Toward Self-Generalizing Objects and On-the-Fly Map Generalization

Mamane Nouri Sabo, Yvan Bédard, Bernard Moulin, and Eveline Bernier
Centre for Research in Geomatics / Université Laval / Québec / QC / Canada

Abstract

Map generalization is a complex task that sometimes requires human intervention. In order to support such a process on the fly, we propose a generalization approach based on self-generalizing objects (SGOs) that encapsulate geometric patterns (forms common to several cartographic features), generalization algorithms, and spatial integrity constraints. During a database enrichment process, an SGO is created and associated with a cartographic feature or a group of features. Each SGO created is then transformed into a software agent (SGO agent) in a multi-agent on-the-fly map-generalization system. SGO agents are equipped with behaviours that enable them to coordinate the generalization process. This article presents the concept of the SGO and two prototypes developed to support this approach: a prototype for the creation of SGOs and another for the on-the-fly map generalization (which uses the created SGOs).

Keywords: self-generalizing object (SGO), on-the-fly map generalization, geometric pattern, database enrichment

Résumé

La généralisation cartographique est un processus complexe qui demande parfois l'intervention humaine. Afin de supporter un tel processus à la volée, nous proposons une approche de généralisation qui se base sur les SGO (objets auto-généralisants ou *self-generalizing objects*) qui encapsulent à la fois des patrons géométriques (qui sont des formes communes à plusieurs objets cartographiques), des algorithmes de généralisation et des contraintes d'intégrité spatiales. Lors d'un processus d'enrichissement de la base de données, un SGO est créé et associé à chaque objet ou groupe d'objets de la carte. Les SGO créés sont ensuite automatiquement transformés en agents logiciels (agents SGO) dans un système multi-agent de généralisation à la volée. Les agents SGO sont dotés de comportements qui leurs permettent de coordonner le processus de généralisation. Dans cet article, nous présenterons le concept des SGO et les prototypes (un prototype pour la création et l'enrichissement des SGO et un autre pour la généralisation à la volée, utilisant les SGO créés) développés pour supporter cette approche.

Mots clés : objets auto-généralisants (SGO), la généralisation cartographique à la volée, patron géométrique, enrichissement des bases de données

1. Introduction

In the past, map production was the cartographer's responsibility, because of the specific skills required to create maps. Users received maps created in advance and generally produced in a series. However, the technological developments of the last decade have generated a new type of medium for spatial data dissemination, the World Wide Web. In addition, there are new applications such as Web mapping and spatial database querying. With the emergence of these new applications, cartographic data has become more accessible to the general public and can be better adapted to users' needs.

This democratization of cartographic data has resulted in new user requirements for visualization tools and geographic data. Beyond a simple static visualization, users expect flexible querying of data and more powerful tools for personalizing maps. Most often, they want to define parameters such as map scale, symbology, and map content and to highlight individual cartographic features with a different colour, a more detailed shape, or a new symbol in order to get maps better adapted to their needs. The management of symbols and map content is fairly well supported by current GISs. However, these tools have a limited ability to manage levels of abstraction. For example, in ArcGIS and similar systems, reference scale controls and visible scale constraints are used to control the layers to be shown and the width of symbology according to the map scale displayed. Such mechanisms do not allow the user to adapt the map content for each individual feature (which would be overwhelming) or to adapt the level of detail of a group of map features (requiring a large number of fine-grained combinations of reference scale controls and visible scale constraints). In addition, such mechanisms do not eliminate possible conflicts (e.g., superposition of objects) on the fly for the displayed map scale. Since each scale and layer combination requires a different solution, these issues cannot be resolved in advance using existing specialized batch-oriented generalization modules. Similarly, Web mapping sites such as Google Maps lack real map-generalization capabilities, since they rely on the selective display of one map among several pre-defined maps at different scales. Zooming within the selected map provides no generalization capability; zooming to a scale outside of the range of the selected map simply produces another pre-built map. Furthermore, these sites are typically very limited with respect to map customization, since users cannot select or highlight individual objects or perform other key generalization tasks (e.g., object displacement, object elimination, changing symbols, enlarging symbols without enlarging the scale). Considering today's state of the art, however, GIS and Web mapping solutions represent a useful and necessary evolution, especially for online map users.

According to Allen Newell (1990), a response time of 0.1 to 10 seconds is needed to perform cognitive tasks and maintain one's train of thought. Similarly, a recent study sets 4 seconds as the new threshold of acceptability for online shopping (Young and Smith 2006). Not all users require a quasi-instantaneous response time, but certain applications do (e.g., spatial online analytical processing, or SOLAP). Who would object to the idea of seeing their maps generated on the fly and efficiently, with results better tailored to the displayed scale or better customized to their exact needs? Self-generalizing objects (SGOs) represent a step toward this ultimate objective. The ideal solution for helping users to create maps with specific content and at a specific scale would be to have a single large-scale database and to derive maps at smaller scale according to users' needs using on-the-fly automatic cartographic generalization processes. However, in order to derive smaller-scale maps according to users' needs, either basic and pre-defined multiple representations are used to mimic on-the-fly cartographic generalization or, more usually, level-based batch processes are deployed. In conventional cartography, map generalization is responsible for reducing the complexity of a map through a process of scale reduction. This process emphasizes the essential while suppressing the unimportant; it maintains logical and unambiguous relations between map objects and preserves the aesthetic quality of the original map (Weibel and Dutton 1999).

Over the past decades, several projects have dealt with cartographic generalization (e.g., Douglas and Peucker 1973; Mackaness 1994; Lamy and others 1999; Sarjakoski and Kilpelainen 1999; Sester 2000; Weibel and Dutton 1998). At first, research on the automation of map generalization focused on the development of algorithms, with particular emphasis on algorithms for line simplification. During this early period, generalization operations were typically applied to all occurrences of an object class, irrespective of the object's characteristics or context. The expected power of the algorithms used was to include the necessary parameters for all possible geometries that might be encountered in any given context. Except in fully controlled and well-defined situations, however, this is an impossible objective. Map generalization is a complex process, requiring the coordinated use of several operations. In order to determine the required algorithms, along with their sequences and corresponding parameters, knowledge-based approaches are used (Allouche and Moulin 2005). This category includes, among others, rule-based approaches (Nickerson and Freeman 1986; Armstrong 1991; McMaster 1991), constraint-based approaches (Beard 1991; Weibel and Dutton 1998; Ruas 1999), and multi-agent approaches (Baeijs, Demazeau, and Alvares 1995; Lamy and others 1999; Li, Zhou, and Jones 2002; Jabeur 2006, Jabeur, Boulekrouche, and Moulin 2006). A knowledge-based approach requires that our knowledge of the generalization process be formalized into a chain of reasoning paths, each leading to a particular decision or procedure (Müller 1991). Unfortunately, cartographic knowledge is very difficult to formalize, given that map generalization is a subjective and holistic process.

Despite significant progress over the past few years, automatic cartographic generalization remains a difficult task and continues to require human intervention (Sabo, Cardenas and others 2005). For applications, such as Web mapping, that require extremely short response times, today's alternative is to use a multiple representations database (MRDB) to simulate on-the-fly

map generalization. In its simplest but most common form, an MRDB is a spatial database used to store various representations of the same territory (Devogele, Trevisan, and Raynal 1996; Weibel and Dutton 1999) at different levels of abstraction (e.g., various scales). In its most complete form, rarely implemented, the MRDB acts as a storage unit for various geometry, semantics, and graphics of the same object (Bédard, Bernier, and Devillers 2002) from which, based on the user's request, the most suitable representations are selected for the desired scale.

There are several types of map-based Web sites that allow users to zoom in and out of a particular region, usually based on simple enlargement/reduction of the presented map. When the enlargement/reduction ratio reaches a high number (e.g., 5×), the represented map is replaced by another map that may differ significantly in the degree of generalization. In such an approach, map scales are pre-defined and do not allow users to get the corresponding data at any desired level of abstraction.

In order to allow users to generate maps on the fly and at arbitrary scales, we propose an approach based on geometric patterns (shapes representative of several cartographic features), combined with sets of generalization algorithms (applicable to patterns and to cartographic features) and spatial integrity constraints, all encapsulated in a single structure called a self-generalizing object (SGO). This approach allows us to include additional human expertise in an efficient way at the level of individual cartographic features, leading to database enrichment that better supports automatic generalization.

We begin this article with an overview of Web mapping and on-the-fly generalization concepts. We then present the SGO concept and its underlying components: geometric patterns, generalization algorithms, spatial integrity constraints, and their combined usage. Next we introduce two working prototypes: the *SGO Creation Engine* and the *On-the-Fly Map Generalization System*. The latter is based on a multi-agent approach and allows us to generate maps at arbitrary scales. We conclude by presenting perspectives and remaining challenges.

2. On-Demand Mapping and On-the-Fly Map Generalization

2.1 ON-DEMAND MAPPING

An on-demand map is a map generated according to a specific user's requirements, in contrast to maps generated in mass quantities for the generic needs of large groups (e.g., topographic maps). The user's requirements may affect the map's scale, its content, the objects to be highlighted or filtered, the symbols to be used, the time necessary for delivery, and so on. The map may be created manually or automatically, on paper or using an electronic support, with or without generalization, and with or without delay. Therefore, the main goal of on-demand cartography is to provide users with data at an adequate level of abstraction, whose content and graphic representation are suited to their needs, while taking into account the specifications of each user's visualization tools and hardware constraints (e.g., screen size, network bandwidth).

Traditional cartography is not unfamiliar with on-demand maps. Indeed, several types of maps with low print runs exist in traditional cartography; those used in extremely specialized fields are typically generated on demand. The main difference between traditional on-demand mapping and today's is that the former tries to satisfy a small group of users from a specific field, whereas on-demand mapping in the Internet era tries to satisfy not only the needs of a selected group of people but also those of individuals accessing the Web. Consequently, many more constraints must now be taken into account.

