Web service approaches for providing enriched data structures to generalisation operators

Neun, M ; Burghardt, D ; Weibel, Robert

Abstract: Web service technologies can be used to establish an interoperable framework between different generalisation systems. In a previous article three categories of generalisation web services were identified, including support services, operator services and processing services. This paper focuses on the category of support services. In a service-based generalisation system, the purpose of support services is to assist the generalisation process by providing auxiliary measures, procedures and data structures that allow the representation of structural cartographic knowledge. The structural knowledge of the spatial and semantic context and the modelling of structural and spatial relationships is critical for the understanding of the role of cartographic features and thus for automated generalisation. Support services should extract and model this knowledge from the raw data and make it available to other generalisation operators. On the one hand the structural knowledge can be expressed by enriching map features with additional geometries or attributes. On the other hand, there exist various hierarchical and nonhierarchical relationships between map features, many of which can be represented by graph data structures. After a brief introduction to the interoperable web service framework, this paper proposes a taxonomy of generalisation support services and discusses its elements. It is then shown how the complex output of such services can be represented for use with web services and stored in a reusable fashion. Finally, the utilisation of support services is illustrated on four implementation examples of support services that also highlight the interactions with the generalisation operators that use these auxiliary services.

DOI: https://doi.org/10.1080/13658810701348997

Web Service Approaches for Providing Enriched Data Structures to Generalisation Operators

MORITZ NEUN*, DIRK BURGHARDT and ROBERT WEIBEL

University of Zurich, Department of Geography, Winterthurerstrasse 190, CH-8057 Zurich, Switzerland

Web service technologies can be used to establish an interoperable framework between different generalisation systems. In a previous article three categories of generalisation web services have been identified, including support services, operator services and processing services. This paper focuses on the category of support services. In a service-based generalisation system, the purpose of support services is to assist the generalisation process by providing auxiliary measures, procedures and data structures that allow to represent structural cartographic knowledge. The structural knowledge of the spatial and semantical context and the modelling of structural and spatial relationships is critical for the understanding of the role of cartographic features and thus for automated generalisation. Support services should extract and model this knowledge from the raw data and make it available to other generalisation operators. On the one hand the structural knowledge can be expressed by enriching map features with additional geometries or attributes. On the other hand there exist various hierarchical and non-hierarchical relationships between map features, many of which can be represented by graph data structures. After a brief introduction to the interoperable web service framework this paper proposes a taxonomy of generalisation support services and discusses its elements. It is then shown how the complex output of such services can be represented for use with web services and stored in a reusable fashion. Finally, the utilisation of support services is illustrated on four implementation examples of support services that also highlight the interactions with the generalisation operators that use these auxiliary services.

Keywords: automated map generalisation, web services, feature generalisation services, generalisation support services, structural knowledge, supporting data structures, graph data structures, triangulations

1. Introduction

Structural knowledge, that is, the knowledge of the spatial and semantical context of map features and their structural relationships is critical for the understanding of the role of the features represented on a map and thus for automated generalisation (Armstrong, 1991; Ruas and Plazanet 1996; Regnauld, 2005). This supporting knowledge, however, has rarely been coded directly into the commonly available cartographic databases. Hence, algorithms for extracting structural knowledge from the raw cartographic data are required in order to 'enrich' the available cartographic databases (Ruas and Plazanet 1996). Such algorithms – like other generalisation algorithms – are being developed by various research groups, typically on their platform of choice. The problem, then, is that although the generalisation toolbox is seemingly steadily filling up no real progress can be made in research nor in production as long as these tools stay on different platforms (Edwardes et al. 2003). Web services can be used to make this knowledge available in a platform independent way allowing the combination of different algorithms and supporting data structures using a common interface (Edwardes et al. 2007, Burghardt et al. 2005).

In a previous article (Burghardt et al. 2005) we have introduced, for the first time, a comprehensive architecture of generalisation web services and demonstrated its use on an initial implementation of simple generalisation algorithms. As the original paper argues, a complete service-based generalisation system will require three types of services: operator services, support services, and process services. Operator services implement generalisation operators (e.g. simplification, smoothing, aggregation); support services provide supporting measures and data structures to the operator services; and process services are responsible for workflow control (of operator and support services) and evaluation of the generalisation results. This paper focuses exclusively on support services, attempting to define their role within a service-based generalisation system; proposing a taxonomy of support services that are considered essential for structure recognition in map generalisation; discussing issues of the interoperable storage and exchange of the out put of such support services; and presenting and discussing what we believe to be representative implementation examples. With this, we hope to

* Corresponding author. Email: neun@geo.unizh.ch

provide a foundation for the systematic further development of generalisation web services, an example being the WebGen framework (§ 2.4), a service platform that represents a joint effort of several authors residing in different research groups.

Generalisation web services encompass a whole set of different methods and processes in order to deliver generalised maps adapted to constraints such as legibility or user needs. The use of web services in generalisation is not limited to the creation of web maps for display in a web browser. Generalisation services can also be used for the coupling of different generalisation platforms and for the provision of generalisation functionalities as an interoperable toolbox within a map production context.

In Section 2, we will briefly review the main elements of the initial framework for generalisation web services as a foundation of this paper and to provide an embedding for support services used in generalisation. In Section 3, the paper goes on to define and discuss a taxonomy of generalisation support services. In Section 4, it is shown how the complex output of such services can be represented for use with web services and stored in a reusable way. In Section 5, the utilisation of support services is illustrated with four implementation examples of support services in combination with generalisation operators that use the output of these support services. The examples have been chosen so as to form a representative subset of the main types of support services found in the proposed taxonomy. Sections 6 and 7 end the paper with a discussion and conclusions, respectively.

2. Generalisation Web Services

This section briefly reviews the main elements of previous work (Burghardt et al. 2005, Edwardes et al., 2007) to provide the basis for the discussion of the following sections.

2.1 Interoperability through Web Services

There are several advantages of using a Service Oriented Architecture (SOA) such as Web Services for generalisation. In an SOA environment, resources on a network are made available as independent services. These services interoperate according to a formal specification and can be accessed without knowledge of their underlying platform implementation (Channabasavaiah et al., 2003). The platform independence makes the development independent from the operating system and the hardware used. Furthermore, services can be integrated into any software platform, such as web browsers, GIS, or map production software (Regnauld, 2006). It is even possible to write specific algorithms for special computer architectures such as clusters, grids, or other parallel processing systems and to offer such services to the subscribers (Burghardt and Neun, 2006).

Services extend the traditional approach of providing a software library with a well defined interface. Software libraries can often only be used in one particular programming language. Services however allow the interoperable use with different languages on different platforms. Thus services are more flexible, exchangeable and also comparable. Valuable functionalities from different platforms can be combined. Finally, (web) services can be accessed either over the internet or locally.

Web Services are not only useful for accessing data over the web but also for accessing processing functionalities on a remote server. The Open Geospatial Consortium Web Processing Services (OGC, 2005) are a first step towards web services for the execution of spatial algorithms, including those for generalisation. So far, however, there have been no specifications for the ‘Feature Generalisation Service’ mentioned in OGC (2002). The starting point for such a Generalisation Service should be some suitable small services (ICA, 2004) without having to deal with the harmonization of data types and structures (Lehto and Sarjakoski, 2004; Illert and Afflerbach, 2004). For interoperable services for the visualisation of spatial information (Fitzke et al., 2004) this has already been demonstrated, yet not with generalisation.

2.2 Types of Generalisation Services

The overall generalisation process involves both rather simple independent generalisation operations, which are applied only to individual map features, as well as highly context-dependent operations, which require comprehensive control over the generalisation workflow. Hence, Burghardt et al. (2005) argued that three types of generalisation services must be offered in order to enable comprehensive web-based generalisation. These three categories are shown in Figure 1 and briefly discussed below.

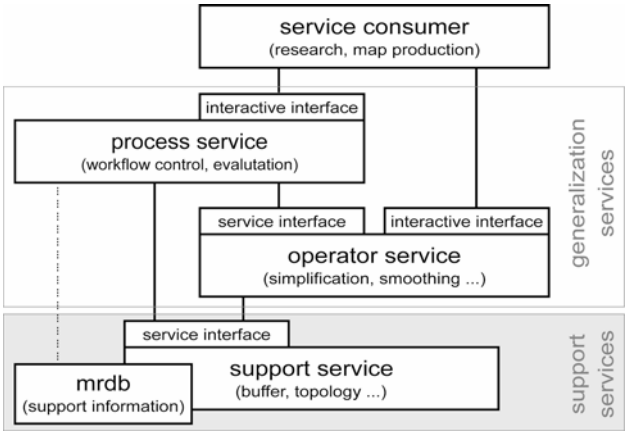


Fig. 1: Categories of generalisation services, the highlighted support services form the focus of this paper

2.2.1 Generalisation Support Services

Support Services can be seen as a provider of additional (enriching) cartographic information in support of the automated generalisation process. Examples are services for buffering or for creating a topological data structure, a skeleton or a constrained Delaunay triangulation (details see § 3). Support services take raw cartographic data with a simple structure as input and deliver either a simple structure but with additional enriching information (e.g. a priority ordering) or a more complex data structure with object relations as output (e.g. a graph data structure representing adjacency relationships between map features). Thus a main goal of such services is to make structural information explicit, representing common structural properties such as alignments, neighbourhood or proximity relations, which can be usefully exploited by generalisation operations. Support services can also be used for calculating measures and constraints. Sometimes the establishment of such supporting information is very expensive in terms of time and memory so that optionally their persistent storage in a special multi-resolution database (MRDB) can be useful e.g. for real-time generalisation. Support services together with multi-resolution databases are the fundamental service category, which offers its functionalities to all other service types (see Fig. 1). The representation and analysis of relations between map features, such as topological or metrical relations, are a fundamental part of GIScience (Worboys and Duckham, 2004). Therefore it is important to note that support services such as the ones discussed in § 3 could be equally useful in a more general, non-generalisation environment, particularly for the purposes of spatial analysis and decision support.

2.2.2 Generalisation Operator Services

Operator Services deliver the functionality of standalone generalisation operators such as the ones defined by McMaster and Shea (1992). Examples are services for simplification, smoothing, aggregation, amalgamation, merging, collapse, refinement, exaggeration, enhancement and displacement of cartographic features. These generalisation operator services can be further subdivided for point, line and area features and specialised depending on feature classes. Operator services themselves can use the functionalities of support services. For instance, a feature displacement service may make use of a triangulation data structure delivered by a triangulation support service. Generalisation operator services form a service-oriented toolbox for realising generalisation processes. In the current version of the WebGen prototype implementation more than 15 operator services are available and are continually extended.

2.2.3 Generalisation Process Services

Process Services use services from the lower categories for the control and guidance of generalisation operators (Fig 1). These include services for automated generalisation workflow control, as well as services for the evaluation of generalisation results. Automated control of the generalisation process presently receives ample attention as a research topic. Approaches for process control currently favoured in the map generalisation community include optimisation techniques such as simulated annealing (Ware et al., 2003) and energy-minimization (Bader et al., 2005) as well as agent-based methods (Barrault et al. 2001). For a review of such methods see Harrie and Weibel (2007); for a review of early work in this area see Buttenfield and McMaster (1991).

