Generalisation on the web: Towards 'scale-aware' web mapping clients

Julien Gaffuri
JRC – IES – SDI unit
Via Enrico Fermi, 21027 Ispra, Italy
Julien.Gaffuri@gmail.com

Introduction

Generalisation is the simplification performed on spatial data when their representation scale decreases. Generalisation automation has been the topic of researches for years and nowadays, operational automated generalisation techniques exist. Nevertheless, automatic generalisation techniques are still not used in a huge majority of web mapping infrastructures. A future challenge of geographical information science is to make existing spatial data processing methods usable in the emerging spatial data infrastructures (Neber, 2008). This paper presents an on-going work that aims at introducing generalisation techniques in the web – it focuses on the presentation of a new web mapping client with generalisation capabilities.

1. Generalisation on the web

Figure 1 shows four examples of Internet maps with obvious readability problems. The information represented on these maps is not adapted to their scales: They represent too many objects, some of them overlap, some other are too small, etc. Generalisation addresses these readability issues by simplifying the data to make the important information readable (or improve its readability).

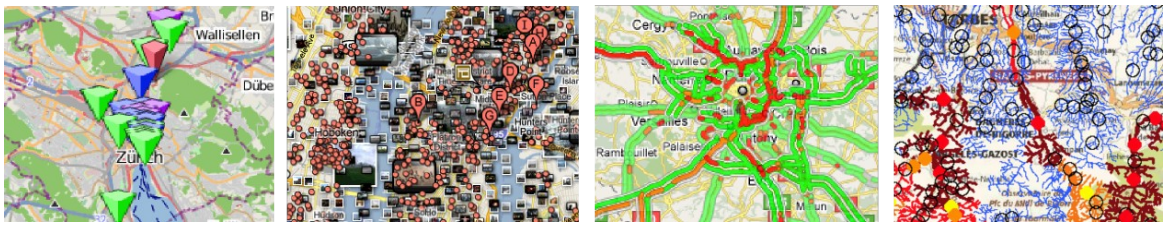

Figure 1: Examples of Internet maps

Generalisation can be divided into model generalisation and graphic generalisation. Model generalisation is the transformation from detailed to generalised concepts. When the representation scale decreases, some objects are aggregated into other objects corresponding to a more generalised concept. Figure 2 shows the example of urban concepts at different scales. Graphic generalisation is the transformation of map symbols to make them readable: too small symbols are deleted or enlarged, overlapping or too closed objects are displaced, etc. Figure 3 gives two examples of such operations. For both kinds of generalisation, automatic methods exists (Mackaness et al, 2007).

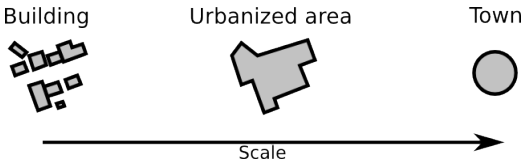

Figure 2: Model generalisation: buildings, urbanized areas, and towns

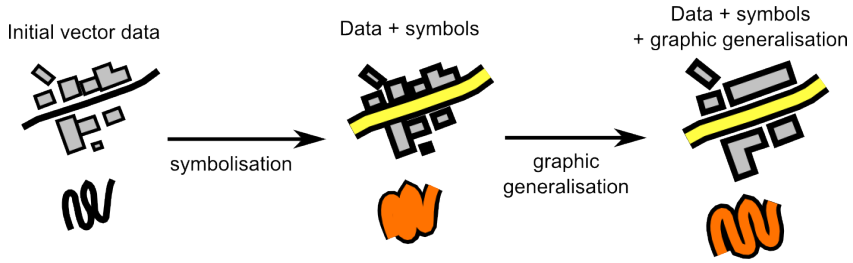

Figure 3: Two examples of graphic generalisation

Introducing generalisation on the web is not an easy task: Existing generalisation and web-mapping systems have different architectures, and their integration is hard. (Gaffuri 2011) discusses this issue and proposes an architecture to plug generalisation on the web. This architecture (Figure 4) is composed of four types of servers: Raster web mapping servers diffuse raster map data produced using model and graphic generalisation techniques. Vector web mapping servers diffuse vector data from a multi-scale database built using model generalisation techniques. Legend web mapping servers provide cartographic styles for vector data. Graphic generalisation libraries servers diffuse graphic generalisation libraries to be loaded dynamically by the clients depending on their generalisation needs. Finally, this architecture is based on the use of a new kind of client this paper focuses on.