The digital solutions that are currently available typically use a single and homogeneous level of abstraction for their maps. Changing the level of abstraction generally means exchanging the map for another one created at a different scale. However, as manual cartography has already shown, the level of abstraction need not be uniform within a map. For example, on a street map, the elements of the road network are often more detailed than the elements of other features, such as those of the hydrographic network, which are presented at a higher level of abstraction. Indeed, with the exception of topographic maps, which try to achieve a certain balance in terms of levels of abstraction for all categories of map features, maps usually present heterogeneous levels of abstraction.

The number of Internet users has recently exceeded the 1 billion mark (MMG 2006), representing more than 16% of the world's population. The increasing number of Internet users accessing cartographic products drastically increases the number of requirements in terms of spatial data. These requirements are followed by technological constraints. Thus, even though the Internet has encouraged the development and democratization of geographic data, it has also brought with it new production challenges relating to both technological issues (e.g., data transfer rates, map supports) and user issues (e.g., heterogeneous users, users who lack basic cartographic knowledge). It is impossible to know in advance everything about the user's context or the kinds of customization he or she will require. For example, in order to adapt the level of abstraction, cartographic generalization is necessary. With an interactive and dynamic medium such as the Internet, ''map on demand'' has become synonymous with ''map generated in real time,'' creating an expectation that all map-creation processes, as well as generalization processes, must be performed instantaneously.

Therefore, the map-generalization process must take place on the fly in such cases.

2.2 ON-THE-FLY MAP GENERALIZATION

2.2.1 What is on-the-fly map generalization?

On-the-fly map generalization can be defined as the ''creation, in real time and according to the user's request, of a cartographic product suited to its scale and to its purpose, from a larger-scale database'' (Weibel and others 2002, 320). To provide on-the-fly cartographic generation, the process must rely on fast, effective, and powerful methods. Unfortunately, cartographic generalization is known to be a complex and time-consuming process. Furthermore, in order to produce maps suited to a user's requests, on-the-fly map generalization must also be flexible enough to take numerous factors into account.

One of the main problems associated with on-the-fly map generalization is determining whether or not the generated map is suited to the user's needs. In conventional cartography, validation of results, in accordance with the producer's specifications, is done *a posteriori* by the map producer. In the case of on-the-fly map generalization, however, data producers have limited control over the content of the resulting maps and, consequently, over the quality of the generalized data. What is more, most users of Web mapping applications have little or very limited knowledge about the map-making process and, therefore, cannot assess the quality of the resulting product. As a result, on-demand map-production tools in general, and on-the-fly map-generalization processes in particular, must generate both *predictable* and *good-quality* results for users.

All these requirements (speed, complete automation, flexibility, predictability, and quality) are major challenges, considering that the problem of conventional cartographic generalization is not yet entirely solved.

2.2.2 State of the Art of On-the-Fly Map Generalization

Currently, there are three main research orientations in the field of on-the-fly map generalization (Weibel and others 2002).

2.2.2.1 Approach based on generalization algorithms

This approach is based exclusively on specific generalization algorithms (e.g., Douglas and Peucker 1973). As previously mentioned, however, generalization is a complex and time-consuming process. Even if the technological developments of recent years have allowed for the creation of high-speed processors, it is nevertheless necessary to find methods that can accelerate processes in order to support on-the-fly map generalization. This explains why some authors have proposed methods based on pre-computed attributes in order to accelerate the generalization process (e.g., Van Oosterom and Schenkelaars 1995). Unfortunately, these approaches do not take into account the local spatial environment of map objects. Furthermore, because of its complexity, generalization cannot be carried out by simply applying generalization algorithms sequentially, without taking into account the map objects' spatial neighbourhood. Finally, existing federative generalization methods based on artificial-intelligence techniques (e.g., the multi-agent approach) are very complex and require long treatments and adjustments, which currently limits their use for on-the-fly map generalization.

2.2.2.2 Multiple representations approach

As previously mentioned, the ideal solution to allow users to get data at the desired level of abstraction would be a single database at the most detailed level of cartography from which we could automatically create products at smaller scales. However, this is still impossible given the current state of automatic map generalization. Therefore, some researchers use multiple representations as an alternative solution. This approach proposes to store several pre-defined representations of a given object (usually at different scales) within the same database. The simplest representations are usually obtained via manual or semi-automatic generalizations of the most detailed representations. Most of the time, this approach produces a multi-scale database that includes several map layers at different scales. The appropriate scale is then selected according to the user's request. In recent years, much effort has been devoted to managing multiple representations (Martel 1999; Bédard and others 2002; Devogele, Badard, and Libourel 2002; Vangenot 1998; Timpf 1998; Sarjakoski 2007; Bernier and Bédard 2007). In terms of personalization, the multiple representations approach is extremely limited, because all scales need to be pre-defined. Other important problems related to this approach include the difficulty of creating the necessary map scales and of structuring such a database; data redundancy; and the difficulty of propagating data updates.

2.2.2.3 Approach combining generalization algorithms and multiple representations

The problems associated with automatic generalization and multiple representations have given rise to a third approach that combines these two approaches. In this combination, a multi-scale database is used and, depending on the user's request, the most appropriate scale is selected. When the selected scale does not match the user's requested scale, the selected data can be refined by applying simple generalization operations (e.g., selection or simplification). This is a two-step process whereby the system selects the representation that is closest to the user's desired map representation

(based on the closest map scale) and the generalization process is applied to this closest representation. Several authors (e.g., Cecconi, Weibel, and Barrault 2002; Jabeur 2006) use this approach. Its advantage is that it reduces the effort needed for generalization and improves the quality of the result: the smaller the difference between the initial map scale and the desired one, the easier the generalization process. To be truly efficient, however, this method must rely on a database that includes several scales, leading to the typical problems associated with multiple representations. To improve this third approach, it is necessary to develop new methods that minimize as much as possible the problems associated with both automatic generalization and multiple representations. It is in this context that we propose an approach combining generalization algorithms and geometric patterns, a type of multiple representations approach in which data redundancy is kept to a minimum.

3. The Self-Generalizing Object (SGO)

3.1 WHAT IS AN SGO?

In order to support a process of map generalization on the fly, we propose an approach based on a fast generalization method (geometric patterns) and database enrichment, all encapsulated in an object called the self-generalizing object (SGO). This enrichment, which exploits human expertise, allows us both to generate explicit spatial information (e.g., the alignment of buildings) that is implicit in the initial database and to introduce into the database generalization knowledge (e.g., procedural knowledge) that is difficult to formalize.

An SGO can be defined as an object based on fast generalization methods and database enrichment, and associated with a cartographic feature in order to facilitate and accelerate the automatic generalization of this feature. To make a cartographic feature generalizable, the SGO uses four fundamental components:

1. *Geometric patterns* (shapes representative of several cartographic features) – see section 3.2.1
2. *Process patterns* (recurring sets of generalization algorithms and their sequences) – see section 3.2.2
3. *Spatial integrity* constraints – see section 3.2.3
4. A *behaviour pattern* (in order to coordinate the generalization process) – see section 3.4.2

Thus, during the database enrichment phase, the operator can create (interactively and/or using an automatic routine) an SGO for every map feature and enrich it by specifying its components (geometric patterns, process patterns, spatial integrity constraints, and behaviour pattern). Once created and enriched, an SGO contains the necessary tools, data, and knowledge to generalize the objects with which it is associated. The process of generalization is initiated by the SGO in order to satisfy a violated integrity constraint. For this purpose, SGOs are equipped with behaviours that enable them to coordinate the generalization process and to interact with other SGOs. To satisfy a violated constraint, an SGO uses geometric patterns and/or process patterns as generalization methods. Although the SGO's enrichment can be considered additional work for cartographers, this work is done only once, whereas the resulting SGOs may be used thousands of times (i.e., every time map generalization occurs).

The difficulty of automating the generalization process, and particularly of formalizing generalization knowledge, has led some researchers to propose a generalization approach that allows the division of tasks between humans and computers (Weibel 1991; Weibel and Buttenfield 1988). This division of tasks is achieved by means of interactive generalization tools. By using these tools, the cartographer can select the objects or groups of objects to be generalized; determine the generalization

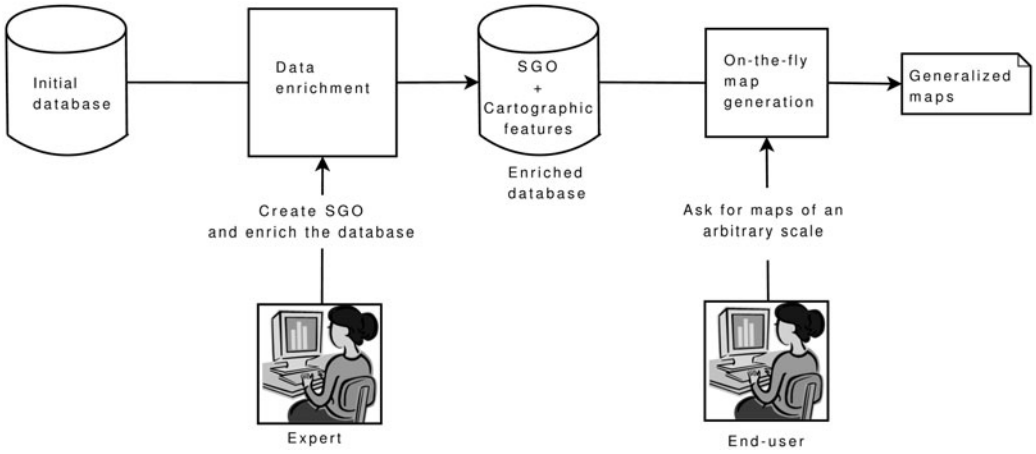

Figure 1. The SGO philosophy.