2.3 Generalisation Service Architectures

Generalisation services are delivering generalised data to the requesting client. But where does the original un-generalised data originate from which is then generalised by the service? In general two main scenarios are possible.

The first scenario is that the data comes from a geographic database, usually via a Web Feature Server (WFS). Thus this approach is merely an extension to a database. This ***middleware generalisation service*** delivers pre-generalised data. Such an approach is described by Sarjakoski et al. (2005) using a modified WFS. A possible use is an adaptive zoom for Web Map Services as it is currently the domain of multi-resolution databases (MRDB). This service would require a fully automated, real-time execution of the generalisation process. Another possible use are generalisation support services (the focus of this paper) which pre-process cartographic data e.g. from a WFS so that they can then be used by more complex generalisation algorithms.

In contrast, the second scenario foresees that users can generalise their own data, by sending them to the service. Such a ***generalisation toolbox service*** offers its processing capabilities including algorithm logics and computational power to the user. The ability to provide specialised or novel algorithms in a platform independent way allows the evaluation and integration of generalisation functionalities from different sources without forcing the users to adapt their systems to any specific needs. In an open toolbox service model everybody can present his/her own generalisation services. Through the Internet and the use of platform independent technologies such services can reside on servers all over the world. For discovering these services a Registry indicates which services are available, where they are located and what algorithms they offer.

Possible usage scenarios include research as well as map production. The map generalisation research community has an interest into generalisation services, driven by the desire to develop a common open research platform that would allow testing and sharing of generalisation algorithms (Edwardes et al., 2003). For the distributed and collaborative development of generalisation techniques the service model can help the coupling of algorithms and data structures on different platforms. The evaluation of algorithms etc. is facilitated. Generalisation toolbox services can support research cooperation by sharing of techniques within the cartographic research community (Regnauld, 2006), for example new algorithms, enriching data structures or measures.

As the service's logics reside only on the server of the owner even copyright protected methods can be offered, evaluated and incorporated without having to give executables or even code away. Hence, in map production services can help the coupling of heterogeneous production platforms in order to generate a seamless generalisation workflow. Further scenarios would even include offering specialised services on a per-use basis with a fee to occasional users of generalisation functionality.

2.4 Generalisation Toolbox Service Platform

The prototype of a generalisation toolbox service platform is the WebGen framework. Initial implementation details were reported in Neun and Burghardt (2005). The WebGen platform consists of client and server components. The client plug-in offers the configuration and selection of the desired service. The data to be processed by the service together with the parameters for the algorithm are encoded and sent to the service. The result, returned by the service, is then decoded and presented in the client. Currently client plug-ins for the JUMP Unified Mapping Platform (2006), a client as a browser based AJAX web page (Asynchronous JavaScript and XML) and a prototype plug-in for the Clarity generalisation software by 1Spatial (2007) are available. A client with similar functionalities could also be developed for other desktop GIS having an API for adding functionalities.

The first implementation of a generalisation server for the WebGen framework uses JAVA. Services written in JAVA can be offered directly. Algorithms written in other programming languages or algorithms that are available as executable programs can also be included with full functionality using file-based data transfer. Every service can also make calls to other services on the same or on another generalisation server. A simple example is that an algorithm, needing a buffer around a point, can send the point geometry to the buffer service, receiving back a buffer geometry.

Initially rather simple services (e.g. Douglas-Peucker line simplification, building simplification) with no consideration of spatial context were implemented in order to demonstrate the feasibility of the service-based approach. In this paper more advanced service types and their implementation will be described, with a focus on generalisation support services that form the foundation of structure recognition in map generalisation.

3. Taxonomy of Support Services

The execution of generalisation operators or algorithms depends on the input they receive. Important elements include algorithm parameters, the character of the map features to generalise, and also mutual influences between map features, such as roads exerting a push on nearby houses in map feature displacement. Thus the knowledge of the spatial context is a very important factor for generalisation (Mustière and Moulin, 2002). Most of this knowledge about relationships is only implicitly contained in off-the-shelf cartographic data sets. Generalisation algorithms usually have to create this knowledge about the spatial context in the form of auxiliary data structures during their execution. Examples range from the creation of a simple buffer to a topological data structure, a skeleton or a constrained Delaunay triangulation.

In this paper, we are assuming that we are dealing with vector cartographic data as well as vector data structures and algorithms. Raster cartographic data and algorithms are rarely used in map generalisation (e.g. with raster land-cover data). Hence we restrict our following taxonomy to the vector domain.

As mentioned above support services enrich map data with additional information such as cartometric measures, contextual knowledge such as relations between map features or also additional meta-data. Many of the measures and contextual data structures that form elements of our taxonomy of generalisation support services are available in some form in commercial GIS platforms. Hence, one might think that starting off from commercial GIS might be a suitable approach to develop appropriate supporting functions. However, there are two main reasons that speak against that approach. First, commercial GIS developers tend to favour functionality that satisfies the interests of a broad customer population, while map generalisation as well as other GIS fields such as spatial analysis or spatial queries often requires adapted and specialised forms of supporting data structures, as will be shown below. Hence, one of the intentions of proposing our taxonomy is also to show the breadth of support measures and data structures needed in this particular field and show their utility, in order to stimulate interest in developing these functions in GIS packages. Second, GIS platforms often have a rather monolithic architecture. Using web services the enriching data can be made available in a platform-independent way which allows the combination and evaluation of different services using a common interface. The developer can use support services as components for his/her system, relying on the defined interface of each service, without the need to rewrite code. It is potentially also possible to use multiple support services, even from different providers. If the service and the service consumer reside on the same computer, the services can also be accessed locally without network overhead and with the reliability that the service is always available. Support services can be used for data pre-processing with expensive calculations as well as for real-time generalisation.

The taxonomy of generalisation support services presented below tries to identify commonalities and differences between services. Different levels of complexity exist for support services in terms of their output data. Simple support services just do create supporting entities like geometries or attributes. More complex services create information about the relations between map features or between their properties. Thus categories of support services can be distinguished by the type of the supporting data they offer, and thus by the output they deliver to the service consumer (Fig. 2). Using the output complexity of a support service for the classification identifies an important commonality/difference between the different support services which can be used as a framework with different levels of complexity. The output defines also the complexity of the interface which is needed to access such a support service. In order to be able to build a common ‘library’ of support services for generalisation such a framework can help the distributed development process with the involvement of many different parties in the generalisation community.

There are two main types of support services. First, there are those services that equip entities (i.e. map features) with new or modified geometries or attributes (see § 3.1 for details and examples). Second, there is the large family of Relations (see § 3.2). Relations can exist between objects but also between properties like object states. They range from (directed or undirected) graphs, such as transport graphs and triangulations, to hierarchies which can be expressed by trees. All graphs and trees can also be represented as a matrix.

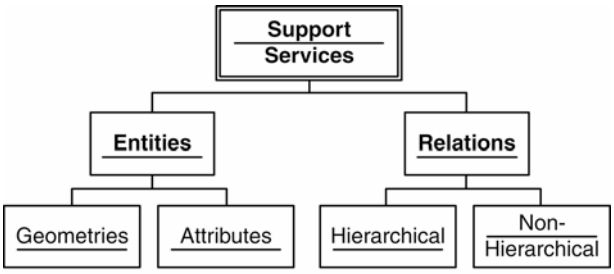


Fig. 2: Support service types (categorised by service output complexity)

3.1 Entity Support Services

The most obvious output of a support service are entities with supporting attributes or geometries in the form of the simple features (OGC, 1999). These services are just enriching features with additional attributes or are creating new support geometries. Functions to read and write these supporting data structures are included in most GIS software packages. Consequently, the discussion below will be brief.

3.1.1 Creating Geometries

The output of such a support service is just the derived geometries. OGC-style simple features can easily express them. Thus, no extra data format is needed to get results from the service.

A simple example of a geometry creating support service is the creation of the buffer polygon from an input geometry, which could then be used by a feature selection service. Taking, for instance, a road and a set of houses as input a selection service may return the houses contained in a certain buffer around the road. The output of both services is just simple features. A further example of supporting geometries includes the computation of alignment lines, chaining together a group of map features such as buildings (Christophe and Ruas, 2002). The creation of inflection points or identification of local extremes for line generalisation (Plazanet et al., 1995) serves as a final example of creating supporting geometries. It delivers a series of critical points which can then be used, among others, for line segmentation (partitioning) or the creation of trend lines, approximating a line by connecting the inflection points (Fig. 3).

Fig. 3: Original line; inflection points; trend line generation

3.1.2 Generating Attributes

These services take map features as input and return the features with modified or new attributes. In essence, most of these services are performing an analysis of the shape and structure of map features, also termed cartometric analysis (McMaster and Shea, 1992). An example of such a function is the calculation of the sinuosity of a line (Plazanet et al., 1995), which is stored as an attribute of the line feature. Plenty of other measures can be calculated, such as area or shape index of a polygon, shortest distances to nearest neighbours, and many more. Often, these measures can be used in a comparative way to establish priority orderings among map features (e.g. small polygons may be defined as insignificant and therefore omitted). The calculation of measures or constraints (Burghardt and Neun, 2006) can be used in an automated generalisation process for choosing appropriate operators or for evaluating algorithm results. Thus a support service can calculate for example the cost of the violation of a minimum building size constraint and attach this value as an attribute to the buildings of a map. Such a severity value can be used to trigger the appropriate generalisation operator and the parameters for every feature individually. If for example the minimum building size constraint is violated the building feature must be enlarged until the constraint is satisfied.

3.2 Relation Support Services

Spatial, structural and semantical relationships are essential ingredients for many complex generalisation operators (Mustière and Moulin, 2002; Regnauld, 2005). The corresponding relation support services have a more complex and context dependent output than the entity support services. Despite the fact that the value of such relational structures is well known they are unfortunately only sparsely available in commercial GIS and if available they are often not generic and rich enough to be exploitable for the purposes of map generalisation.

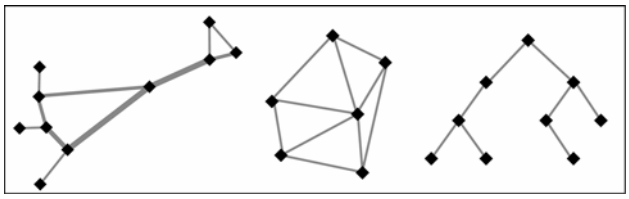


Fig. 4: Examples of graphs: Weighted graph, triangulation and tree

In general relations between cartographic features can be modelled as graph-like structures (Fig. 4). Examples are topological data structures (e.g. polygonal maps), transport and neighbourhood graphs, triangulations, or surface networks. As a more specific case, hierarchies can be expressed by trees being a special form of a graph with no cycles, rooted in a single node. The modelling and implementation of such graphs will be shown in §4.3.

In the next two sections hierarchical relations and non-hierarchical relations, respectively, are reviewed. For the sake of brevity we exclude relations for 3-D objects (e.g. terrain surfaces), though obviously both hierarchical and non-hierarchical relations can also be found in more than two dimensions. Examples of hierarchical relations for surfaces include hierarchical TINs (De Floriani and Puppo, 1995); the prime example of non-hierarchical relations in 3-D are surface networks (Pfaltz, 1976; Rana, 2004).

