algorithms, data structures or measures. It also supports external collaboration through the application of generalisation in other GIS research areas, for example in geo-visualisation and location-based services.

In the “Issues of Interoperability in Map Generalisation” (ICA, 2004) a strong interest for developing a standardised service based architecture has been expressed. This special type of service, the “Generalisation Service”, has so far not been standardised by the Open Geospatial Consortium (OGC). The starting point for such a Generalisation Service should be some suitable small services (ICA, 2004) without dealing with the harmonisation of data types and structures (Lehto and Sarjakoski, 2004; Sester et al., 2004; Illert and Afflerbach, 2004). The use of free software for Generalisation Services should be preferred. This has already been demonstrated for interoperable services for the visualisation of spatial information (Fitzke et al., 2004), the use of free software for Generalisation Services should be preferred. There are several advantages of using Generalisation 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. 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. Last, the service can be accessed over the internet or locally.



Figure 1: Generalisation service for an open research platform

2. Categories of Generalisation Services

The complexity of Generalisation Services can be seen at different levels of abstraction going from simple services on independent features up to highly context dependent services with control over the whole workflow (Burghardt et al., 2005):

1. **Generalisation support service:** e.g. services for buffering or for creating a topological data structure, a skeleton or a constrained Delaunay triangulation. Results of such a service can be seen as additional (enriching) cartographic information with respect to the automated generalisation process. The main goal of

such services is to make information explicit, representing common structural properties such as neighbourhood or proximity relations and alignments which can be usefully exploited by generalisation operations (Neun et al., 2004) and optionally stored in a Multi-Resolution Database (MRDB).

2. **Generalisation operator services:** deliver the functionality of standalone generalisation operators like the ones defined by McMaster and Shea (1992). Examples are services for simplification, smoothing, aggregation, amalgamation, merging, collapse, refinement, exaggeration, enhancement and displacement. These generalisation operator services can be further subdivided for point, line and area objects and specialised depending on object classes. It is obvious that rivers, borders and railway lines have to be generalised in a different way, despite the fact that all of them are line objects. The generalisation operators of this second service category are offered in an interactive mode, with the user selecting appropriate generalisation operators/algorithms as well as setting the control parameters of the algorithms.
3. **Generalisation process services:** using services from lower categories for the control and orchestration of generalisation operators. Examples are services for automated orchestration, services for evaluation of generalisation results and the meso agents as described for the advanced Generalisation Service category (Ruas 2000). Automated control of the generalisation process presently receives ample attention as a research topic. Besides agent-based modelling, combinatorial and continuous optimization approaches are also proposed in the literature. Simulated annealing (Ware et al., 2003) as a combinatorial optimization approach allows the selection of generalisation 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).

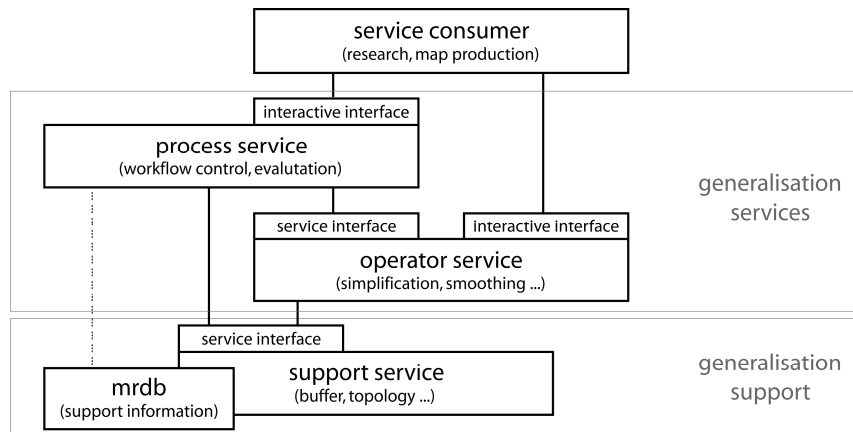


Figure 2: Categories of generalisation services

All approaches mentioned so far, however, are still quite afar from a perfect modelling of the overall map generalisation process. In the meantime, an interim mechanism for controlling the generalisation 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:

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

3. Implementation

The prototype implementations of Generalisation 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 Generalisation Services over the internet. For the implementation of the Generalisation 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 visualising cartographic data. JUMP is written in Java, which allows easy application service provision (ASP) over the

internet via Java servlets on a web-server. These servlets take care of the communication between client and server and pass the call to the generalisation algorithms.