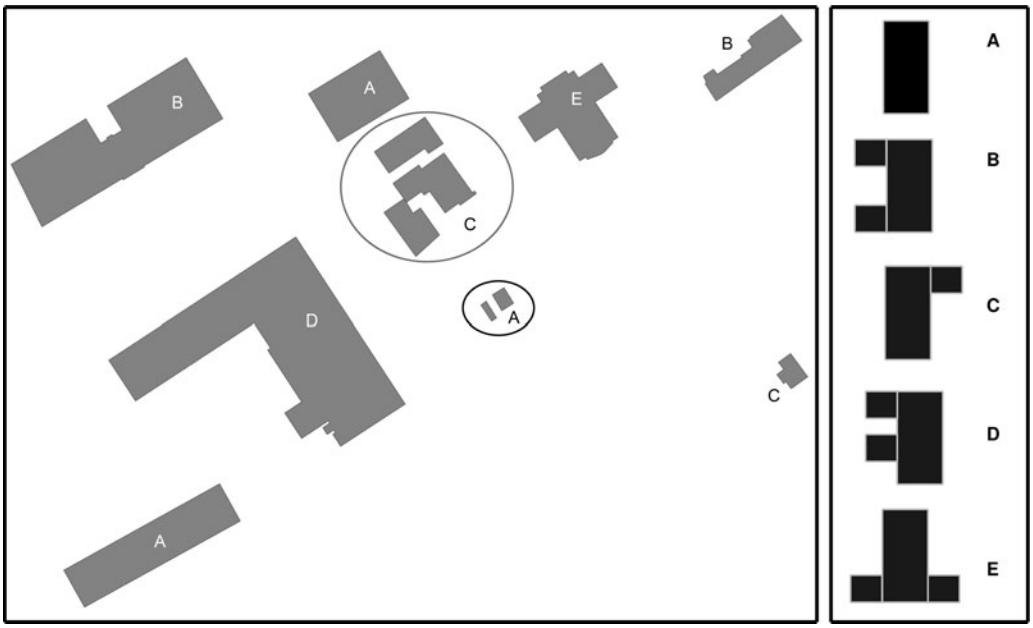

Figure 2. Detected patterns of shape on a map (left) and their corresponding geometric patterns (right).

operations, algorithms, and parameters to be applied; assess the quality of the generalized data; and, finally, detect conflicts and eliminate them. Unfortunately, even today, much of the cartographer's work is lost because, typically, only the generalized map, and not the tasks that the cartographer performed (e.g., the chosen sequence of generalization algorithms), is stored in the database. Sometimes it is even necessary to redo all these tasks in order to generate a map of the same territory at a different scale. The philosophy underlying the SGO approach is to either recover or reproduce the work necessary for map generalization and then to store it. Therefore, the SGO can provide means to recover the work completed by experts during the generalization process. Figure 1 demonstrates the philosophy underlying the SGO concept.

3.2 SGO COMPONENTS

3.2.1 Geometric Patterns

In any domain, the concept of pattern is related to the concept of recurrence. In our environment, many phenomena and objects have recurrent geometries. Consequently, when viewing the map, the eye discerns patterns of shape, orientation, connectedness, density, and distribution (Mackaness and Edwards 2002). For example, on a map, many buildings can have a similar shape. This similar shape represents a shape pattern (Figure 2, left). Thus, the geometric pattern (Figure 2, right) is a geometrical construction of a shape pattern (i.e., a geometrical construction of a recurring shape). The geometric patterns created are then stored in a geometric patterns database.

The idea is to associate a geometric pattern with each cartographic feature (when possible). When a geometric pattern is associated with a cartographic feature, parameters such as the feature's orientation, position, and size are defined and stored. Then, during the generalization process, instead of being generalized using traditional generalization algorithms, the feature is simply replaced by the associated geometric pattern (or its simplified form) according to the required level of abstraction. In order to match the geometric pattern to the geometry of the cartographic feature being replaced, the geometric

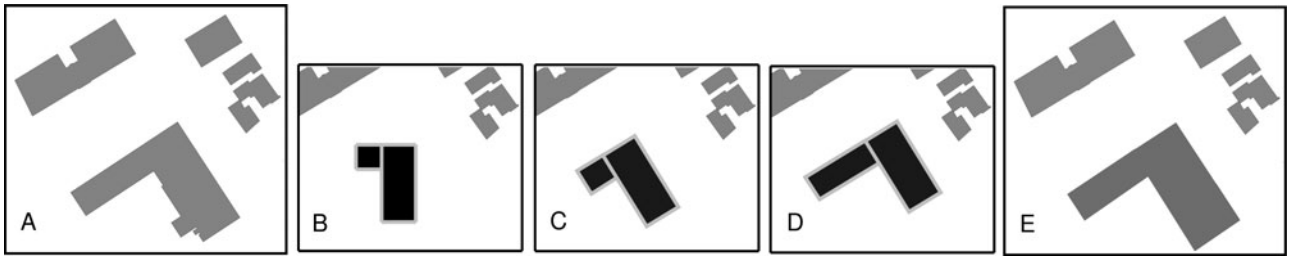

Figure 3. Use of a geometric pattern in a cartographic generalization process: (A) Initial map; (B) Initial feature replaced by a geometric pattern; (C) Orienting the pattern; (D) Scaling the pattern; (E) Result.

pattern is adjusted using simple operations (rotation, positioning, and scaling) and stored parameters (e.g., position, size, and orientation). This solution can be thought of as a combination of a look-up table (stored geometric patterns) and a small amount of computation (matching or generalizing the geometric pattern). The use of a geometric pattern during a cartographic generalization process is illustrated in Figure 3.

To make it flexible, a geometric pattern is composed of one or several primitives (basic elements that make up a geometric pattern). For example, geometric patterns of buildings can be composed of one or more connected primitives (e.g., juxtaposed rectangles). This flexibility facilitates the generalization of the geometric pattern, which, in turn, allows us to generate a simplified form of the pattern. When a geometric pattern is generalized, the operation is kept under control by the geometric pattern structure (composed of connected primitives) and the coordination of the geometric pattern's adjustment operations ensured by the SGO. A geometric pattern is generalized using simple generalization operations: elimination of a primitive, combination of primitives, and changing the size of a primitive. A Unified Modeling Language (UML) class model of our data structure for a geometric pattern is illustrated in Figure 4; Figure 5 shows some of the generalization operations for the geometric pattern; and Figure 6 depicts a geometric pattern along with its simplified forms.

The concept of the geometric pattern is not limited to buildings; it may be also applied to other classes of objects (e.g., road cloverleaves, cul-de-sacs, silos). For instance, a highly sinuous road segment, such as those found in the Alps, can be generalized using a geometric pattern

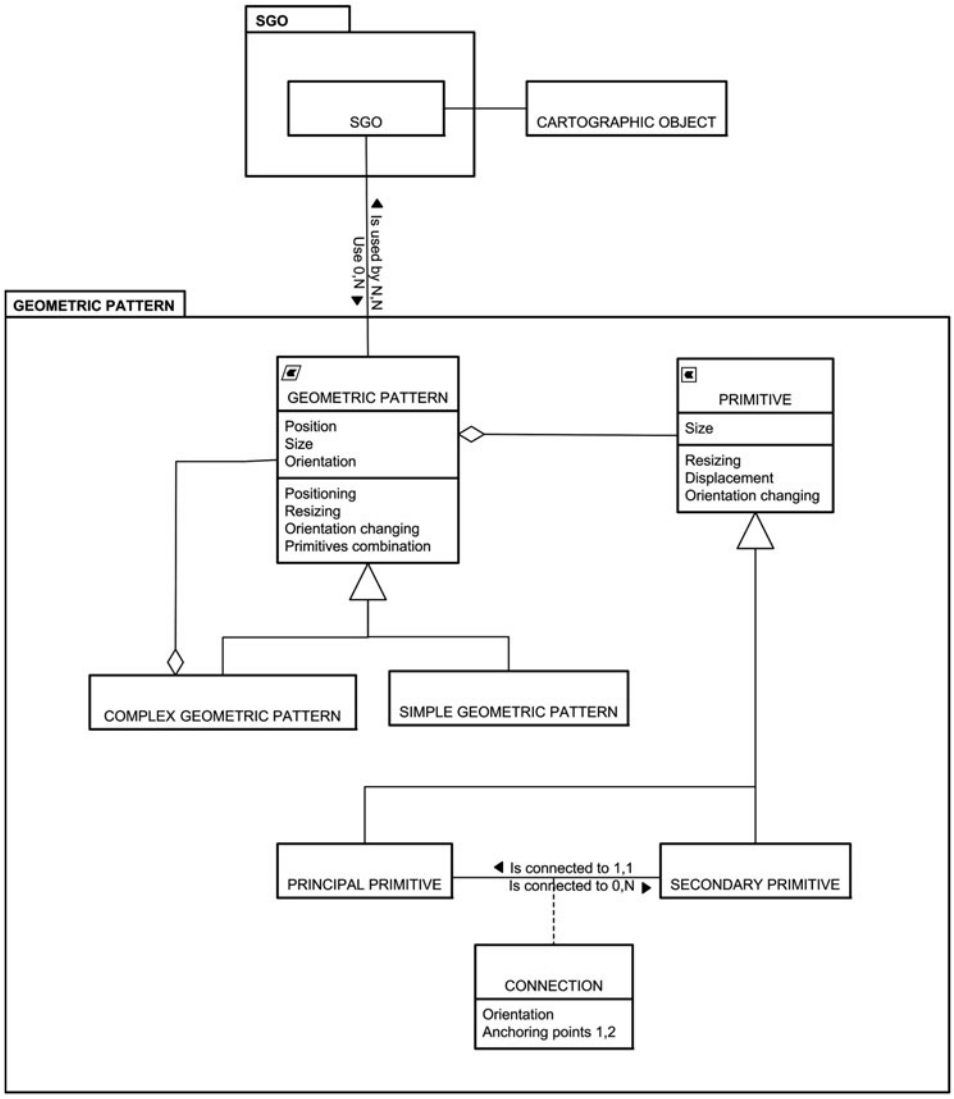

Figure 4. UML class diagram for the data structure of a geometric pattern.