3.2.1 Hierarchical Relations

Hierarchical Relations can exist between cartographic features. Hierarchy creating criteria may be any property of the features involved such as their relative positions or an attribute such as a class scheme. Hierarchical relations can be expressed by trees. In some cases the hierarchy creating criterion can be a semantical property as with similarity trees or dendrograms generated by classification or clustering algorithms. In other cases a hierarchical relation represents a structural property as with network orderings, complex features, or reactive trees. Hierarchical relations can also be seen as a more specialized type of relation than the non-hierarchical relations.

Similarity trees or dendrograms: For the aggregation of geometries or the translation of features from one feature schema to another, often a reclassification of the feature categories is needed. A reclassification needs some sort of rule set on how to assign new categories to the input features. This rule set in many cases represents a strict hierarchy which assigns a new category to every input category (e.g. 'deciduous forest' and 'coniferous forest' are both reclassified to 'forest'). This hierarchy can be defined by the user of the system or generated automatically, for instance by a statistical evaluation of the properties of the input categories. A reclassification service would request a similarity tree from the support service and then classify the map features accordingly (see example in § 5.3). Thus a similarity tree expresses the semantic similarity or adjacency of the feature categories, which can then be used for reclassifications or aggregations (van Smaalen, 2003). A special case is encountered if multiple output categories are possible for a given input category. However, in such a case these dependencies are no longer strictly hierarchical and a weighted decision graph or a transition matrix can be used instead (see 3.2.2).

Hierarchical network ordering: A well-known example for a hierarchy is the Horton-Strahler order of hydrological networks (Horton, 1945; Strahler, 1952). This is a very obvious and intuitive example to represent a river network as a tree structure from the outlet to the source links (except for a few exceptions such as braided streams). This ordering does not have to be returned by the support service as an explicit tree data structure but it could just be added to the attribute table of the river network.

Reactive data structures and Levels of Detail: For the efficient storage and access of line simplifications or polygon aggregations in MRDBs tree structures provide a useful hierarchy for the features involved. Van Oosterom (1994) proposed tree structures for on-the-fly generalisation

including the Reactive Tree for storing and retrieving map objects at multiple detail levels and the BLG tree for line approximation by the Douglas-Peucker algorithm in a multi-resolution environment. Van Oosterom and Schenkelaars (1995) further describe the GAP Tree, useful for polygon aggregation in polygonal subdivisions. For the generation of multi-resolution databases (MRDB) out of different datasets the *matching* of the features on the different Levels of Detail (LoD) is indispensable. The links between the features on the larger scale with the matched features on the reduced scale are mostly of nature 1:0, 1:1 and 1:n as shown by Timpf (1998). 1:1 and 1:n relations are partonomic relations (Bobzien et al. 2006) and can be expressed by a simple tree. These 'vertical' links (Neun et al., 2004) between the map features on different LoDs help the automated propagation of changes while updating the MRDB. However, n:m relations as they exist, for instance, in typification operations (e.g. 5 buildings are typified to 3 buildings) are more complex to model and require a DAG (directed acyclic graph) as they cannot be modelled by a tree.

Complex features and partitioning: Map features often form meaningful groups, that is, complex map features consisting of simple features. Examples include a cluster of buildings that form a hamlet; an alignment of similar buildings along a road; a group of buildings and surrounding streets forming a city block; or several fields, ponds, trails, etc. forming a park. Since complex features represent groups they are the simplest and also most general case of hierarchical (partonomic) relations. Often, however, complex features such as the above examples are not stored explicitly in cartographic databases. Thus, they have to be created either interactively by user definition or automatically by cartographic pattern recognition (e.g., clustering procedures). The knowledge about these partonomic relations informs the selection of appropriate generalisation methods and parameters in order to preserve the original structure of the map. Groups of map features can also be derived through spatial partitioning of map data into regular or irregular tiles. For instance, settlement generalisation is facilitated by partitioning the settlement into city blocks formed by the street network (Ruas 2000). Additionally, such partitions may also be exploited to increase the speed of complex generalisation operations by reducing the amount of features that have to be generalized at once. Also, the distributed processing of a partitioned dataset on multiprocessor computer architectures or even on clusters is possible (Burghardt and Neun, 2006).

3.2.2 Non-Hierarchical Relations

Common representations of non-hierarchical relations are graph data structures and matrices (see 4.3). The graph structures may contain cycles, are possibly weighted, and/or directed. The support services in this category describe the spatial and semantical relations between cartographic features in diverse ways.

Minimum spanning trees (MST): Although it is named a tree the MST does not express a hierarchical relationship between the features it connects. The MST can be used for a variety of purposes in generalisation. It has been used, for instance, for the selection of candidate roads in automated road matching for multi-resolution databases (Lüscher, 2005). Bader et al. (2005) use a *ductile truss* for managing building proximity relations in the building displacement process. This elastic truss connects the building centroids, adding further edges to a MST and forming cycles until every building is connected to at least two neighbours. Roads or railway networks form graphs as well. Such transport graphs can be used to generalise a road network connecting a set of cities by selecting roads in variants of the minimum spanning tree (Mackaness and Beard, 1993; Thomson and Richardson, 1995). Hence the connectivity of the transport network is assured.

Topology graphs: For expressing direct adjacencies between map features topological data structures are used. A topological data structure is an extended planar graph and uses nodes, edges and faces to represent the topological relations of map features. Thereby the space is subdivided completely by the nodes and edges of the map features. The use of such a topology graph in an algorithm ensures data integrity, shared boundaries and connected networks. For example, in generalisation topological rules (embedding, planar enforcement, etc.) can be used to prevent holes which are created due to a geometry type change from a polygon to a line or point (Bobzien and Morgenstern, 2002). A data format for interoperable storage and exchange of topological data structures is available through GML 3 (OGC, 2004).

Neighbourhood or proximity graphs: For expressing the relations of a feature with its neighbourhood in a more general way than with the pure topology structures such as a nearest neighbourhood graph or a relative neighbourhood graph can be used. Thus neighbourhood graphs can also express relations where the generalisation of a feature influences its surrounding features though they are not directly adjacent. Anders (2004) used neighbourhood graphs for the interpretation of spatial data, data analysis

and data enrichment of disjoint objects (e.g. buildings). For the establishment of neighbourhood graphs very often triangulations between the features are used. Regnauld (2005) proposes a triangulation and a proximity graph that extends the triangulation of feature centroids with the true distance between feature geometries (an application is shown in Fig. 12 below). This proximity graph can be used to derive clusters of close features but still maintains a relation between all triangulated features.

Triangulations and related structures: The *Delaunay triangulation* of a point (vertex) set (Fig. 5) is a collection of triangles satisfying the property that no other vertices are within each triangle's circum-circle. The constrained and the conforming Delaunay triangulations are adaptations of the Delaunay triangulation which can be used to triangulate over polygonal objects by incorporating the polygon edges as predetermined or constrained triangle edges.

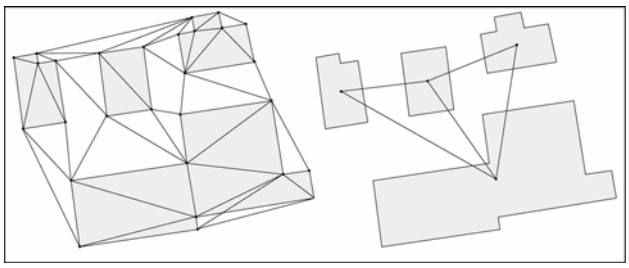


Fig. 5: Delaunay triangulation (non-constrained) of polygon vertices (left) and polygon centroids (right)

Jones et al. (1995) use the constrained Delaunay triangulation in their simplical data structure (SDS) for implementing exaggeration, collapse, amalgamation, reduction and displacement algorithms. Ruas (1998) uses a Delaunay triangulation of building centroids (see example Fig. 5 right) to represent proximity relations for managing the building displacement process. Thus triangulations can be used for expressing neighbourhood and proximity relations as shown by Burghardt and Cecconi (2007) for the typification of buildings.

The *Voronoi diagram* is the geometric dual to the Delaunay triangulation. The constituent Voronoi polygons are also known as Thiessen polygons defining the border of the space which is closer to the contained object (e.g. a point) than to any other object. Thus, the result is a complete tessellation of the space between the objects. Chithambaram and Beard (1991) use these properties for creating the *skeleton* of a polygon. The skeleton is of interest in generalisation as it represents the 'interior structure' of polygons. Hence, the skeleton – and more specifically, the straight line skeleton – has been used by several authors for purposes such as polygon aggregation in polygonal subdivisions (Bader and Weibel, 1997), collapse and partial collapse of polygons to lines (Haunert and Sester, 2004), and road centre line generation (Petzold et al., 2005).

Decision graphs: Strictly partonomic relations can be expressed by hierarchical trees. Partonomies with multiple associations, however, cannot be represented in a tree because of the possibility of cycles in the structure. As shown in Fig. 6 an alley could, depending on its size, become part of a city block or part of the road network. A weighted decision graph (or a weighted adjacency matrix) can express such relationships. Such a graph can also be represented as a directed acyclic graph (DAG) which has, if needed, weighted edges.

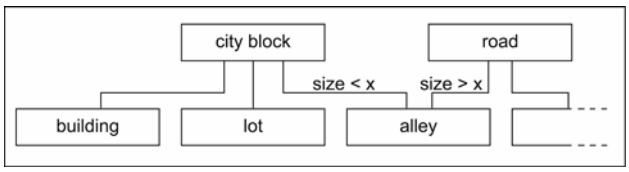


Fig. 6: Partonomy with multiple associations

The modelling of such context-dependent relationships often involves semantical as well as geometrical and structural properties. Context-dependent decision graphs have been used as a rule-base in model generalisation for transforming feature schemata (Bobzien, 1999), and also for the reclassification of features, preceding aggregation or amalgamation operations.

3.3 Summary of Support Services for Vector Data

Above, we have proposed and populated a taxonomy of support services to aid the generalisation of vector cartographic data. These generalisation support services are summarised in Table 1. Selected implementation examples of representative services are shown later in § 5.

Entity Support Services		**Relation Support Services**	
Creating Geometries	**Generating Attributes**	**Hierarchical Relations**	**Non-Hierarchical Relations**
- buffering - partitioning - creation of alignment lines - creation of inflection points	- calculation of line sinuosity - area - shape index - various cartometric measures - classification (§ 5.4)	- similarity tree or dendrogram (§ 5.3) - LoD links in MRDBs - network ordering - reactive data structure - complex features	- minimum spanning tree (MST) - transport graph - topology graph - neighbourhood graph (§ 5.1, § 5.2) - triangulations and related structures (§ 5.1) - decision graph (§ 5.3) - weighted matrices (§ 5.4)

Table 1: Support service types

While we argue that our taxonomy is comprehensive, it is by no means meant to be populated exhaustively. Besides providing an overview of the kinds of functionalities needed in support of generalisation operations, the main purpose of building this taxonomy is to allow to devise an appropriate set of data structures for support web services. Therefore, we tried to find commonalities of the data structures needed for the representation of the entities and relations involved. The output of the four support service types differs in terms of complexity and availability for representation in a data structure, for storage in a file format and for transfer over networks. Data structures for storing, exchanging and utilizing the supporting knowledge offered by support services are discussed in the next Section.