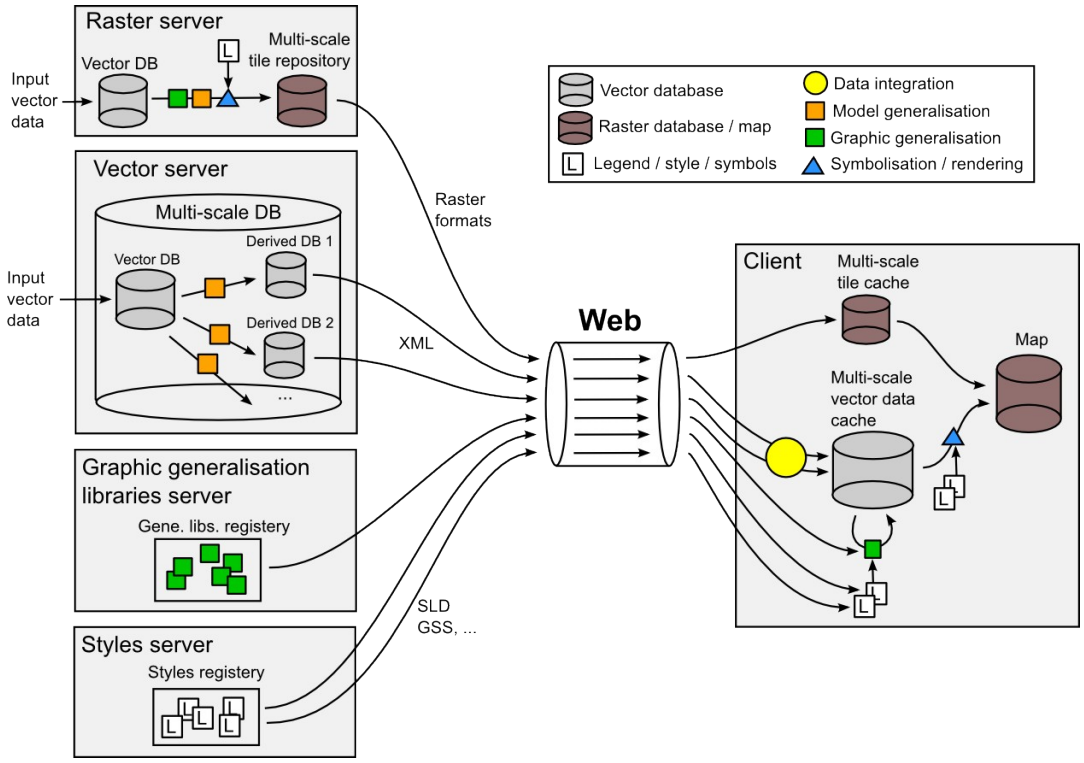

Figure 4: An architecture to enable Internet maps generalisation (from Gaffuri 2011)

2. A 'scale-aware' client for web mapping

On the Internet, maps are not displayed like other images. A map is displayed within a specific client that makes some interactive operations possible (basically: zoom, pan, layer

addition/removal). This interactivity makes stronger the need for methods to really adapt the data to the user's expectations. Especially, when a user zooms in and out, the displayed data should be adapted to the new visualization scale to solve the readability problems, as the ones shown on figure 1. A 'scale-aware' client would be a client that displays spatial data according to the requested visualization scale. Such client is able to transform the data to adapt it to the visualization scale. We present how such a client may work.