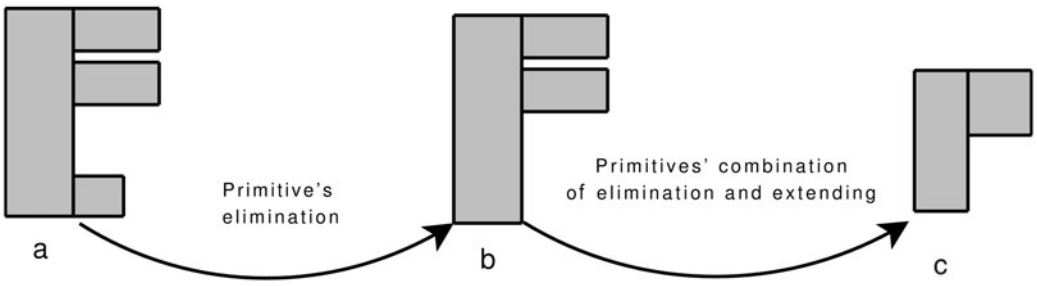

Figure 5. Generalization operations for a geometric pattern.

composed of several successive bends. This geometric pattern can be adjusted using the same operations as those used for geometric patterns of buildings.

The flexibility of geometric patterns and the simplicity of the associated operations (e.g., operations used to adjust or simplify a geometric pattern) allow us to simplify and accelerate the generalization process. Our experiments with a data set from Quebec City have shown that one geometric pattern and its simplified forms allow the replacement of 2844 individual buildings or 365 groups of buildings appearing at several different scales. In addition, adjusting a geometric pattern is more than 20 times as fast as the simplification operations used in traditional automatic generalization. (For more information about geometric patterns, see Sabo, Bédard, and Bernier 2005; Sabo, Cardenas, and others 2005.)

3.2.2 Process Patterns

Not all map features can be simplified through the use of geometric patterns in the same manner. For example, for features that have complex, non-recurring shapes – such as rivers – it is difficult, or even impossible, to find a shape pattern, and consequently it is impossible to use a geometric pattern. In addition, certain patterns appear only at a small scale (e.g., at a very small scale, almost all buildings are represented by rectangles). Therefore, the principal idea behind our approach is to use generalization algorithms in place of, or as a complement to, geometric patterns in these circumstances.

Under similar conditions, objects with similar geometric, semantic, and spatial context characteristics will be generalized in approximately the same way. Generalization of these objects will require the application of the same group of algorithms, and probably in the same sequence. Accordingly, the idea is to create a set of generalization algorithms for each group of objects with similar characteristics. For example, we will have different sets of algorithms for buildings having orthogonal angles, depending upon whether they are aligned or scattered. Unlike scattered buildings, aligned buildings will require a set of algorithms that include aggregation operations. We call such a set of generalization algorithms a ''process pattern,'' since the algorithms can be repeated again and again with different parameters for different cartographic features. Thus, a process pattern is a recurrent group of generalization algorithms and sequences that is used to generalize several map features with similar characteristics. Like geometric patterns, process patterns are based on the principle of recurrence. They typically apply to groups of similar objects, and, contrary to the more traditional algorithmic-based approach, they need not be applied to the complete map layer or to an entire object class.

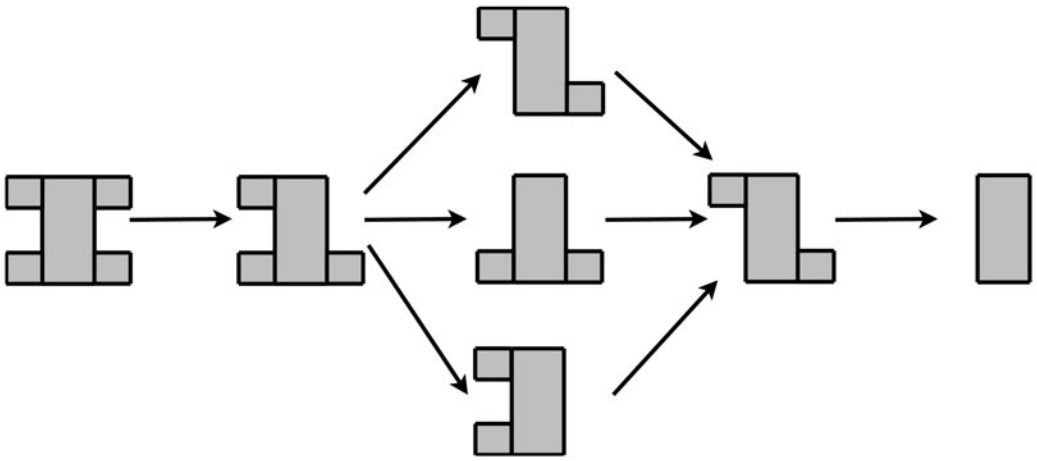

Figure 6. A geometric pattern and its different simplified forms, obtained by iterative generalization (elimination of primitives; Sabo, Bédard, and Bernier 2005).

Process patterns allow us to simplify the choice of generalization algorithms during the database-enrichment phase, as well as to overcome the limits of current cartographic generalization with respect to the choice of generalization algorithms and their sequences. The automatic choice of algorithms and their sequences is a complex task whose results are sometimes dubious. Therefore, the choice can be made during a database-enrichment process so that the generalization process can continue to benefit from the cartographers' expertise. With the integration of several algorithms in a process pattern, we now simply select the one we want during the enrichment phase, instead of having to choose individually, for each cartographic feature, the algorithms necessary for each generalization operation.

Process patterns allow us to take into account the dependency that exists between various generalization algorithms, given the interdependence of many of their operations. An example is the relationship of simplification and smoothing operations. The simplification of certain linear objects, such as rivers, generally increases their angularity, which is not acceptable for this kind of object. To correct this situation, objects are smoothed. Smoothing provides an alternative aesthetic appearance by decreasing the angularity caused by the simplification operation. In order to consider the dependency that exists between these generalization operations, we need to integrate them into the same process pattern, which can provide a select synchronization of the operations.

In some cases, it is important to break down complex generalization operations into more simple ones. This, in turn, allows a few of these simple operations to be executed during the database-enrichment phase that precedes generalization. For example, displacement is composed of many simple operations, including proximity calculation and conflict detection, calculation of the displacement vector, the displacement itself, and evaluation of the displacement's consequences. By breaking down the displacement into several simple operations, we can determine, for example, the proximity relationships and the direction of displacement in the event of a possible proximity conflict during the data-enrichment phase. Other operations can be included in a process pattern and executed on the fly during the generalization process.

A process pattern can implement either one generalization operation or several at once. In cases where it uses only one, it is called a *simple process pattern* (e.g., for an isolated building that does not require a contextual generalization operation, such as aggregation, can only implement a simplification operation when other generalization operations applied to the individual features, such as exaggeration, are neglected). However, when the process pattern implements more than one generalization operation, it is called a *complex process pattern (*e.g., where a process pattern of a building requires several operations such as simplification, exaggeration, and displacement for its generalization). Complex process patterns are composed of several simple process patterns. Moreover, the latter can consist of either a *single method* or *multiple methods*. In contrast to single-method process patterns, those having multiple methods make use of several algorithms for the same generalization operation. For example, certain simplification algorithms, such as Douglas-Peucker, are not suited for significant scale changes. Instead, they can easily be used in combination with another algorithm that is better suited to such great scale changes. Thus, depending upon the importance of the scale change, either one algorithm or the other can be used. These distinctions make the use of algorithms easier and more flexible.

A process pattern can be applied to a group of map features (e.g., aligned buildings) in order to complete a contextual generalization operation (e.g., aggregation). In this case, the process pattern is called a *group process pattern*. If the process pattern is applied to a single feature, however, it is an *individual process pattern*. The UML class model in Figure 7 shows the various types of process patterns.

3.2.3 Spatial integrity constraints

According to Elsa Maria João (1998), geometric data models for generalization are effective only if they can record spatial relationships between features. For some databases, these spatial relationships can be deduced through simple spatial queries. Unfortunately, certain important relationships (e.g., building alignment) are not explicit in spatial databases. Existing methods are not able to adequately detect all the necessary spatial relationships; moreover, this detection may be very complex and time-consuming, and the intervention of a human operator is usually required to validate the results. This situation is not acceptable for an on-the-fly map-generalization process; not only must the results be instantaneously available, but human intervention is not possible. For these reasons, the relationships must be explicit and must be generated during the database-enrichment process.

Detection of spatial relationships is a necessary but not sufficient condition for a generalization process. In order to coordinate generalization processes, we must express these spatial relationships in the form of spatial integrity constraints. Moreover, these constraints must be satisfied during the generalization processes. SGO spatial integrity constraints are rules that must always be satisfied by an SGO during a generalization process in order to ensure the coherence and reliability of the generated map's features (e.g., aligned buildings must remain aligned after generalization). Therefore, during the creation of an SGO, the spatial relationships considered to be relevant for

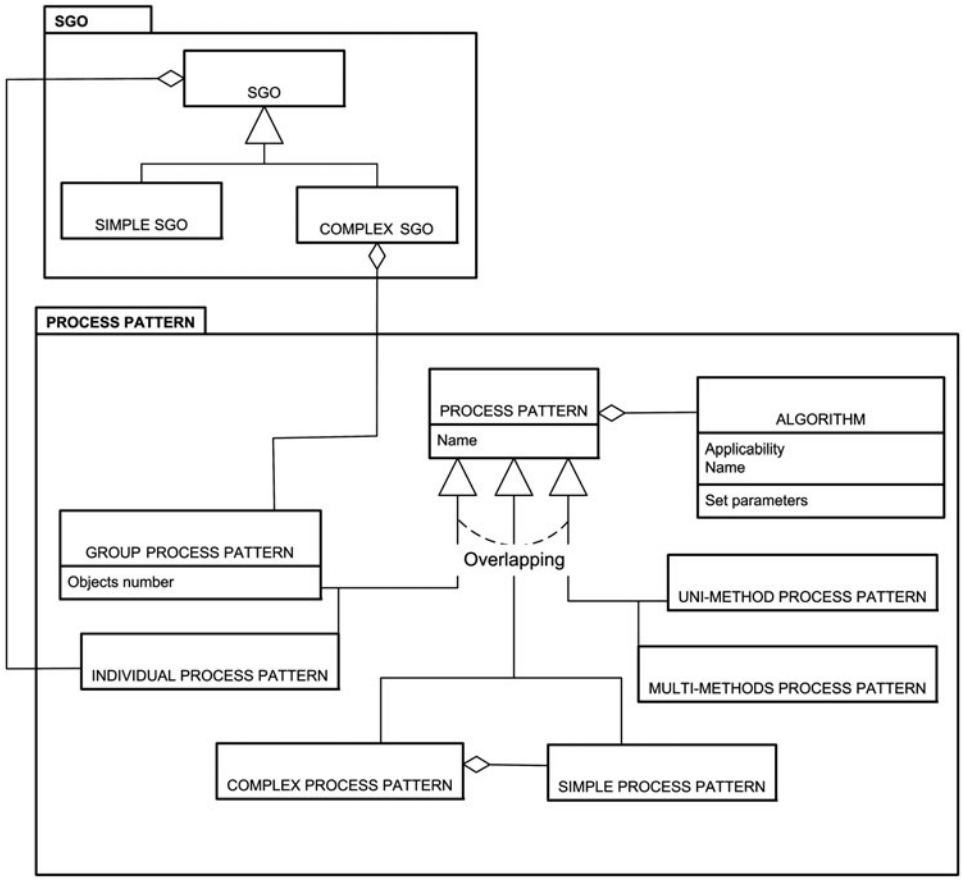

Figure 7. UML class model of the process pattern.

cartographic generalization can be defined and expressed in the form of spatial integrity constraints. The number and types of constraints to be considered depend upon the characteristics of the cartographic features. This task is performed only once, during the database-enrichment phase, rather than having to be repeated every time generalization is necessary.