4. Storing, Exchanging and Utilizing Data from Support Services

Generalisation support services should deliver enriching data structures and information which are otherwise expensive to calculate or difficult to implement on the target platform. Thus they should facilitate the development of new algorithms by providing output from complex auxiliary algorithms, such as complex measures, auxiliary geometries or map feature relations, with an easy interface and in a preferably simple format. The aim is to make support services available to developers of service-based generalisation architectures in such a way that they can use these in conjunction with generalisation operator services without having to know in detail how the auxiliary data are generated. For support services that are complex and expensive to create a persistent data format is desirable in order to allow reuse of the supporting data structures in a workflow of generalisation operators in the web services environment.

4.1 Deploying Entity Support Services

The two classes of *Entity Support Services* are well covered by existing standards. Supporting entities are either geometries or additional attributes for map features. For the representation of geometries the Simple Features (OGC, 1999) are well known and widely used. Formats such as Shapefiles (ESRI, 1998) or GML (OGC, 2004) can be used for map features with attributes. Owing to these widely used formats the supporting knowledge in these entities can be exploited with every standard GIS.

4.2 Deploying Hierarchical Relation Support Services

For the representation of hierarchical tree-like relations between map features or between their properties standard geometries and attributes are usually not sufficient. Therefore other formats for representing and exchanging hierarchical structures are needed. XML is a very flexible format and has an implicit hierarchical structure. Thus it can be used for representing hierarchies directly. Elements of an XML data structure can enclose other elements. Hence they are creating a nested structure, which represents a strict hierarchy. Such an XML representation can also be used in an object-oriented manner in the main memory of a computer. In doing so almost all XML parsers for object-oriented programming languages are offering a tree-like data structure for querying the content of an XML document. Thus XML offers a format for storage and network transfer as simple text and, when using an XML parser, an object-oriented format that can be queried, modified and kept in main memory.

A similarity tree can be modelled using the hierarchical nested structure of XML. Listing 1 shows the simple code example of different forest types which belong all to the parent class 'Forest'.

```
<webgen:Hierarchy>
    <webgen:Node value="forest">
        <webgen:Node value="deciduous forest" />
        <webgen:Node value="coniferous forest" />
    </webgen: Node >
</webgen:Hierarchy>
```

Listing 1: Simple XML similarity tree for reclassification

The relations between the different classes can easily be queried by requesting the parent or child nodes of the XML tree. Thus when querying the XML document with a standard XML parser the node "deciduous forest" has a pointer to its parent node "forest". An application example of such a hierarchy is given later in § 5.3.

4.3 Deploying Non-Hierarchical Relation Support Services

The available output formats for *non-hierarchical* relation support services are much more heterogeneous. Non-hierarchical formats require other, more complex (and diverse) data formats. In GML 3 (OGC, 2004) a format for topology graphs is available. Ways for representing other non-hierarchical relations such as triangulations using graphs or matrices are discussed in this section.

Non-hierarchical relations can be represented by graph-like structures. In a basic object-oriented representation (Fig. 7) a graph consists of a list of nodes and a list of edges. Each node contains only a list of the incident edges. The edges have two variables, which point to the two end-nodes of the edge.

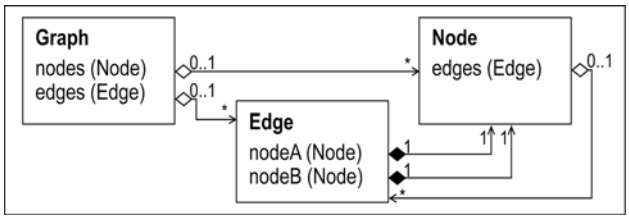


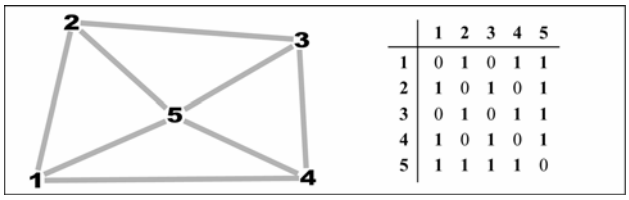
Fig. 7: Basic object-oriented graph representation

For its use as a generalisation support structure for different sorts of non-hierarchical relations such a basic graph can be easily extended and specialised through class inheritance. In graphs that represent cartographic features, where the nodes correspond to nodes or vertices of cartographic features, the nodes have coordinates and/or contain a link to the corresponding cartographic feature. For planar graphs like topology graphs or triangulations the basic graph model can be extended with faces or triangles. An example of such an extension for Delaunay triangles is shown by Regnauld (2005).

This in-core representation is optimized for the querying and modification of the graph but it can only be used in the main memory of a computer. Thus a transfer data format is needed that is optimised for efficient storage, transfer over a network and parsing on the receiving system.

Non-hierarchical relations can also be represented as matrices relating every feature to all others. A matrix is basically the dual of a graph; every graph can be expressed by a matrix and vice versa (see Fig. 8). But using a matrix for representing a graph with many nodes but only few edges per node is very inefficient in terms of data size. This is because in a matrix every possible edge, also the ones that

do not exist, is represented. Thus for a complete graph or another structure where every feature is related to all others also a matrix representation is an effective in-core representation.



	1	2	3	4	5
1	0	1	0	1	1
2	1	0	1	0	1
3	0	1	0	1	1
4	1	0	1	0	1
5	1	1	1	1	0

Fig. 8: Simple graph and matrix representation

More efficient data formats or languages for storing and exchanging graphs are for example very simple text files containing node and face (i.e. triangle) lists (Shewchuk, 1996) or the XML based Graph eXchange Language GXL (Holt et al., 2000). GXL is very expressive and tries to represent a maximum of different graph types. For representing specific graphs also own tailored list formats using e.g. XML can be created by representing all edges as tuple of the identifiers of their two end-nodes. All these serialised data formats are based on a list representation such as an adjacency list for simple graphs or a triangle list with associated nodes and possibly their adjacent triangles. The advantage of these data formats is that they are compact, they do not contain much redundancy and they can easily be saved in a file or be sent over a network using standard protocols.

However, using these list representations as an in-core data structure for graph representation is not very practical and efficient. An object-oriented data structure is much easier to query, modify and extend. Thus when deploying complete graph structures with web services using XML a conversion between the in-core representation and the transfer format has to be carried out and vice versa.

Also matrices can be represented using XML structures. Usually in this case table-like XML representations are more compact and better to parse than the edge lists used for representing graphs. Such a matrix representation in XML can even be exploited directly by traversing the XML structure without having to parse the complete structure.

In the WebGen research platform XML and HTTP are used for the data exchange between the different services and the client, making the usage of an XML based graph data format straightforward. As there exists within GIS no specialised standard format for representing graphs, a data format which can easily be created and read in different programming environments or GIS platforms is desirable. Thus, parsers for the different generalisation support data structures are needed. In the case of an object-oriented representation of the graph on the computer the data structure is converted to an XML structure, transferred over a network and parsed on the receiving system into its own, internal data structure, which may not necessarily be the same as the transfer format. In the WebGen framework we are using an XML representation for the transfer of graphs which is based on a redundancy free adjacency list. This XML representation can be parsed efficiently into an object-oriented model and vice versa. As there is no redundancy the data volume depends directly on the number of nodes and edges. This ensures a fast network transfer of the graph. For transferring a matrix of decision borders (see § 5.4) a table-like XML structure was used and proved to perform well.

4.4 Stateful Handling and Deployment of Support Services

If multiple generalisation services are used in a workflow when all the data has to be passed from one service to another and if a particular service is subsequently used with the same data but only slightly changed parameters, this will cause massive transfers of data over the network. Thus, instead of using normal stateless web services a stateful approach can be helpful. Stateless services have no memory to remember preceding calls and their parameters and data. Conversely, a stateful service creates a session for every user where settings, parameters and data can be saved and retrieved when the service is called multiple times.

The advantage of *stateless services* is their simplicity. The service has only one input source for parameters and data. The data which is transmitted may consist of simple feature data but also of any other data structure which has for instance been recalculated by another support service. However, the amount of data which has to be transmitted upon every request can be quite high. Thus the stateless services are more suited for testing environments where the datasets are rather small. When calling a support service the result may, for example, be a complex graph structure in an XML format as described in § 4.3. Such a representation may be used by other services. To query and extract information from such an XML format is usually simple and can be handled on the client. However, if

the graph has to be modified also the logics to insert, delete and change nodes must be available on the clients.

A *stateful support service* remembers old data or parameters and can access and use them upon being called again. Such a service creates a session for every calling client. Upon a new call old settings and data structures can be requested and reused. For example a triangulation service can retain the triangulated structure persistently and the triangulation may be queried or modified again later. Thus the calling client calls the support service once with the input data and parameters. After this initialisation step the generated data structure is kept persistently on the server. Later on the calling client may query the data structure, for example, by retrieving the whole graph or by only demanding a single edge and its neighbours. The calling client may then also send requests to insert, delete or modify a node. The functionalities of such a stateful support service fundamentally depend on the functionalities of the implemented support data structure.

Alternatively the parameters and data may also be kept persistently in some sort of more generic stateful service where other services residing on different servers can access it. When dealing with large datasets and when a service chain is involved with a workflow of services that are executed sequentially a stateful environment avoids network traffic and removes load from the client. In this scenario a client may be either the calling user program or another service which would then be a workflow service which is triggering the single services on the service chain. Such a stateful service could be an extended, transactional WFS which would then store control parameters, map data plus supporting data structures. All algorithms in a particular workflow could then access this store and retrieve the data they need for their execution, subsequently updating the store.

In Fig. 9 an example sequence for the use of a stateful transaction server for spatial web services is shown. Here the session is initialised once and the data (map features and other parameters) which will be treated in the following workflow are uploaded. After that the data can be accessed and modified by other services. Hence the transaction server is basically an extended network-enabled data model for storing feature data and supporting data structures. The advantage is that during a generalisation process not all the data have to be passed from one generalisation service to another. With this approach the individual services are only retrieving the data they need, storing their results back on the transaction server.

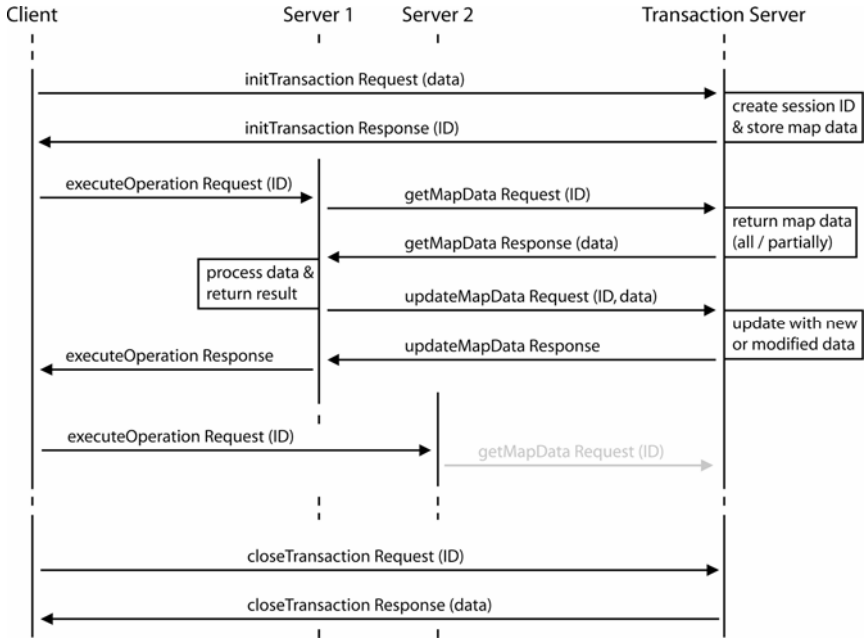


Fig. 9: Transaction server architecture for support data structures. Requests of the second server are set in gray font or omitted.