Principles

The principle of the scale-aware client presented on figure 4 are the following:

- Raster background data: Raster data are static and cannot be transformed. For a better interactivity, raster data use is limited to the map background.
- Scale-aware queries: The spatial data queries include not only the extend of the data to display, but also the scale. Vector servers are able to handle such queries and deliver multi-scale data prepared using model generalisation techniques. One advantage is that such generalised data are faster to be transferred through the network than non-generalised data.
- Vector data integration: Existing web mapping clients overlay data from different sources without real integration: The relations between these data layers are not analysed. Our client makes such integration possible.
- Style choice: Some styles used to display the vector data can be loaded from style servers. They can also be specified by the user (this is not possible on raster data).
- On-the-fly graphic generalisation: The client stores the map data in a local multi-scale cache. The map is improved using graphic generalisation processes computed on-the-fly by the client, according to the specified style. This approach where model and graphic generalisation are shared between respectively the server and the client has been used by (Harrie et al 2002) and (Sester & Brenner 2005). Depending on the need, some software components needed to compute generalisation are loaded dynamically from a generalisation libraries servers.
- Progressive display: The client displays the data progressively while they are loaded, and also while they are generalised. The more time a given location at a given scale is displayed, the more the readability is improved (this requires the use of 'anytime' generalisation algorithms). Because the objects generalised versions are stored, this generalisation is not repeated when the user zoom-in and out. When the user changes the map styles, the map objects also dynamically and progressively adapt to the new style.

Proposition for an architecture

Figure 5 proposes an architecture of a scale-aware client that respects the previous principles (in this example, the client displays a map composed of three layers).

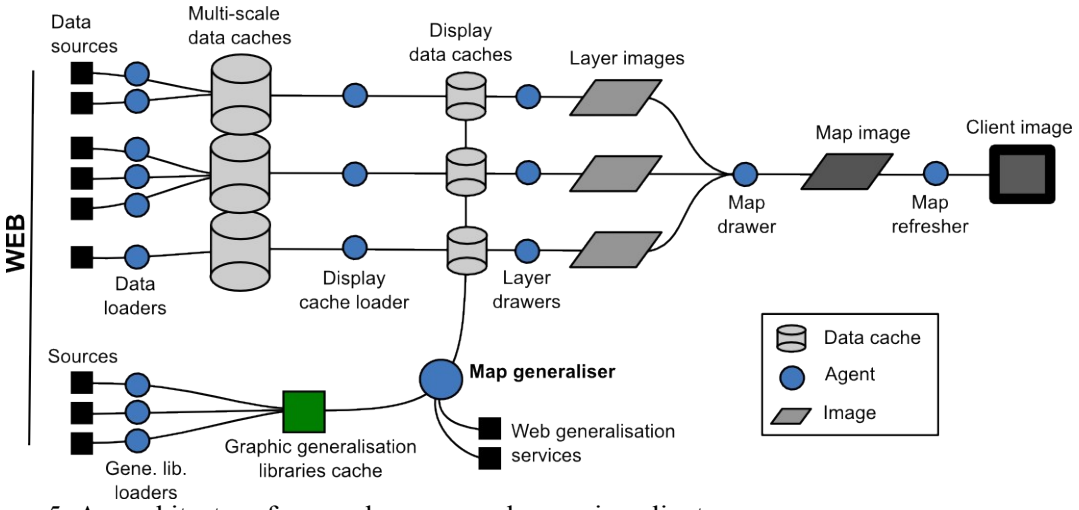

Figure 5: An architecture for a scale-aware web mapping client

This architecture lies on the following data structures:

- The data sources: A data source is a reference to spatial datasets and styles to be loaded. These data sources are local or remote and may use standard web services like WFS and WMS. Each layer can have several data sources.
- The multi-scale data caches: The data of each layer are loaded and stored locally within this data cache. This cache is a local multi-scale database: It stores the objects as "multi-scale features". A multi-scale feature is composed of a set of different "scaled-features". A scaled-feature is a feature (geometry + attributes) with a scale extends and a style. The scale extends is the scale range for which the scaled-feature is a priori suitable. By this way, each multi-scale feature can have different representations along the scale line. In this approach, the scale is considered as a dimension, like time, as proposed by (van Oosterom and Stoter 2010).
- The display data caches: Each layer has a data display cache; this cache is the set of the scaled-features to display in the layer according to the map extends and the visualisation scale. These objects are not copied, but indexed. This cache is used to make redraws of the layers faster.
- The layer images: The images of each layer. It is the result of the display of the cache objects with their styles.
- The map image: This image is the result of the layer images overlay.
- The graphic generalisation libraries cache: This data structure contains the graphic generalisation libraries that have potentially been loaded from remote servers to be used during the generalisation process.

The management of these data structures to display the map on the client is done through different processes. In order to ensure a progressive display of the map, these processes are parallel and act together on the data structures previously presented. The agents responsible of these processes are:

- The data source loader: Its role is to load the data from a data source into the layer data cache. There are two kinds of data source loaders: Some are launched only one time and load all the data of the data source (It is the case, for example, when the data source refers to a file). Some other are launched every time the map location changes and load only the data that are within the map spatial extend (It is the case, for example, when the data source is a WFS).

- The display cache loader: Its role is to index the scaled-features of the multi-scale data cache to be displayed on the layer image. This selection is performed depending on the scale and the extends of the map display (a spacial spatial index can be used).
- The layer drawer: Its role is to draw the scaled-features of the display cache on the layer image.
- The map drawer: Its role is to draw the map image with a simple overlay of all layer's images.
- The map refresher: Its role is to draw the client image. This image is an overlay of the map image and other additional items (the buttons, the scale bar, etc.).
- The map generaliser: Its role is to transform the scaled-features indexed by the display data caches to adapt them to the visualization scale. The generalisation process uses the generalisation libraries loaded into the cache. Such generalisation libraries can also be made available using web processing services. The improved features are stored in the data caches: Generalisation operations are performed only one time, and they are performed only on the displayed objects, for the requested visualisation scale.
- The generalisation libraries loader: Its role is to load a generalisation library that is required by the map generaliser for the generalisation process.

These processes are parallel and launched depending to the user's actions on the interface. This architecture ensure the display is progressive depending on the network speed, the generalisation process and the user actions. A first prototype that implements some of the presented components has been developed as part of the opencarto library (https://sourceforge.net/projects/opencarto/). A demo is available on-line on the project's website.

Conclusion

We have presented the architecture of a web mapping client that is able to adapt on-the-fly the data it displays, according to the scale. Because spatial data available on the web are different to the traditional topographic data for which most of the generalisation techniques have been developed for, there is a need for generalisation techniques especially dedicated to new data and usages of Internet maps. To go further, the presented client could be improved by adding other automatic mapping method such as label placement or color choice improvement methods as the one proposed by (Christophe 2011).

References

Christophe, S. (2011). Creative colours specification based on knowledge (COLourLEGend System), In *The Cartographic Journal* (to be published).

Gaffuri, J. (2011), Improving web mapping with generalisation, In *Cartographica* (to be published).

Harrie, L., T. Sarjakoski, and L. Lehto (2002). A variable-scale map for small-display cartography. In *Joint International Symposium on GeoSpatial Theory, Processing and Applications (ISPRS/Commission IV, SDH2002)*, Ottawa, Canada.

Mackaness, W. A., A. Ruas, and T. Sarjakoski (Eds.) (2007). *Generalisation of Geographic Information: Cartographic Modelling and Applications*. Elsevier.

Neber, D. D. (2008). *Developing Spatial Data Infrastructures: the SDI cookbook*. Global Spatial Data Infrastructure.

Sester, M. and C. Brenner (2005). Continuous Generalization for Visualization on Small Mobile Devices. In P. Fisher (Ed.), *Developments in Spatial Data Handling*, pp. 355-368.

van Oosterom, P. and J. Stoter (2010). 5D modelling of geo-information: full integration of space, time and scale dimensions. In GIScience 2010, Sixth international conference on Geographic Information Science, Zurich, 14-17th September, 2010