For the purpose of generalization, we distinguish two separate types of spatial integrity constraints. The first, called *individual constraints*, are applied to single map features (e.g., minimal size). Such a constraint is related to the nature of the feature to which it is being applied. The second, known as *group constraints*, are applied to groups of map features (e.g., proximity constraints and alignment constraints). Unlike an individual constraint, a group constraint applies to the relationships existing between several SGOs. We differentiate two types of group constraints:

- A *binary constraint* connects two SGOs (e.g., proximity constraint). During the generalization process, a violation of this kind of constraint is solved through direct interaction between the SGOs to which it applies.
- N-*ary constraints* connect several SGOs (e.g., an alignment constraint or an inclusion constraint).

When generalization is guided by constraints, these constraints must be adapted to different situations. Map feature transformations generally affect several constraints. For example, eliminating an object that is connected to another object by a proximity constraint involves deactivating this constraint. For certain constraints, however, eliminating one object does not necessarily mean the disappearance of the constraint but may instead lead to its transformation. For instance, if several buildings are aligned along a rectilinear road, and, at a specific level of detail, that road disappears, what will happen to this constraint? First of all, the constraint specifies that the buildings be lined up at approximately the same distance from the road (because the road is rectilinear). Furthermore, if we look at this constraint as singular, the disappearance of the road should automatically have caused the disappearance of the constraint, leading to the disappearance of the alignment constraint. However, if this constraint is considered to be two distinct constraints (alignment and proximity), the elimination of the road will cause the disappearance of the proximity

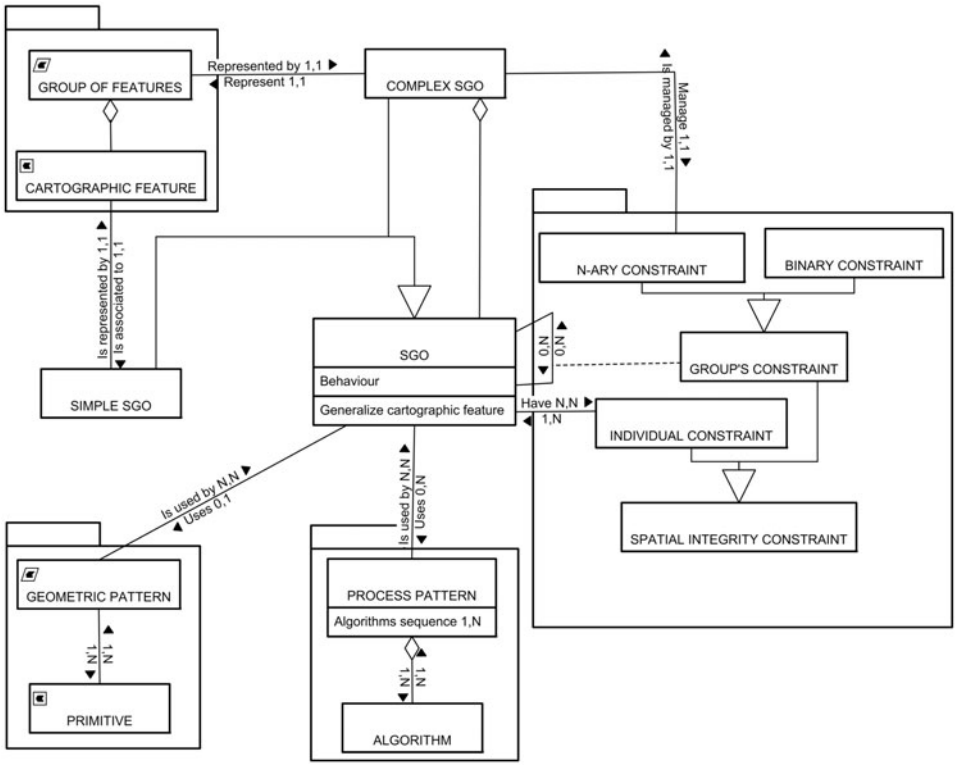

Figure 8. Simplified UML class diagram of the data structure used to manage SGOs.

constraint only. To ensure proper generalization, constraints must be sufficiently flexible to adapt to new situations, and, therefore, it is necessary to break down complex constraints into several simple constraints. Thus, each complex constraint will be an aggregate of simple constraints.

3.3 TYPES OF SGOS

For objects that are well suited for geometric patterns (e.g., simple buildings), the corresponding SGO may contain only geometric patterns. For objects that are not suited for geometric patterns (usually because their shape is not repetitive), such as hydrographic elements, the associated SGO may contain only process patterns. Finally, some map features may have an SGO that contains both geometric patterns and process patterns.

Depending upon the number of objects with which it is associated, an SGO can be either simple or complex. A *simple SGO* is associated with a single cartographic feature (e.g., a single building). In order to preserve the properties of a map's group of features (e.g., to maintain the alignment of selected buildings) and to facilitate the generalization operations applied to a group of features (e.g., aggregation), several SGOs can be grouped together to form a *complex SGO*, defined as an aggregate of several SGOs (simple and/or complex), created to handle an *n*-ary constraint. It is not necessary to create a complex SGO to handle a binary constraint (e.g., proximity constraint). The same SGO can be part of several complex SGOs. Each time an SGO enters into the composition of a complex SGO, its value increases to give it more importance when the system is faced with an elimination of map features. During the generalization process, SGOs are selected or eliminated according to their importance. Furthermore, a complex SGO can be a member of another complex SGO; there is no limit on the number of SGO imbrications. Figure 8 shows the data structure used to manage SGOs.

3.4 SGOS AND THE GENERALIZATION PROCESS

In the previous sections, we presented the static aspect of an SGO (i.e., a structure that permits the storage of all database enrichments). In this section, we present its dynamic aspect (i.e., coordination of the generalization process), including the relationships and interaction between SGOs and SGO behaviour patterns.

3.4.1 Relationships and Interactions between SGOs

In order to coordinate the generalization process while taking the environment into account, SGOs have various types of relationships that allow them to interact during the generalization process. Depending upon the type of SGOs involved, there are three categories of relationships between them:

- *Vertical relationship*: an asymmetrical relationship between a complex SGO and its members.

This relationship is asymmetrical because the complex SGO has a hierarchical level higher than that of its members. This is a parent–child relationship in which interaction can be both to and from the complex SGO.

- *Horizontal or transversal relationship*: a relationship between two SGOs at the same hierarchical level that is not a parent–child relationship. It is this kind of relationship that links SGO members belonging to the same complex SGO, whether two simple SGOs, two complex SGOs, or a complex and a simple SGO. The only conditions are that the two SGOs not be in a parent–child relationship and that they have the same hierarchical level. Thus, this kind of relation cannot connect a complex SGO with its members.
- *Oblique relationship:* a non–parent–child relationship between SGOs from different hierarchical levels. This type of relationship can be specified during the data-enrichment phase or generated automatically during the generalization process after the aggregation of several cartographic features by a complex SGO. Following this aggregation, the complex SGO inherits the relationship (e.g., proximity constraints) of its SGO members, representing the aggregated objects. Therefore, the complex SGO will be in relationship with SGOs of a lower hierarchical level.

3.4.2 Behaviour Patterns

SGOs are linked to the exact geometry of a cartographic feature and include expert knowledge in order to facilitate the generalization of the associated feature. This enrichment permits the creation of a complete behaviour pattern that allows the SGO to know how to generalize the associated cartographic feature at smaller scales. An SGO's behaviour pattern controls the order in which spatial integrity constraints are checked, the way in which a violated constraint is satisfied, and the interactions between various SGOs.

The main role of the behaviour pattern is to coordinate generalization operations. It tells the SGO which generalization method (geometric pattern vs. generalization algorithm) to use during a generalization process, as well as allowing the SGO to decide which geometric pattern to use for a given scale and how to adjust or simplify the selected geometric pattern. The need to apply a given operation is controlled by the behaviour pattern. If there are several algorithms for the same operation, the behaviour pattern selects the most suitable one. The SGO's behaviour pattern also allows the determination of the parameters of a generalization algorithm during the generalization process. Given that the sequence of the algorithms is already defined in the process pattern, the behaviour pattern allows the SGO to change this predefined sequence if necessary.