In collaboration with other stateful support services this transaction server can also act as an authority to manage the data seamlessly by providing unique identifiers and matching tools to merge enriched data from different sources.

Some of the implementation examples presented in following section are implemented as stateful support services. These include a triangulation, a proximity graph, alignment groups, and a matrix that are persistently modelled in the service and can thus be queried and modified.

5. Implementation Examples of Support Services

In this section selected examples of some of the generalisation support services which are implemented in the WebGen framework are explained. These examples demonstrate possible usage scenarios and benefits of using support services. They have been selected so as to provide a representative sample of the functionalities and data structures employed by the proposed taxonomy of generalisation support services.

- As an example for non-hierarchical relations a stateful service for triangulations and proximity relations is shown. This service is subsequently used by a building typification and a building displacement algorithm, thus demonstrating the collaboration of support services and operator services (§ 5.1).
- As an example for structural relations expressed by non-hierarchical relation a service for the detection of building alignments is shown. This service can be queried for alignment relations and can return aligned buildings as groups or complex features (§ 5.2).
- As an example for spatial and semantical context represented hierarchically or as a decision graph a service for the reclassification of features and their schemata is described. This service creates and changes attributes and can be used for aggregating land cover data. Additionally a conditional (non-hierarchical) decision graph for these schema transformations is presented (§ 5.3).
- The final example presents a service for supervised classification of buildings into urban structure types. The building classifier is implemented as an entity support service returning attributes assigned to each building feature. As it is a supervised classifier, this service in turn calls a training service returning a matrix representing relations between building shape measures (§ 5.4).

5.1 Proximity Relations for Building Generalisation

In this section we present a stateful support service serving triangulations and further proximity relations. The proximity graph extends a Delaunay triangulation of the centroids of map features (Fig. 10) by additional measures for the 'real' distance between two geometries (Regnauld, 2005). Thus for every edge in the triangulation linking two map features an additional edge in the proximity graph exists. In Figure 12 a proximity graph between buildings as well as buildings and roads is shown. In order to make the combined proximity and triangulation graphs available to other services they are implemented as a stateful support service representing the graph persistently and offering functionalities to query and modify the graph structure. This support service will be used subsequently by two operator services, a typification and a displacement algorithm for buildings. Hence, these two operators are first briefly explained.

The typification algorithm adopts the idea of Burghardt and Cecconi (2007) and uses the Delaunay triangulation of the building centroids supplied by the support service (Fig. 10) for deriving which buildings are closest to each other. In an iterative process this shortest edge is then collapsed and the buildings are replaced by a placeholder. After that collapse the triangulation is updated and the new shortest edge is derived. This process continues until the desired number of buildings has been replaced. The edges of the triangulation are weighted according to the size of the two buildings they connect. Thus an edge between two small buildings is shorter (i.e. has a lower weight) than an edge between two large buildings and the two smaller buildings are replaced first by a placeholder. For generating the placeholder, the building with the larger area of the two collapsed buildings is chosen, enlarged in order to meet the same black-to-white ratio and positioned at the centroid of the two buildings (Fig. 11). This method works well in semi-dense, suburban and rural areas. In other areas an amalgamation algorithm could be used instead, for example.

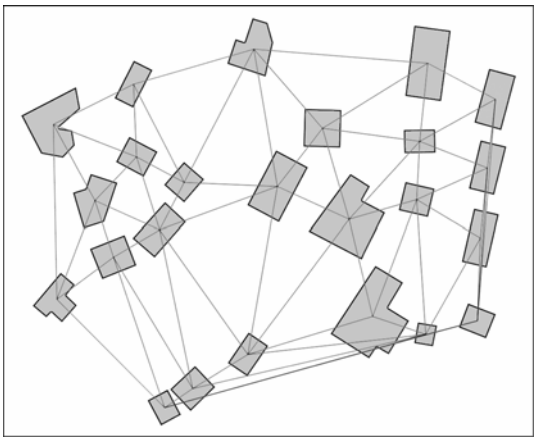


Fig. 10: Triangulation of building centroids (reproduced by permission of swisstopo, BA071153)

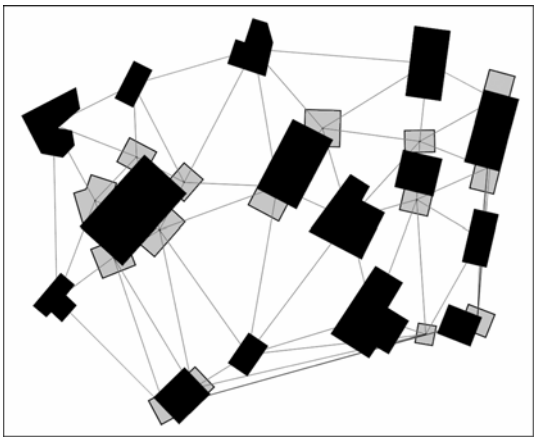


Fig. 11: Typification by collapsing shortest edges (reproduced by permission of swisstopo, BA071153)

The algorithm for building displacement which we developed uses the proximity graph from the triangulation service. This displacement algorithm aims to achieve sufficient distances between the map features so that they are distinguishable and do not visually coalesce. The algorithm can respect different required minimum distances between buildings and between buildings and roads. In an iterative process all edges which are shorter than the minimum distance are requested from the proximity graph. The edge is elongated to the minimum distance which pushes the buildings away from each other. If the proximity edge is between a building and a road segment only the building is moved. After making these modifications the iteration starts again with requesting the edges which are too short. If multiple buildings are very close to each other this process can last a couple of iterations as the buildings once pushed away from each other get again displaced by other buildings. The process stops when no more edges are found that are too short. If not enough space is available to sufficiently separate all features a deadlock may occur. Therefore a maximum number of iterations can be defined. If this limit is reached, the number of remaining short edges and their length can be used to evaluate whether the displacement was sufficient or whether the number of buildings must be reduced e.g. by a further typification operation. The result of this displacement operator is shown in Figure 13.

Fig. 12: Proximity graph before displacement, (road symbols not to scale; reproduced by permission of swisstopo, BA071153)

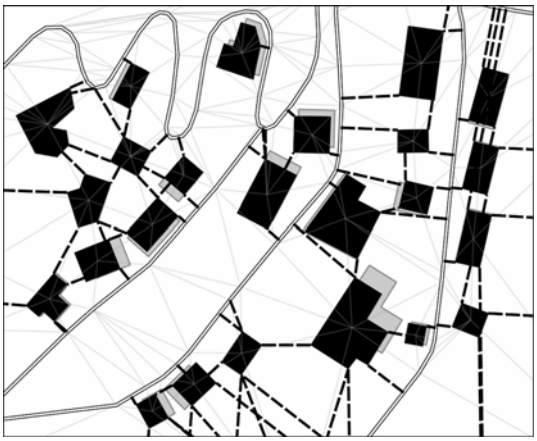


Fig. 13: Proximity graph after displacement (road symbols not to scale; reproduced by permission of swisstopo, BA071153)

The WebGen framework allows the use of multiple services with each other. In our implementation the two operator services typification and displacement can use the same proximity graph support service for their purposes. This sequence will now be described shortly.

Before the two operator services can use the stateful support service it must be initialised (see also Fig. 9). Therefore, both building and road features are sent to the support service. Internally the service creates the triangulation and proximity graph data structure and saves it persistently using a unique identifier. The two operator services can access now the graph using this identifier. Typically, the number of the buildings is first reduced by typification, followed by a displacement of the buildings to meet the legibility constraints of the minimum separation distance.

As described above the typification service requests the shortest edge between two buildings from the triangulation support service using the identifier to access the correct graph. As the triangulation and proximity graph contains also the roads there could also be shorter edges between buildings and roads but these will be ignored. Thus, merge operations between two buildings across a road are prohibited. The two buildings connected by the shortest edge are merged to a new placeholder. Then requests to delete the two old nodes from the triangulation and to insert the placeholder are sent to the support service. This process continues iteratively until the desired number of buildings have been replaced by placeholders.

After the typification the displacement service uses the same triangulation and proximity graph service. Therefore it uses the same identifier to access the service. As described above the displacement service retrieves a list of all proximity edges that are to short from the support service. The proximity edges are between buildings as well as between buildings and roads (Fig. 12). All the short edges are elongated by changing the positions of the buildings. Roads remain in their positions. The changes are committed back to the proximity graph service which keeps the graph structure up to date.

Finally the buildings represented by the triangulation and proximity graph can be sent back to the calling client who triggered the typification and displacement workflow. The initial triangulation with its proximity graph is shown in Figure 12. The final result is shown in Figure 14.

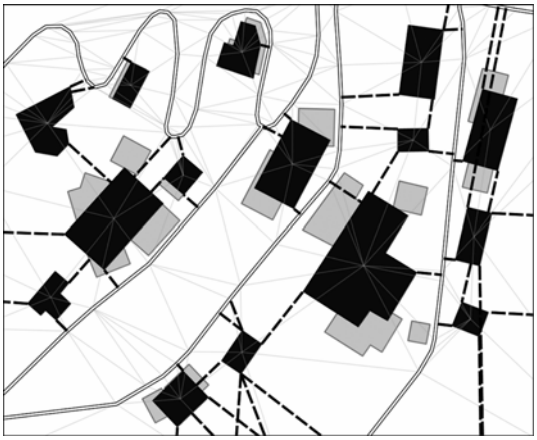


Fig. 14: After typification and displacement (road symbols not to scale; reproduced by permission of swisstopo, BA071153)

This approach of a stateful triangulation service shows how multiple transfers of massive amounts of data between services can be avoided. The building and road features are only transmitted once to the triangulation support service and can be used by multiple operator services. Through the continuous query and update calls this data store remains synchronized. In a scenario with several processing and evaluation steps this same support service can be used throughout the entire workflow. The proximity relations need only be established once and are then continuously modified.

5.2 Detection of Building Alignments

Another enriching structural knowledge are alignment relations as they exist, for example, between buildings. Alignments basically express a special neighbourhood relation. As an algorithm for the detection of alignments an approach inspired by Burghardt and Steiniger (2005) is implemented in the WebGen framework. This algorithm uses principal component analysis to derive the homogeneity of building alignments along roads. Therefore the candidate buildings along every road are characterised using measures about their size, shape, orientation and about their group characteristics (distances to neighbours etc.). Figure 15a shows the homogeneity of building alignments. The darker the colour of the building the more homogeneous is the alignment the building belongs to. Certain buildings such as corner buildings may also be part of several alignments.