3.1 Registry for Generalisation Services

In an open research model every researcher can present his own generalisation service. Through the internet and the use of platform independent technologies such services can reside on servers all over the world. For discovering these services some “Yellow Pages” are needed, indicating which services are available, where they are located and what algorithms they offer (Burghardt et al., 2005). This is the “Registry” for generalisation services. The registry offers a single access-point where all further information is to be found. Whilst services can change or move they are always to be found again through the registry. This model for sharing and discovering generalisation services can be summarised by the “publish-find-bind” paradigm (UDDI, 2004), shown in Figure 3.

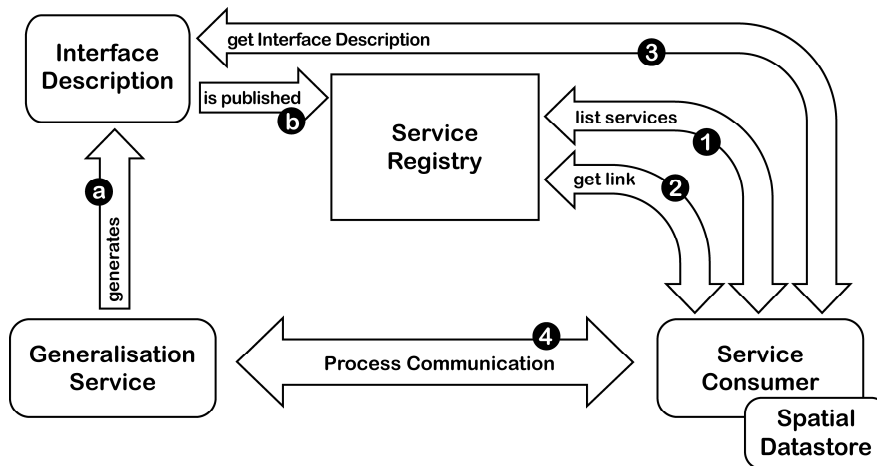


Figure 3: Registry for Generalisation Services

To offer a Generalisation Service the following steps have to be performed by the service provider, for instance by a research group (cf. Figure 3). The first task is to create (a) an interface description containing the parameters of the Generalisation 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 Generalisation Services can use the registry to find (1) available services. Selecting a desired Generalisation Service the client obtains the link (2) to the interface description 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 automatically create a user interface for the selected service, allowing the specification of generalisation 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 service endpoint. Finally, the process communication (4) is responsible for the transfer of parameters and data between user and service provider.

3.2 Process Communication and Feature Metadata (Encoding and Upload)

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 automatically encoded 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 generalisation 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 introduces a third possibility for getting the data to the Generalisation Service. In this case the client would simply send the URL of the data-source to the Generalisation Service as a parameter in the call. The access to the data-source could be a Web Feature Server (WFS). The server itself would then access the data source, process the data and send it back to the user.

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 3.3). Data, used by generalisation algorithms, is always geometrically dominated. This means that every feature has at least one geometry plus possibly other attributes. The interface of a road generalisation algorithm could for example require a geometry for each road and the road class (highway, major road, etc.). Figure 4 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>

```

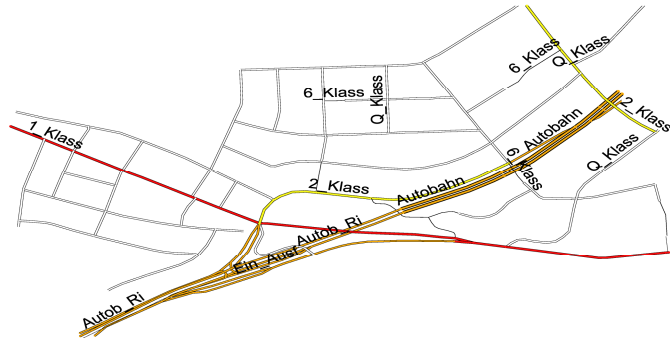


Figure 4: Exemplary feature schema needed for an algorithm

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.

3.3 Service Invocation and Client Implementation

On the client side both an example for the browser based Generalisation 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 3.1). After selecting a particular Generalisation 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 3.1). With this URL the plug-in automatically looks every time it is started for all available Generalisation Services. The result of this query is displayed in a selection list to the user (see Figure 5).

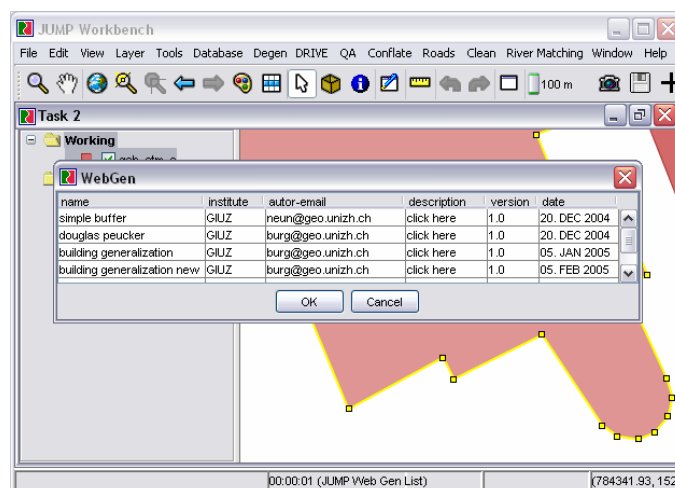


Figure 5: List of available services (from the Generalisation Service registry)

From the list of available services the user selects the desired Generalisation Service. The user is then presented an entry form for the corresponding algorithm's parameters. The configuration of all those entry forms is dynamically created from the interface description. An example of such a simple interface description for the Building Simplification algorithm is shown in Figure 6. The data format for the interface description is XML.