The SGO's behaviour pattern is determined by the type of SGO (simple or complex), the type of cartographic feature associated with the SGO (e.g., building feature), and the type of constraints applied to this SGO (e.g., alignment constraint). For a simple SGO, the behaviour pattern allows the coordination of generalization operations applied to a single cartographic feature (e.g., simplification, displacement). For a complex SGO, in addition to allowing generalization operations (e.g., aggregation), the behaviour pattern also enables the SGO to coordinate and/or facilitate the actions of its members in order to ensure the coherence of the group. For example, a complex SGO of aligned buildings can

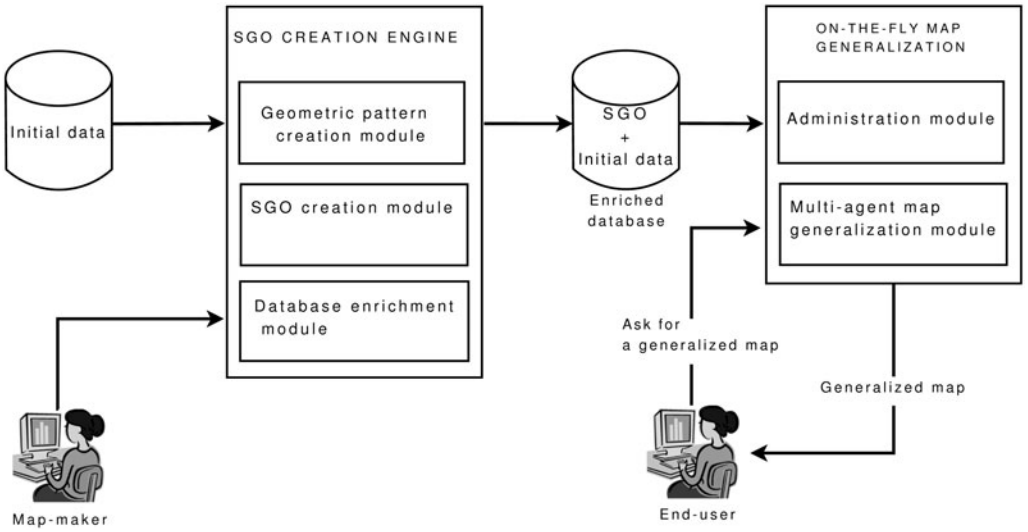

Figure 9. A simplified view of the architecture of both prototype systems: the SGO Creation Engine and the On-the-Fly Map-Generalization System.

facilitate displacement by computing the axis of alignment. When an SGO (simple or complex) is a member of a complex SGO, it has two distinct behaviour patterns; the first enables the SGO to solve its personal constraints, and the second allows it to respect the constraints imposed on all members of the group (e.g., alignment).

There are also general behaviours common to almost all SGOs (e.g., all SGOs check their internal constraints, such as minimum size, before checking their external constraints, such as proximity). The more specific behaviours depend primarily upon the type of SGO constraint (e.g., behaviour pattern of aligned buildings).

4. Applying the SGO Concept

To validate the SGO approach within the proposed framework, two prototypes were developed: (1) an *SGO Creation Engine* and (2) an *On-the-Fly Map-Generalization System*. The two prototypes were developed in Java and use an open-source API. The SGO Creation Engine is devoted to the creation of different SGOs and their components (geometric patterns, process patterns, etc.), while the On-the-Fly Map-Generalization System is dedicated to on-the-fly map generalization based on multi-agent technology, using the previously created SGOs as input. Figure 9 shows the simplified architecture of these prototypes.

4.1 A PROTOTYPE FOR THE CREATION AND ENRICHMENT OF SGOS (THE SGO CREATION ENGINE)

This prototype is based upon the Java Topology Suite (JTS) library (Vivid Solutions 2006). The JTS library is a Java API of 2D spatial predicates and functions that uses an explicit precision model and robust geometric algorithms. JTS implements the Simple Features Specification for SQL, published by the OpenGIS Consortium. For the visualization of geographic data, the prototype uses the GeoTools API (Codehaus Foundation 2006), an open-source Java GIS toolkit for developing OpenGIS-compliant solutions.

In order to test the prototype, we used data in a Shapefile format (ESRI) from Canada's Ministry of National Defence. These data are on a scale of 1:1000 and cover part of Quebec City. Two classes of objects, namely "buildings" and "roads," were used for SGO creation. These data cover 814,537 m^2 and contain 50 buildings (average building area = 408.83 m^2), distributed in six building blocks, and 52 street segments and 27 crossroads forming 21 building blocks (i.e., 15 building blocks are empty). All features of these two classes are represented by polygons.

With this prototype, the creation of SGOs is achieved in three main phases: creation of geometric- and process-pattern databases; importation of map features and creation of simple SGOs; and enrichment of created SGOs.

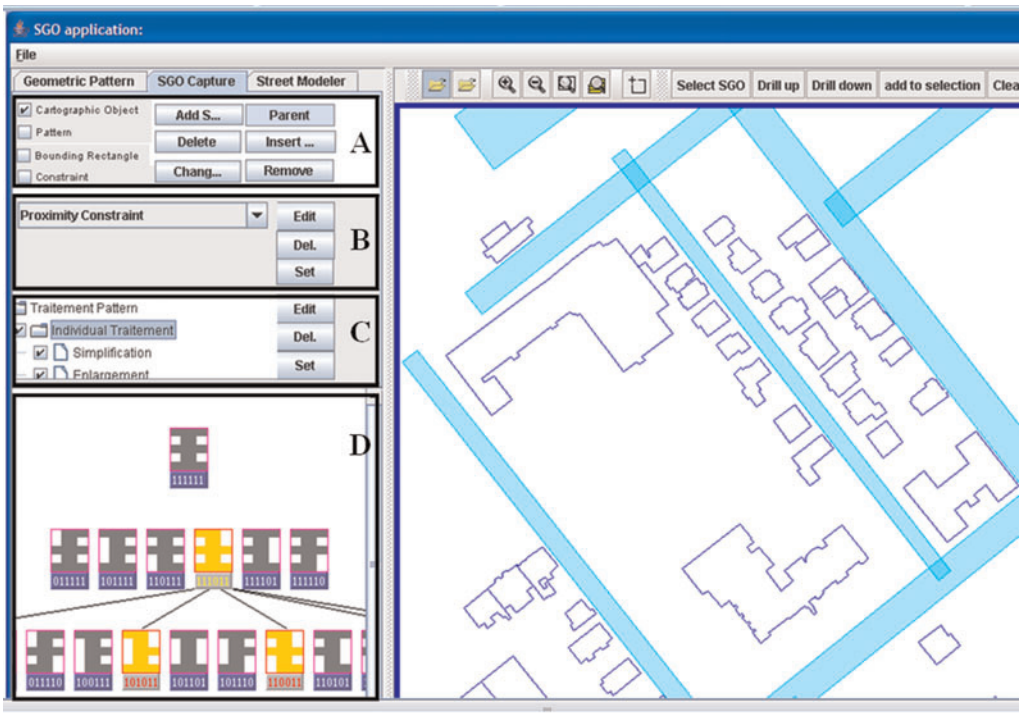

Figure 10. User interface of the prototype SGO Creation Engine. (A) Complex SGO creation panel; (B) Constraint specification panel; (C) Process pattern specification panel; (D) Geometric pattern (on top of the hierarchy) and its simplified forms.

4.1.1 Creating Geometric- and Process-Pattern Databases

During this phase, geometric and process patterns are created that will be used by the SGO. The prototype allows the automatic creation of each geometric pattern and its various simplified forms. As an input, only the number of primitives in the geometric pattern (i.e., number of juxtaposed rectangles) is used. Based on this number, specified by the operator, the system creates the geometric pattern. This geometric pattern is then automatically simplified in an iterative way in order to create all its various possible simplified forms. The geometric pattern and its simplified forms are stored in the geometric-pattern database. The result is represented in a hierarchical tree of simplified shapes (derived from the pattern; see Figure 10(D)), from which we can choose the desired geometry at the required level of detail.

This phase also allows us to interactively form the various process patterns. A process pattern is created by simply selecting from a list which generalization operations are to be included. The order of the selected operations determines the sequence in which they are to be applied during a generalization process (of course, this order can be changed by the operator in order to generate a new process pattern). For each operation selected, a list of available algorithms is presented. From this list, algorithms that will be used to carry out each generalization operation are selected. Finally, the completed process pattern is then added to a list of available patterns.

4.1.2 Importing a Map's Features and Creating Simple SGOs

The SGO creation phase is fully automated. This prototype allows us to import buildings and road-network objects from our initial data, which is in Shapefile format. For each feature imported, a corresponding SGO is created. A simple SGO (an SGO of a single cartographic feature) is created for each building feature. This automatically created simple SGO is not enriched (i.e., it does not have a geometric pattern, a process pattern, or a spatial integrity constraint). The imported road-network features are polygonal, and each street is made up of polygonal segments and junctions. Thus, for each road element (segment or junction), a simple SGO is created and automatically stored in an SGO database.

4.1.3 Enriching Created SGOs

During this phase, building and road-network SGOs (created during the previous phase) are enriched. Complex SGOs of buildings (e.g., an SGO representing a group of aligned buildings) are interactively created by grouping several SGOs together. For each complex SGO created, a spatial integrity constraint is defined that allows us to preserve the created group during the generalization process. Currently, three group constraints (i.e., constraints applied to a group of map features) are implemented: the proximity constraint (e.g., between two buildings, between a building and a street), the alignment constraint, and the block constraint, allows buildings to remain within the same block. However, created building

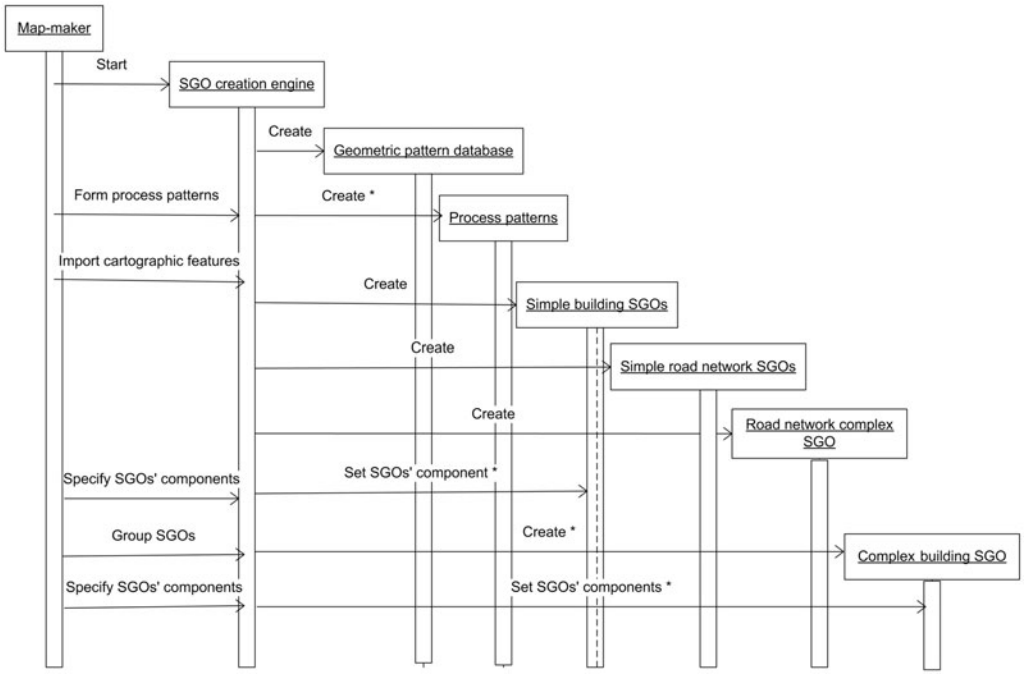

Figure 11. UML sequence diagram of database enrichment (the stars represent interactive processes).