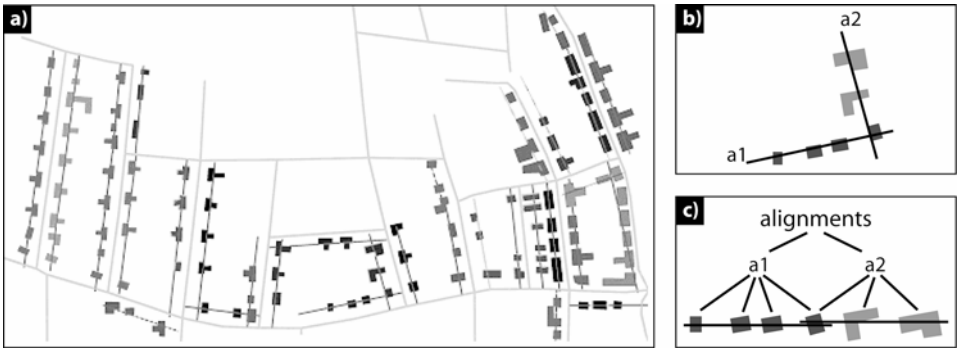


Fig. 15: a) Building alignments (darker buildings are more similar and more strongly aligned). b) Corner buildings are shared by two alignments. c) The root node links all alignments in one data set together. (Reproduced by permission of swisstopo, BA071153)

The knowledge about these alignments may be represented in several ways. Probably the simplest way would be that the alignment support service delivers each alignment group as a separate set of buildings. Another possibility would be to assign an attribute to the original buildings containing e.g. an identifier for each alignment group and its homogeneity value. These two approaches have the disadvantage that they loose important information such as that a building can be part of more than one alignment. Thus to regain this lost knowledge costly matching procedures might become necessary afterwards.

Using a graph-like representation that is structurally comparable to proximity graphs does not have this disadvantage. Every building can be linked to its neighbours in the same alignment(s). Such a graph can be traversed starting at the root node. This allows a complete representation of all buildings in a data set with their alignment relations. Corner buildings are instantaneously recognisable as they are part of two alignments. An example can be seen in Figure 15b, where the two alignments ‘a1’ and ‘a2’ share a corner building.

The support service for the detection and management of building alignments is implemented as a stateful support service. This service builds up an internal data structure with buildings as nodes, and the alignments as their parent nodes. The root node links all the alignments in one data set (Fig. 15c). This alignment support service can be initialised with buildings and roads. Once initialised, several possibilities exist for query and modification. It is possible to retrieve all alignments as separate groups but it is also possible to query the neighbours of a single building. For example, the typification algorithm (§ 5.1) could be extended to check whether a building is part of an alignment. Also new buildings can be inserted and their alignment relations with the other buildings established. Hence this support service offers both, a simple possibility to merely retrieve alignment groups and more complex capabilities to query individual properties or to modify the alignment structure. This non-hierarchical alignment data structure provides structural knowledge that is flexibly usable throughout a context-dependent generalisation process, restraining for example the typification and displacement of buildings in order not to destroy the existing patterns.

5.3 Attribute and Schema Transformation Services

When generalising maps very often the attribute schemata between the source and the target scale are changing. Attributes may be removed or combined on the target scale, or attribute values may obtain a different domain. A typical example for such an attribute change is the change of feature classes as it occurs when generalising land cover data. According to Hampe et al. (2003) a generalisation process for land cover consists of three steps: reclassification (schema transformation) of the object types; aggregation of adjacent areas with equal object type; and shape generalisation.

The reclassification needs some sort of rule base to inform the assignment of the input features to the target classes. This rule base in many cases is a strict hierarchy that may be represented by a similarity tree which assigns a target class to every input class (e.g. 'deciduous forest' and 'coniferous forest' are both reclassified to 'forest'). In more complex cases the rules must respect the semantical, spatial and topological context. These reclassification rules form a non-hierarchical decision tree which assigns a target class for example only if the feature has a certain attribute, a certain size or is adjacent to a certain feature class (e.g. 'deciduous forest' only becomes ‘forest’ if it is large enough to be visible on the target scale or if it is adjacent to another ‘forest’). Owing to the different requirements, the provision of reclassification rules should also be made by different support services, that is, services for

hierarchical static similarity trees (§ 5.3.1) and services that use a context-dependent decision graph (§ 5.3.2).

5.3.1 Hierarchical Reclassification using a Similarity Tree Support Service

A similarity tree or reclassification hierarchy can be defined by the user of the system or generated automatically, for instance by a statistical evaluation of the properties of the input categories to establish their relevance as it was done by Fuchs (2002) using multivariate methods for deriving a statistical aggregation hierarchy of soil types. Thus two types of support services for providing such a similarity tree are possible. The more complex service generates the similarity tree out of the raw data which it receives as input. The simple service just provides a static similarity tree which has been defined by a user. A similarity tree can be modelled using the hierarchical nested structure of XML (§ 4.2). Such an XML file (see Listing 1) can easily be queried in order to perform a reclassification. Typically the reclassification service iterates over a set of input features. For every feature it derives the parent class of the current feature class (XML query getParent()) and assigns it to the feature. Hence every 'deciduous forest' or 'coniferous forest' feature would become 'forest' after that step. The feature classes are usually stored as attributes with the land cover features. Thus, for every feature the reclassification changes the value of this class to its parent class. For a typical reclassification service this similarity tree could be requested as an XML hierarchy from a support service, defined either by a user or generated automatically. For user-generated similarity trees the support service acts just as file server, delivering the static hierarchy. Automatically generated similarity trees are inferred from the input data using e.g. statistical methods.

For testing purposes two examples have been implemented. First, a static similarity tree support service which serves an XML hierarchy (see Listing 1) to reclass land cover data from 28 different classes to 12 user-defined classes. Second, a dynamic similarity tree service that generates automatically a hierarchy based on the distribution of the feature classes. For distribution based reclassification three methods have been implemented: quantiles; preserve small classes; and preserve large classes. These methods are just simple examples for testing purposes. More advanced methods like the Hierarchical Clustering proposed by Fuchs (2002) or statistical methods like natural breaks or quantiles could be implemented the same way to generate a similarity tree. It is not only possible to retrieve the whole similarity tree but also to query the corresponding parent class for a given class. This service can be used if only few features are processed. So, other services can request similarity tree information without having to care about the decoding of the XML hierarchy and without having to download the whole similarity tree.

Figure 17 shows the result of the reclassification and merging step. In the reclassification step the original 28 land cover classes (Fig. 16) are reduced to 12 classes. The blank areas are not empty; they are occupied by the class 'other'. In the merging step adjacent polygons having the same class were combined.

Fig. 16: Land-cover data 28 classes (reproduced by permission of swisstopo, BA071153)

Fig. 17: Land-cover data 12 classes (reproduced by permission of swisstopo, BA071153)

5.3.2 Non-hierarchical (Conditional) Reclassification and Attribute Schema Transformation

In the reclassification example above a pure hierarchy is used to modify the class attribute of every feature. In this example a more complex approach is demonstrated which can change feature attributes, including the class attribute, based on rules with conditions.

As can be seen in Figure 17 some of the small features are assigned to the 'other' feature class but it would have been better if these features got assigned to the class of a neighbouring feature. Thus, in order to perform reclassifications or other processes where attributes change or features get assigned a new class or merged, a rule-based approach can be helpful. Figure 18 shows a simple decision graph with some condition checks which decide the assignment of a new feature class.

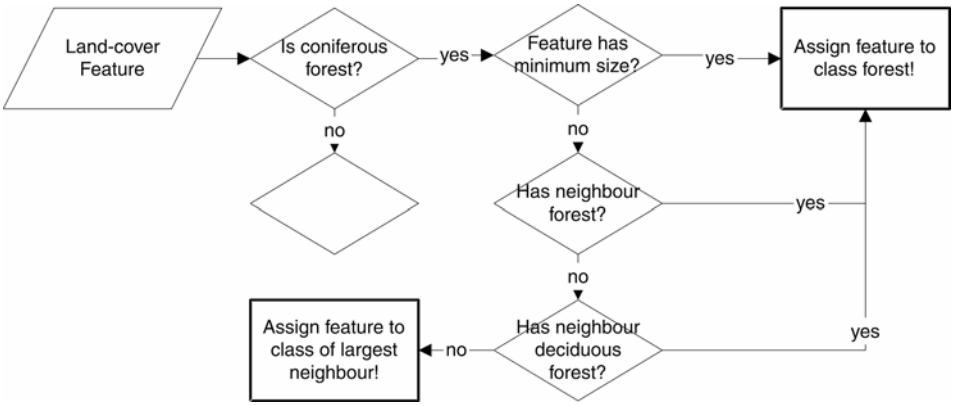


Fig. 18: Conditional decision graph

Transformation rules for feature classes and attributes can be subdivided into semantical, geometrical and structural (topological) rules (Bobzien, 1999). Semantical rules are simply checking attribute values of the source features, whereas geometrical and structural rules are evaluating properties and the context of a feature. In the WebGen framework the geometrical and structural properties are provided as measures by support services.

The implemented example of a rule base service is used for the conditional reclassification. Therefore it is intended to apply transformation rules onto the supplied features. Bobzien (1999) proposes therefore a rule syntax which can be seen in the rules column of Listing 2. These transformation rules are loaded into the support service where they are decoded and applied.

source class	*target class*	*attribute*	*target value*	*rules*
coniferous forest	forest			#size >= 200
coniferous forest	forest			#touches 'forest'
coniferous forest	forest			#touches 'deciduous forest'
coniferous forest	farmland			#size < 200 AND #touches 'farmland'
	farmland	cultivation	coniferous	
coniferous forest	other			#size < 200 AND #touches 'other'
…	…	…	…	…

Listing 2: Conditional reclassification and attribute schema transformation rules

Geometrical and structural properties here are preceded by a pound sign. Currently simple geometrical properties such as #size, #length or #width are implemented in WebGen as well as the structural property #touches. Rules that change the class like the transformation from coniferous forest to farmland can be followed by additional rules and assignments of attributes from the source onto attributes of the target schema. Similar conditional services could also be of use for schema transformations in model generalisation or for a building typification algorithm which respects e.g. semantical rules for the generation of the placeholder.

5.4 Urban Structure Classification

An example for a very costly support service, implemented in the WebGen framework, is the classification of buildings according to their urban structure. In Steiniger et al. (in press) five types of urban structures were defined. The knowledge whether a building is in an industrial and commercial area, in a inner city area, in a urban area (dense buildings), in a suburban area (dispersed buildings) or in a rural area (single buildings) can be used for different purposes. Examples are different colourings or the amalgamation of buildings in dense areas and the preservation of single buildings in less dense areas (see examples in Fig. 19).