```

<?xml version="1.0" encoding="UTF-8" ?>
<webgen xmlns:gml="http://www.opengis.net/gml" xmlns:SOAP-ENC="http://schemas.xmlsoap.org/soap/encoding/">
  <name>building simplification</name>
  <method>buildingSimplification</method>
  <endpoint>http://server/soap-endpoint</endpoint>
  <description>simplifies all buildings in a layer and returns resulting buildings</description>
  <config>
    <layer>
      <schema>
        <attribute name="geom" type="GEOMETRY">
          <allowed>gml:Polygon</allowed>
          <allowed>gml:MultiPolygon</allowed>
        </attribute>
      </schema>
    </layer>
    <param name="min edge length" type="SOAP-ENC:double">
      <description>Minimum Edge Length</description>
    </param>
  </config>
</webgen>

```

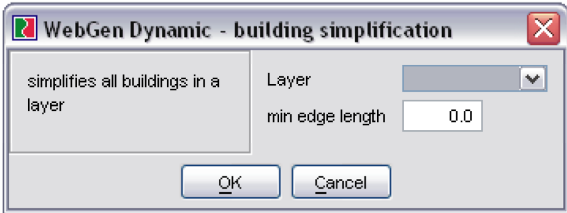


Figure 6: XML interface description

The entire description is in one XML container. The tag <method> specifies the generalisation algorithm which has to be used. It 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 necessary algorithm parameters are indicated by the <param> tag. There can be any number of parameters. In the example is only one parameter, the minimal edge length (type DOUBLE), used.

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.

3.4 Implementing Algorithms on the Server

For the implementation of the two usage-examples (by browser, and by plug-in) the Tomcat-Server for Java servlets (<http://jakarta.apache.org/tomcat/>) offers the needed functionality. Both the standard HTTP communication with HTML-documents for the browser based service and the advanced SOAP-Communication with HTTP and XML can be used on the Tomcat server. The JUMP Unified Mapping Platform (<http://www.jump-project.org>) libraries offer Shape-file functions for the browser example. For the handling of geometries in general the JTS (Java Topology Suite, <http://www.vividsolutions.com>) libraries are used.

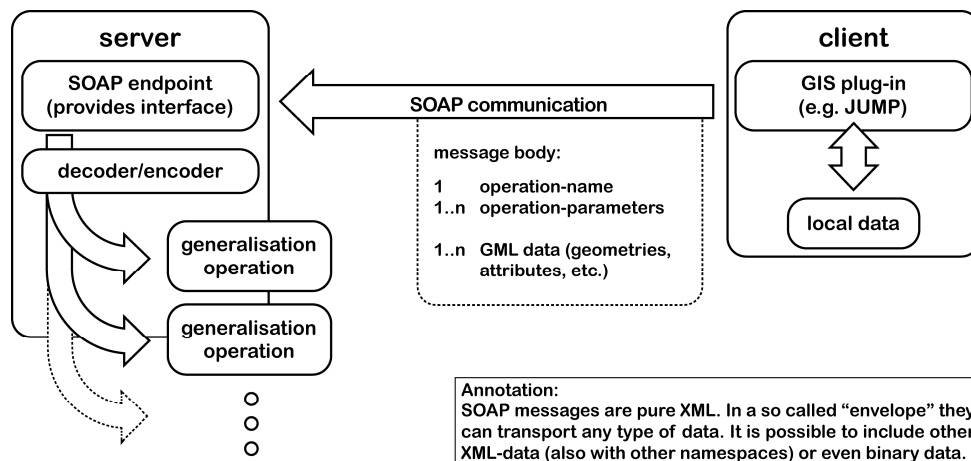


Figure 7: Plug-in - server communication and server proxy

The implementation concept foresees a proxy-endpoint on every service-provider's server, so only one service-endpoint per server is necessary. This means that all the messages which are passed to the server contain the name of the called algorithm. With this information the proxy can send the message to the appropriate algorithm. The proxy-servlet with the communication and message-decoding functionalities together with JUMP and JTS libraries form a small framework which can also be deployed to other servers easily.

Every algorithm on the server is implemented in a separate class. All those classes implement a common interface with one access-method. The proxy passes the decoded message parts, the parameters (as array) and the features (as collection) to the algorithm. With this solution the proxy can handle various calls with different feature and parameter numbers. The handling of features and parameters is part of the algorithm class. A developer of e.g. a Douglas Peucker algorithm would take all the features out of the call and a parameter "tolerance" out of the parameter-array. For the correct communication between client and server, he would therefore create the interface description (see "Registry for generalisation services") which contains the information that the call has to contain a layer with geometric features and a parameter called "tolerance" of type double.

The format of the geo-features which are passed from the proxy to the generalisation algorithm class follow the concept which is used in the JUMP software. The geo-features are in a FeatureCollection which holds all the features and allows access to single features or the whole collection. Every Feature is represented by one spatial attribute (its main-geometry) and zero or more other attributes which can be either spatial (e.g. an additional geometry) or non-spatial (integers, doubles, strings). The spatial attributes are JTS-Geometries and represent a implementation of the OGC Simple Features Specification.