SGOs (simple and complex) are completed by specifying their components (i.e., geometric patterns, process patterns, and spatial integrity constraints). Geometric patterns are associated with SGOs by a simple drag-and-drop operation. Furthermore, they are matched to cartographic feature geometry by clicking several control points on the cartographic feature. The system automatically extracts and stores in the SGO the parameters (i.e., the feature's position, size, and orientation) that allow us to match the geometric pattern to the geometry of the cartographic feature to be replaced. These parameters will be used later on by the SGO during the on-the-fly map-generalization process in order to replace the cartographic feature with its geometric pattern. This phase also allows the operator to associate a process pattern with a specific SGO by interactively selecting the needed process pattern from the list created during the first phase and including it in the given SGO.

Unlike enrichment of building SGOs, the enrichment of road-network SGOs is based on an automatic process. During the enrichment of the road-network SGOs, segments and junctions imported from the previous phase are automatically reconnected in order to re-create the road network. Based on simple road-network SGOs (i.e., SGOs representing segments and junctions), a complex SGO representing each road is individually created. Values are automatically allotted to each road SGO according to their importance in the network (e.g., number of connections and width). These automatically allotted parameters can be changed interactively if necessary.

Figure 10 illustrates the interface of the prototype SGO Creation Engine. This prototype allows us to create and enrich SGOs in a three-step process. Thereafter, the SGOs are used in an On-the-Fly Map-Generalization System, described in the next section. The UML sequence diagram for database enrichment is illustrated in Figure 11. The creation of the SGOs used to test this prototype took about one hour. This processing time obviously depends on the complexity of the map's objects, the ability of the human operator, and his or her familiarity with the tool. In addition, many operations carried out during the interactive process of SGO creation can be automated, and some are already in the course of being automated. Once this automation is done, the operator's intervention can be reduced to a minimum (e.g., validation of the result).

4.2 ON-THE-FLY MAP-GENERALIZATION SYSTEM

Like the previous one, this prototype was developed in Java. Its visualization tool is based on the GeoTools library (Codehaus Foundation 2006). The multi-agent module for this prototype was developed using the Java Agent Development Framework (JADE) environment (JADE 2006). JADE is a software framework, implemented in Java, that simplifies the implementation of multi-agent systems through a middleware implementing the Foundation of Intelligent Physical Agents (FIPA) specifications (FIPA 2006). This prototype supports on-the-fly production of maps at arbitrary scales. As an input, the prototype uses the SGOs created using the first prototype (see section 4.1). The present prototype is composed of two main modules: administration and on-the-fly map generation.

4.2.1 The Administration Module

This module imports the SGO file and allows the cartographer to specify constraint thresholds (e.g., minimum separation between two map symbols). Currently, five constraint thresholds (minimum object size, minimum segment length, maximum object orientation deviation, proximity, and maximum object displacement) can be specified using the administration module. During specification, the system checks the compatibility of each value and emits a visual signal (i.e., a colour change) in the case of an incompatibility. The multi-agent module is launched using this administration module. During the launching process, the system agent in charge of the multi-agent module initialization is created.

4.2.2 The On-the-Fly Map-Generation Module

This module is composed of three sub-modules:

- The *Coordination Sub-Module* contains the *system agent* responsible for the initialization of the other sub-modules. During initialization, this agent creates all the other agents of the system. For each imported SGO, a corresponding SGO agent is created, and a behaviour pattern is automatically associated with it, depending upon its constraints. The system agent also allows the user to interact with the system (e.g., by intercepting the user's requests and extracting the needed parameters, such as the required map scale). During a scale change, the system agent converts all constraint thresholds for the requested scale. The converted thresholds are sent to SGO agents.
- The *Visualization Sub-Module* contains the *drawing agent* as well as the user interface, which includes the navigation tools. The drawing agent is responsible for drawing the generalized data.
- The *Map-Generalization Sub-Module* is in charge of map generalization. This sub-module contains several SGO agents that communicate with each other in order to generalize the selected data. Generalized data are sent to the system agent, which transfers them to the drawing agent. Figure 12 illustrates the interface of the prototype on-the-fly multi-agent map-generalization system.

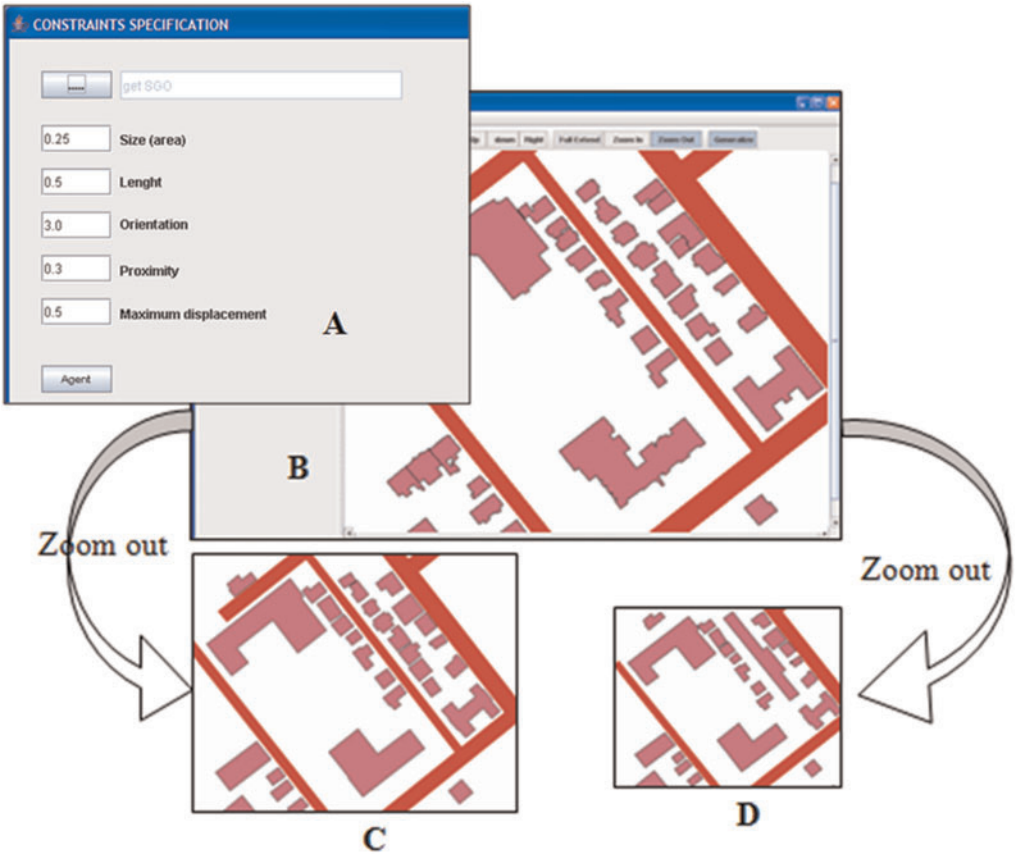

Figure 12. Interface of the administration module (A) and the map-generation module (B) of the on-the-fly multi-agent map-generalization prototype, and different map scales generated on the fly according to the map extent selected by the user (C, D).

Contrary to approaches that use a set of pre-defined scales (e.g., Google Maps), our system generalizes each map object or group of map objects on the fly, according to the level of detail of the map requested by the user. Thus, when a user selects a portion of the map, the system automatically computes the scale of the new map and the generalization parameters according to this new scale (e.g., minimum distance required between map objects to avoid superposition of map symbols). According to the characteristics of each object and the computed generalization parameters, the system determines how this object can be generalized (whether to use geometric pattern or generalization algorithms and which algorithms and parameters to use). All these generalization operations are applied on the fly thanks to the SGO.

Using this prototype, a user (even if he or she has no knowledge about cartographic generalization) can generate maps at arbitrary scales. The user has no parameters to specify, given that all parameters have been specified by the system administrator. The system allows the quasi-instantaneous generalization of data. For example, the average time necessary to generalize 100 map features is about 300 milliseconds. In terms of response time, this is compatible with the requirements of on-the-fly map generalization. The prototype was tested using a Pentium 4 CPU operating at 2.8 GHz with 512 MB of RAM (Random Access Memory) and running Microsoft Windows XP Family Edition and Service Pack 2. The number of buildings used is representative of the number of buildings seen on a computer display at large scale.

5. Conclusion and Discussion

This article presents the concept of the SGO, which integrates geometric patterns, process patterns, spatial integrity constraints, and behaviour patterns. The existence of two generalization methods (geometric patterns and generalization algorithms) used to generalize cartographic features or geometric patterns provides good flexibility in the generalization process. This approach is also based on a process of enrichment that makes explicit information that is implicit in the initial database (e.g., a building's alignment) and introduces specific generalization knowledge (e.g., the choice of the most appropriate generalization algorithms) into the data.