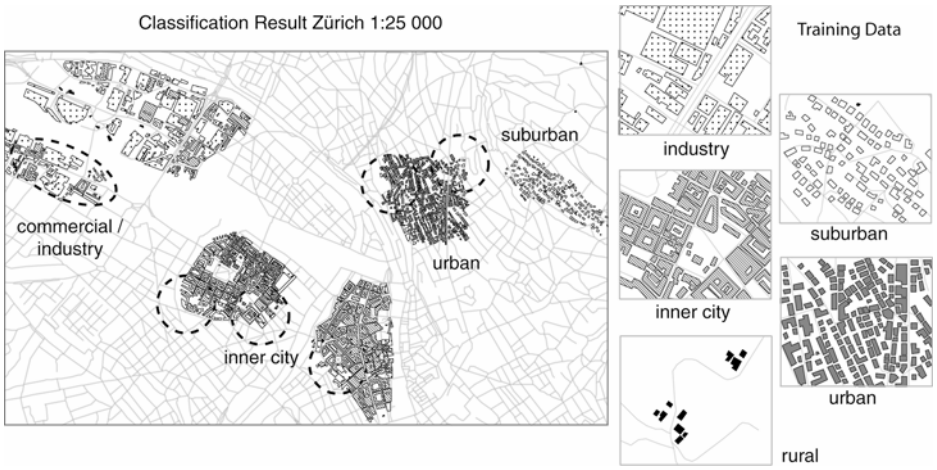


Fig. 19: Urban structure types (Steiniger et al., in press; reproduced by permission of swisstopo, BA071153)

5.4.1 Classification as Enriching Attributes

The approach of Steiniger et al. (in press) for the classification of urban structures uses solely geometrical and perceptual measures of buildings (see process sequence in Fig. 20). Morphological measures that are used include the building area; the number of building corners; the building shape; the building squareness; the building elongation; and the number of courtyards. Relational measures encompass the number of buildings intersected by a buffer; the building area in relation to the convex hull area of the buffer intersecting buildings; and the relation of the building area to the buffer area of all intersecting buildings. When using all these measures for characterising the buildings they are spanning a 9-dimensional space (one dimension per measure) and the borders separating the urban structure types must be defined in this full 9-D feature space.

The implemented classification support service uses the Batch-Perceptron approach (Duda et al., 2000) to derive the separating decision borders (thresholds) using training data where the urban structure types are known. These thresholds are represented as a matrix which describes the relations between different shape measures of the building features. Following this training step in the classification step the buildings that should be classified are first characterised applying the measures. Then the urban structure types are derived using the decision borders and assigned to the building features as attributes.

The advantage of this service is its relative simplicity. Training data and the data to classify have to be sent to the service. As result the data to classify are sent back with two attributes assigned to every feature denoting its urban structure type and the accuracy of this classification. This enriching information can easily be used by various generalisation operators. The disadvantage is that especially the training process in order to derive the decision borders is time-consuming.

5.4.2 Separate Characterisation, Training and Classification

In order to increase the flexibility and also the performance of the extraction of the urban structure types the whole process was subdivided into components. The characterisation of the training data using measures and the derivation of the decision borders in the training process has to be carried out only once and the result can be reused. The characterisation of the input data using the measures has to be done for every dataset which has to be classified. During the characterisation process a number of measures are calculated for every feature positioning it with its characteristics in a multi-dimensional space. The calculation of every measure can be accomplished by a separate support service (Fig. 20) which takes the dataset as input and delivers a list with the measures for every feature in the dataset.

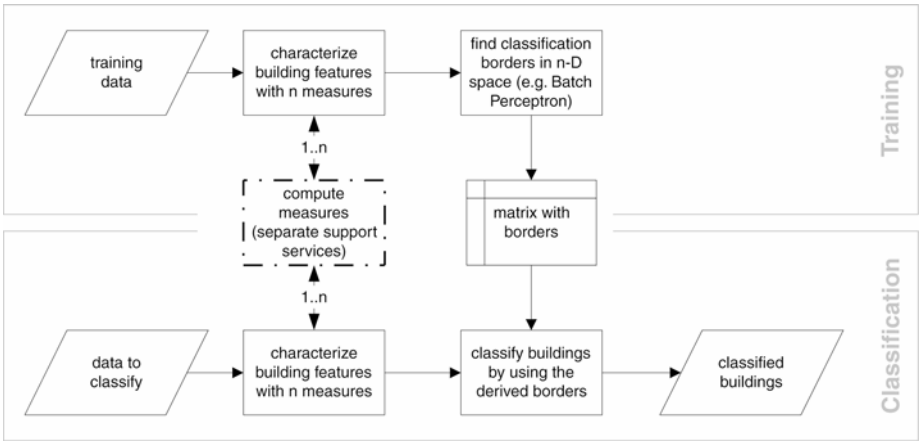


Fig. 20: Classification steps (training, characterisation and classification)

The training and classification processes can both work with an arbitrary number of dimensions of the feature space. Thus, due to the flexible service architecture it is possible to dynamically select the measures that are used in the characterisation, allowing to tune the result of the classification. Some measures like the perceptual measures are quite heavy to calculate. Support services for the measures can reside on different servers and the load can thus be distributed for parallel execution.

The output of the training process is a matrix. The number of columns of this matrix is simply the number of measures that are used to characterize the features. The number of columns is the number of all possible combinations of two different classes (10 in the case of 5 urban structure classes). This matrix is represented and stored like a table using XML. The training support service, implemented as a stateful service (see § 4.4), can store the matrix with the decision borders persistently. The classification service itself requests this matrix and performs the classification. Thus the training process has only to be done once for a specific data type and can then be used various times, saving a lot of time. Together with the possibility to carry out the characterisation of the buildings in parallel on separate servers this offers a great performance improvement potential.

6. Discussion

Many algorithms for enriching data or creating spatial relationships are only available on different standalone platforms and data formats. Thus interoperability and reusability are not ensured, the different implementations of algorithms are not comparable and implementations get even lost with time, particularly in academic research environments (Regnauld, 2005). The development process of advanced generalisation operators takes more time because algorithms or converters for auxiliary data structures have to be generated first.

It is important to note that today most supporting data structures are only generated at runtime and not saved persistently. Some generalisation support structures are very expensive to calculate but could easily be saved persistently for multiple uses, for instance, by other generalisation operators. The approach of stateful support services such as the triangulation service presented in § 5.1 reveals that due to the large amount of calls to a service performance problems might occur due to network latencies. Thus it must be evaluated for every generalisation process at which degree of granularity the subdivision into smaller or larger calls and components is appropriate.

Another support structure which is usually not saved persistently and only visible inside a process are trained models produced by machine learning approaches. Support services could also be used to keep trained models persistent and make them available to other services. However, this must be done with regard to the learning approach used and to the portability of the information learned.

The aim of this paper was to illustrate the possible usage of generalisation support services as web services. The WebGen framework implementation shows a great potential for enabling the interoperable, flexible and reusable deployment of generalisation algorithms. Many of the concepts do not only apply to generalisation but also too many other tasks where spatial data are used in a distributed, heterogeneous network environment. The WebGen architecture is not intended to be a web service for managing, browsing and exchanging data over the Internet but a *processing service* for providing algorithm logics and processing power to be executed remotely. It is, in contrast to other commercial or non-commercial GIS geo-processing servers, not limited to a (proprietary) GIS

platform. The usage of an open protocol and standards such as XML and HTTP enable the interoperable coupling of service providers and consumers.

The use of web services for providing supporting data structures might not always be appropriate: when working with extremely large data volumes or high frequencies of service invocations the process performance might be poor due to network latency, bandwidth and the time needed for creating and parsing the XML messages. For a single service use always the whole data has to be transferred from the calling client to the service and back. For multiple service invocations a stateful approach like the transaction server (see §4.4) can reduce the effectively transferred amount of data.

Developers wanting to use a supporting data structure do not have to re-implement or integrate source code of others; they still need to write a parsing routine to read the XML graph structure, for example. This is clearly a hurdle that must be taken, but as the web services are respecting common interfaces, different support services providing different graph structures, for instance, can then be used without having to recreate the entire parser.

Standardized functionalities to compute and use spatial and structural relationships are sparse in GIS packages, particularly in commercial ones. Furthermore, geographic databases or data transfer formats do not commonly include the modelling and storage of the advanced relationships discussed above. We proposed ways to use this knowledge in a web services environment using XML. What is still missing today, however, is an agreement in the generalisation community on how to exploit these XML-based formats to achieve a standardised data format for representing the generalisation support structures, many or most of which are graph data structures. Such a format can be used in the development of new generalisation support services and possibly also converters for already existing support structures could be developed. Following the categories of generalisation services (see § 2.4) the generalisation support services can then form the foundation of the more advanced generalisation operator and generalisation process services.

7. Conclusion

We distinguish three types of web services necessary for a comprehensive service-based generalisation architecture, generalisation support services, generalisation operator services, and generalisation process services. In this paper, we dwelled primarily on the generalisation support services, which are intended to enrich the raw cartographic input data with additional information such as shape or importance attributes, new geometries, as well as spatial and structural relationships, hence providing indispensable support to the other two service categories. These support services could be equally useful in a more general, non-generalisation environment, particularly for the purposes of spatial analysis and decision support.

As a first contribution, this paper delivered a comprehensive taxonomy of generalisation support services in relation to different generalisation operators. Second, methods were proposed to represent, store and exchange the spatial relations generated by support services, considering the special requirements of distributed architectures. Many relations can be expressed in a graph-like form. Thus, the proposed data structures and formats are mainly graph based. Third, we have presented implementation examples of generalisation support services in the WebGen generalisation service framework. It was demonstrated how generalisation support services can interact with generalisation operator services, delivering complex data structures to complex algorithms. Obviously, as mentioned in the Discussion, there are still plenty of open problems remaining, as the delivery of generalisation capabilities is revisited, facilitated by a different architecture (service-based rather than standalone) and at least partially given more stringent constraints w.r.t. processing efficiency.

We plan to address the above issues in the future by continuously extending the WebGen platform. Given the extensible nature of the service-based approach the WebGen framework lends itself to the collaboration with other researchers, which has occasionally already happened. Currently, attempts are made to combine support services and operator services with process services, in order to achieve automated control of the generalisation workflow.

Acknowledgements

This research was partially funded by the Swiss National Science Foundation (grant 20-101798, project DEGEN). Thanks go to Alistair Edwardes, Ingo Petzold and Matthias Bobzien for helpful comments.

References

1SPATIAL, 2007, Radius Clarity, http://www.1spatial.com (accessed 03/2007).

ANDERS, K.-H., 2004, Graph-Clustering-Verfahren zur Interpretation raumbezogener Daten. PhD thesis, Institute of Photogrammetry, University of Stuttgart.

ARMSTRONG, M. P., 1991, Knowledge classification and organization. In Buttenfield, B. P., McMaster, R. B. (eds), *Map Generalization: Making Rules for Knowledge Representation*, Longman, 86-102.

BADER, M. and WEIBEL, R., 1997, Detecting and Resolving Size and Proximity Conflicts in the Generalization of Polygonal Maps. *Proceedings 18th International Cartographic Conference,* Stockholm (S), 1525-1532.

BADER, M., BARRAULT, M. and WEIBEL, R., 2005, Building displacement over a ductile truss. *International Journal of Geographical Information Science*, 19, 915–936.

BARRAULT, M., N. REGNAULD, C. DUCHÊNE, K. HAIRE, C. BAEIJS, Y. DEMAZEAU, P. HARDY, W. MACKANESS, A. RUAS and R. WEIBEL. 2001. Integrating multi-agent, object-oriented and algorithmic techniques for improved automated map generalization. *Proceedings of XX Int. Cartographic Conference*, Beijing, pp. 2110–2116.

BOBZIEN, M., 1999, Implementationsaspekte der Modellgeneralisierung, In *Mitteilungen des Bundesamtes für Kartographie und Geodäsie*, Vol. 3.