3.5 Time Constraints and Performance Aspects

The implementation example assumes a quick execution of the generalisation services. The client is in a blocked wait mode. In this case the generalised data can be delivered immediately back to the client. Execution time is limited by Client and Server time out settings and users patience. For more time consuming operations it would be possible to create a system, which informs the user when the service is done. So, during the processing of the request the client would not be blocked and could do other work. This has so far not been implemented. The up to today implemented algorithms have all performed very fast, so the bottleneck was not the computing-time but the network-transfer time.

4. Conclusion

In an attempt to stimulate the discussion about Generalisation Services in the generalisation research community this paper explained the minimal requirements for the client and server side implementation of Generalisation Services. It was shown how developers and researchers can make their generalisation 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 Generalisation Services for line and building simplification were implemented and accessed dynamically from the open source Java Unified Mapping Platform. To show the advantages of Generalisation Services the application scenario focuses on interactive generalisation 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 symbolisation was considered so far and only context independent Generalisation Services were implemented. Further, there is also 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 Generalisation Services, which is related to the yet largely unsolved problem of automated orchestration of generalisation operators. To take full advantage of distributed services and grid computation, ways have to be found for separating and distributing generalisation tasks. The partitioning of a generalisation task based on the identification of independent problems (e.g. the generalisation of building blocks surrounded by streets) or the subdivision of the map area (boundary problems should be solved) 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 subject.

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 Topographic Information from Images and Maps*, Birkhäuser Verlag, 59-71.
- Burghardt, D., M. Neun and R. Weibel. 2005. Generalization Services on the Web – A Classification and an Initial Prototype Implementation. In: *Proc. American Congress on Surveying and Mapping, Auto-Carto 2005 (Las Vegas, USA)*
- 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., D. Burghardt, M. Neun. 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.
- Harrie, L. 1999. The Constraint Method for Solving Spatial Conflicts in Cartographic Generalization. *Cartography and Geographic Information Science*, 26(1): 55-69.
- Højholt, P. 2000. Solving Space Conflicts in Map Generalization: Using a Finite Element Method. *Cartography and Geographic Information Science*, 27(1): 65-73.
- ICA. 2004. Brain storming Sessions. *ICA Workshop on Generalisation and Multiple Representation* (Leicester), available from <http://ica.ign.fr/>
- 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)
- 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)

UDDI. 2004. Universal Description, Discovery, and Integration. <http://www.uddi.org/> (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.