BOBZIEN, M., BURGHARDT, D., PETZOLD, I., NEUN, M. and WEIBEL, R., 2006, Multi-Representation Databases With Explicitly Modelled Intra-Resolution, Inter-Resolution and Update Relations. In *Proceedings of the American Congress on Surveying and Mapping, AutoCarto*.

BOBZIEN, M. and MORGENSTERN, D., 2002, Geometry-Type Chance in Model Generalization - a Geometrical or a Topological Problem?. In *Proceedings of the Joint ISPRS/ICA Workshop on Multi-Scale Representations of Spatial Data*, Ottawa.

BURGHARDT, D. and CECCONI, A., 2007, Mesh Simplification for Building Typification. *International Journal of Geographical Information Science*, 21(3): 283-298.

BURGHARDT, D. and NEUN, M., 2006, Automated sequencing of generalisation services based on collaborative filtering. In *Geographic Information Science*, M. Raubal, H. Miller, A. Frank, M. Goodchild (Eds) (Springer, LNCS 4197) pp. 41-46.

BURGHARDT, D., M. NEUN and R. WEIBEL, 2005, Generalization Services on the Web – A Classification and an Initial Prototype Implementation, *Cartography and Geographic Information Science*, 32(4), 257-268.

BURGHARDT, D. and STEINIGER, S., 2005, Usage of Principal Component Analysis in the Process of Automated Generalisation. In *Proceedings of the XXII International Cartographic Conference*, A Coruña, Spain, CD-ROM.

BUTTENFIELD, B. P., MCMASTER, R. B., (1991, eds), *Map Generalization: Making Rules for Knowledge Representation*. London, Longman.

CHANNABASAVAIAH, K., HOLLEY, K. and TUGGLE, E., Migrating to a service-oriented architecture. *IBM DeveloperWorks*, http://www-128.ibm.com/developerworks/library/ws-migratesoa/ (Accessed 07/2006).

CHITHAMBARAM, R., BEARD, K. and BARRERA, R., 1991, Skeletonizing Polygons for Map Generalization. *Technical Papers ACSM*, Baltimore, Vol. 2, pp. 44 - 55.

CHRISTOPHE, S. and RUAS, A., 2002, Detecting Building structures for generalisation purposes. In *Advances in Spatial Data Handling*, D. Richardson and P. van Oosterom (Eds) (Berlin, Springer), pp. 419–432.

DE FLORIANI, L. and PUPPO, E., 1995, Hierarchical triangulation for multiresolution surface description. *ACM Transactions on Graphics*, 14, 363–411.

DUDA, R., HART, P., and STORK, D., 2000, *Pattern Classification*. 2nd edition, John Wiley, New York.

EDWARDES, A., BURGHARDT, D., BOBZIEN, M., HARRIE, L., LEHTO, L., REICHENBACHER, T., SESTER, M. and WEIBEL, R., 2003, Map Generalisation Technology: Addressing the need for a common research platform. In *Proceedings of the Proceedings of the 21st International Cartographic Conference*, Durban, South Africa.

EDWARDES, A., BURGHARDT, D. and NEUN, M., 2007, Experiments to build an open generalisation system. In *Generalisation of geographic Information: Cartographic Modelling and Applications*, W. Mackaness, A. Ruas and T. Sarjakoski (Eds) (Elsevier Science).

ESRI, 1998, ESRI Shapefile Technical Description, http://www.esri.com/library/whitepapers/pdfs/shapefile.pdf.

FITZKE, J., GREVE, K., MÜLLER, M. and POTH, A., 2004, Building SDIs with Free Software the deegree project. In *Proceedings of the 7th Conf. Global Spatial Data Infrastructure*, Bangalore, India.

FUCHS, M., 2002, Methoden zur objektiven Ableitung von Bodenkarten im Folgemastab. PhD thesis, Free University of Berlin.

HAMPE, M., ANDERS, K.H. and SESTER, M., 2003, MRDB Applications for Data Revision and Real-Time Generalisation. In *Proceedings of the Proceedings of 21st International Cartographic Conference*, August, Durban/South Africa.

HARRIE, L. and WEIBEL, R., 2007, Modelling the Overall Process of Generalisation. In *Generalisation of Geographic Informations: Methods and Applications*, W. Mackaness, A. Ruas and T. Sarjakoski (Eds) (Elsevier Science).

HAUNERT, J.-H. and SESTER, M., 2004, Using the Straight Skeleton for Generalisation in a Multiple Representation Environment. *ICA Workshop on Generalisation and Multiple Representation,* Leicester (UK), http://ica.ign.fr/Leicester/paper/Haunert-v2-ICAWorkshop.pdf, (accessed 01/2007).

HOLT, R., WINTER, A. and SCHÜRR, A., 2000, GXL: Toward a Standard Exchange Format, http://www.gupro.de/GXL/ (accessed 07/2006).

HORTON, H., 1945, Erosional Development of Streams and their Drainage Basins. *Bulletin of the Geological Society of America*, 56, 275–370.

ICA, 2004, Brain storming Sessions at the ICA Workshop on Generalisation and Multiple Representation, available from http://ica.ign.fr/.

ILLERT, A. and AFFLERBACH, S., 2004, Global schema specification, GiMoDig-project, Deliverable D5.3.1, Public EC report, http://gimodig.fgi.fi/deliverables.php (accessed 12/2006).

JONES, C.B., BUNDY, G.L. and WARE, J.M., 1995, Map Generalization with a Triangulated Data Structure. *Cartography and Geographic Information Systems*, 22, 317–331.

JUMP, 2007, The JUMP Unified Mapping Platform, http://www.jump-project.org (accessed 03/2007)

LEHTO, L. and SARJAKOSKI, T., 2004, An Open Service Architecture For Mobile Cartographic Applications. In *Proceedings of the Location Based Services & TeleCartography*, G. Gartner (Ed.).

LÜSCHER, P., 2005, Matching von Strassendaten stark unterschiedlicher Maßstäbe und Aufbau einer MRDB. In *Sitzung Arbeitsgruppe Automation in Kartographie, Photogrammetrie und GIS*, Wien.

MACKANESS, W.A. and BEARD, M.K., 1993, Use of Graph Theory to Support Map Generalization. *Cartography and Geographic Information System*, Vol. 20, No. 4, pp. 210–221.

MCMASTER, R. and SHEA, S., 1992, *Generalization in Digital Cartography* (Washington, USA: Association of American Geographers).

MUSTIÈRE, S. and MOULIN, B., 2002, What is Spatial Context in Cartographic Generalisation?, *Conference on Geospatial Theory, Processing and Applications, IAPRS & SIS*, 34(4) 274-278.

NEUN, M. and BURGHARDT, D., 2005, Web Services for an Open Generalisation Research Platform. In *Proc. 8th ICA Workshop on Generalisation and Multiple Representation*, A Coruña, http://ica.ign.fr/.

NEUN, M., WEIBEL, R. and BURGHARDT, D., 2004, Data Enrichment for Adaptive Generalisation. In *Proc. 7th ICA Workshop on Generalisation and Multiple Representation*, Leicester, http://ica.ign.fr/.

OGC. 1999, OpenGIS® Simple Features Implementation Specification for SQL, http://www.opengeospatial.org/specs/ (accessed 07/2006).

OGC, 2002, The OpenGIS® Abstract Specification, Topic 12: OpenGIS Service Architecture, Version 4.3, OGC 02-112. http://www.opengeospatial.org/specs/ (accessed 07/2006).

OGC, 2004, The OpenGIS® Geography Markup Language (GML) Encoding Specification, Version 3.1.1, OGC 03-105r1. http://www.opengeospatial.org/specs/ (accessed 07/2006).

OGC, 2005, The OpenGIS® Discussion Paper: Web Processing Service (WPS), Version 0.4, OGC 05-007r4. http://www.opengeospatial.org/specs/ (accessed 07/2006).

PETZOLD, I., BURGHARDT, D., and BOBZIEN, M., 2005, Automated derivation of town plans from large scale data on an example of area to line simplification. In *Proceedings of the XXII International Cartographic Conference*, A Coruña, Spain, CD-ROM.

PFALTZ, J.L., 1976, Surface Networks. *Geographical Analysis*, 8, 77–93.

PLAZANET, C., AFFHOLDER, J.G. and FRITSCH, E., 1995, The Importance of Geometric Modeling in Linear Feature Generalization. *Cartography and Geographic Information Systems*, 22, 291–305.

RANA, S., 2004, *Topological Data Structures for Surfaces: An Introduction to Geographical Information Science*. John Wiley.

REGNAULD, N., 2005, Spatial Structures to Support Automatic Generalisation. In Proceedings of the *Proceedings XXII International Cartographic Conference*, A Coruña, Spain, CD-ROM.

REGNAULD, N., 2006, Improving Efficiency for Developing Automatic Generalisation Solutions. In Proc. *Joint ISPRS Workshop on Multiple Representation and Interoperability of Spatial Data*, Hannover.

RUAS, A., and C. PLAZANET. 1996. Strategies for automated generalization. In: *Proceedings 7th International Symposium on Spatial Data Handling (Advances in GIS Research II)*, Taylor & Francis, London, pp. 6.1–6.17.

RUAS, A., 1998, A method for buiding displacement in automated map generalisation. *International Journal of Geographical Information Science*, 12, 789–803.

RUAS, A., 2000, The Roles of Meso Objects for Generalisation. In *Proceedings 9th Symposium on Spatial Data Handling*, Beijing, China, pp. 3b50–3b63.

SARJAKOSKI, T., SESTER, M., SARJAKOSKI, L., HARRIE, L., HAMPE, M., LEHTO, L. and KOIVULA, T., 2005, Web generalisation services in GiMoDig - towards a standardised service for real-time generalisation. In *Proceedings of the 8th AGILE Conference on GIScience*, F. Toppen and M. Painho (Eds), Estoril, Portugal.

SHEWCHUK, J., 1996, Triangle: Engineering a 2-D quality mesh generator and Delaunay triangulators. In *Proceedings first workshop on Applied Computational Geometry*, PA, USA.

STRAHLER, A.N., 1952, Hypsometric (Area-Altitude) Analysis of Erosional Topology. *Bulletin of the Geological Society of America*, 63, 1117–1142.

THOMSON, R. and RICHARDSON, D., 1995, A graph theory approach to road network generalization. In *Proceedings of the 17th ICA Meeting*, Barcelona, Spain.

TIMPF, S., 1998, Hierarchical Structures in Map Series. PhD thesis, Technical University of Vienna.

VAN OOSTEROM, P., 1993, *Reactive Data Structures for Geographic Information Systems*, Oxford University Press.

VAN OOSTEROM, P. and SCHENKELAARS, V., 1995, The development of a multi-scale GIS. *International Journal of Geographical Information Systems* **9**(5): 489-508.

VAN SMAALEN, J., 2003, Automated Aggregation of Geographic Objects: A New Approach to the Conceptual Generalisation of Geographic Databases. PhD thesis, Wageningen University and Research Centre.

WARE, J., JONES, C. and THOMAS, N., 2003, Automated Map Generalization with Multiple Operators: A Simulated Annealing Approach. *International Journal of Geographical Information Science*, 17, 743–769.

WORBOYS, M.F. and DUCKHAM, M., 2004, *GIS. A Computing Perspective*, 2nd edition, CRC Press.