This enrichment process overcomes many limitations of the current automatic generalization process, such as the problem of formalizing expert knowledge. In order to validate our SGO, we developed two prototypes, one to create an SGO database and the second, using multi-agent technology, to allow the creation of on-the-fly maps at arbitrary scales (see section 4 above).

In addition to facilitating the generalization process, SGOs also speed up the process to allow on-the-fly map generalization. This significant reduction in processing time is due, on the one hand, to the database-enrichment phase, during which the process patterns and integrity constraints are defined, and, on the other hand, to the simplicity of the operations required to adjust and simplify the geometric pattern.

The approach described here is particularly well suited to cases when we need to produce several maps at varying scales (as opposed to producing only one small-scale map), because we can reuse the cartographic knowledge embedded in the SGOs. The main cost saving is that the enrichment is done only one time and the result can be used several times to generate maps at different scales. In addition, because the SGOs have a hierarchical structure, they can be used to help update data (by propagating data updates to smaller scales) or to generate a multi-scale database in which several geometries of each map object are explicitly linked, unlike the traditional multi-scale databases used in several existing applications (e.g., Google Maps), in which only map scales are linked. The explicit link between occurrences of map objects is also necessary for drilling on map objects in business intelligence applications such as SOLAP.

In order to personalize a map as required by a user during the generalization process, a system must take several factors into account (e.g., map scale, map type). In our approach, a weight is assigned to each cartographic object according to its importance in the map (e.g., the importance of a road segment within the road network). Of course, these weights can be changed according to the type of map required by the user. For example, in a routing system, a high weight can be assigned to roads forming the travel route requested by the user. This allows us, for example, to avoid a situation in which a road suddenly disappears, rendering the user's destination inaccessible. In the current prototype, only the mechanism to allow capture of users' needs (e.g., the scale of visualization and the width of the symbols used) has been implemented. In future phases of the project, however, we will consider using ontologies to model users' needs. For example, for each type of map, an ontology can be created, which will allow the system to take into account specific constraints related to the creation of each type of map (e.g., for a tourist map it is not desirable to aggregate a residential building and a building in which an art gallery is located). For more flexibility, it is possible to allow users to specify their needs using a form that is then automatically transformed into an ontology.

Thanks to the use of geometric patterns, SGOs can facilitate data transfer over the Internet. Indeed, in a client/server architecture, geometric patterns can be stored on the client side (as in the use of fonts in a word-processing application). Thus, instead of transferring a generalized cartographic feature from the server side to the client side, the system can transfer only parameters (position, size, orientation of the feature), allowing its replacement by its geometric pattern. Using these parameters, the geometric pattern can then be adjusted on the client side. The resulting reduction in network traffic, especially for online cartography applications, is of great importance, given that the data-transfer rate is known to be one of the main limitations of such applications (Bertolotto and Egenhofer 2001; Buttenfield 2002).

At first sight, the effort required to create SGOs, which implies the choice of geometric patterns, process patterns, and the creation of spatial integrity constraints for each cartographic feature or group of features, may be perceived as the weak point in this approach. However, this work is done only once and is then reused multiple times, creating significant overall time savings. Moreover, the SGO creation tool greatly simplifies this task. Although many of these operations are interactive for the moment, some may eventually be automated. For example, Desmond Rainsford and William A. Mackaness (2002) have proposed a method for the automatic recognition of alpha-numeric templates that is similar to some of our geometric patterns. In addition, certain operations, such as the choice of algorithms, are already performed during a conventional interactive generalization process. In this case, the task will consist of creating a mechanism to record this work and store it in an SGO database by coupling the system described here with an existing generalization system.

Acknowledgements

The authors are grateful to the Canadian network of excellence GEOIDE, which made this study within the framework of the GEMURE and MUSCAMAGS projects possible. We are also grateful to the Canada NSERC Industrial Research Chair in Geospatial Database for Decision-Support and to the following partners: Defence Research and Development Canada, Natural Resources Canada, Ressources Naturelles et Faune Québec, and Intergraph.

Author Information

Mamane Nouri Sabo, Canada NSERC Industrial Research Chair in Geospatial Databases for Decision Support,

Centre for Research in Geomatics, Université Laval, Québec, QC G1K 7P4 Canada. E-mail: mamane-nouri.sabo.1@ulaval.ca.

Yvan Bédard, Canada NSERC Industrial Research Chair in Geospatial Databases for Decision Support, Centre for Research in Geomatics, Université Laval, Québec, QC G1K 7P4 Canada. E-mail: yvan.bedard@scg.ulaval.ca.

Bernard Moulin, Centre for Research in Geomatics, Université Laval, Québec, QC G1K 7P4 Canada. E-mail: bernard.moulin@ift.ulaval.ca.

Eveline Bernier, Canada NSERC Industrial Research Chair in Geospatial Databases for Decision Support, Centre for Research in Geomatics, Université Laval, Québec, QC G1K 7P4 Canada. E-mail: eveline.bernier@scg.ulaval.ca.

References

Allouche, M.K., and B. Moulin. 2005. "Amalgamation in Cartographic Generalization Using Kohonen's Feature Nets." *International Journal of Geographical Information Science*. 19: 899–914.

Armstrong, M.P. 1991. "Knowledge Classification and Organization." In *Map Generalization: Making Rules for Knowledge Representation*, ed. B.P. Buttenfield, R. McMaster, and H. Freeman. Harlow, UK: Longman Scientific. 86–102.

Baeijs, C., Y. Demazeau, and L. Alvares. 1995. "Application des systèmes multi-agent à la généralisation cartographique." In *3èmes Journées Francophones sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-agents, AFCET&AFIA,* Chambery. 163–76.

Beard, K.M. 1991. "Contraints on Rule Formation." In *Map Generalization: Making Rules for Knowledge Representation*, ed. B.P. Buttenfield, R. McMaster, and H. Freeman. Harlow, UK: Longman Scientific. 121–35.

Bédard, Y., E. Bernier, and R. Devillers. 2002. "La métastructure VUEL et la gestion des représentations multiples." In *Généralisation et représentation multiple*, ed. A. Ruas. Paris: Hermes. 149–62.

Bernier, E., and Y. Bédard. 2007. "A Data Warehouse Strategy for On-Demand Multi-scale Mapping." In *Generalization of Geographic Information: Cartographic Modelling and Application*, ed. W. Mackaness, A. Ruas, and L.T. Sarjakoski. Kidlington, UK: Elsevier. 177–98.

Bertolotto, M., and M. Egenhofer. 2001. "Progressive Transmission of Vector Map Data over World Wide Web." *Geoinformatica*. 5: 345–73.

Buttenfield, B.P. 2002. "Transmitting Vector Geospatial Data across the Internet." *Lecture Notes in Computer Science*. 2478: 51–64.

Cecconi, A., R. Weibel, and M. Barrault. 2002. "Improving Automated Generalisation for On-Demand Web Mapping by Multiscale Database." In *Proceedings of Symposium on Geospatial Theory, Processing and Applications, July 8–11, Ottawa, Canada.* 1–9.

Codehaus Foundation. 2006. GeoTools: The Open-Source Java GIS Toolkit. Available at http://geotools.codehaus.org/Home.

Devogele, T., T. Badard, and T. Libourel. 2002. "La problématique de la représentation multiple." In *Généralisation et représentation multiple*, ed. A. Ruas. Paris: Hermes. 55–74.

Devogele, T., J. Trevisan, and L. Raynal. 1996. "Processus de constitution d'une base de données multi-échelles." *Revue Internationale de Géomatique*. 6: 249–63.

Douglas, D.H., and T.K. Peucker. 1973. "Algorithms for the Reduction of the Number of Points Required to Represent a Digitized Line or Its Caricature." *The Canadian Cartographer*. 10/2: 112–22.

Foundation for Intelligent Physical Agents [FIPA]. 2006. FIPA home page. Available at http://www.fipa.org/.

Jabeur, N. 2006. "A Multi-agent System for On-the-Fly Web Map Generation and Spatial Conflict Resolution." PhD diss., Université Laval.

Jabeur, N., B. Boulekrouche, and B. Moulin. 2006. "Using Multi-agent Systems to Improve Real Time Mobile Map Generation." In *Advances in Artificial Intelligence*, ed. L. Lamontagne, and M. Marchand. Berlin: Springer. 37–48.

Java Agent Development Framework [JADE]. 2006. JADE home page. Available at http://jade.tilab.com/.

João, E.M. 1998. *Causes and Consequences of Map Generalisation*. London: Taylor & Francis.

Lamy, S., A. Ruas, Y. Demazeau, M. Jackson, W.A. Mackaness, and R. Weibel. 1999. "The Application of Agents in Automated Map Generalisation." In *Proceedings of the 19th International Cartographic Conference, Ottawa, Canada*, vol. 2. 1225–234.

Li, M., S. Zhou, and B. Jones. 2002. "Multi-agent Systems for Web-Based Map Information Retrieval." *Lecture Notes in Computer Science*. 2478: 161–80.

Mackaness, W.A. 1994. "An Algorithm for Conflict Identification and Feature Displacement in Automated Map Generalization." *Cartography and Geographic Information Systems*. 21: 219–32.

Mackaness, W.A., and G. Edwards. 2002. "The Importance of Modeling Pattern and Structure in Automated Map Generalization." Paper read at the Joint ISPRS/ICA Workshop on Multi-scale Representations of Spatial Data, 7–8 July, Ottawa, ON.

Martel, C. 1999. "Développement d'un cadre théorique pour la gestion des représentations multiples dans les bases de données spatiales." M.Sc. thesis, Université Laval, Québec, QC.

McMaster, R. 1991. "Conceptual Framework for Geographical Knowledge." In *Map Generalization: Making Rules for Knowledge Representation*, ed. B.P. Buttenfield, R. McMaster, and H. Freeman. Harlow, UK: Longman Scientific. 21–39.

Miniwatts Marketing Group [MMG]. 2006. World Internet Users and Population Stats. Available at http://www.internetworldstats.com/stats.htm.

Müller, J.C. 1991. "Building Knowledge thanks for Rule Based Generalization." In *Proceedings of International Cartographic Conference,* Bournemouth. 257–65.

Newell, A. 1990. *Unified Theories of Cognition*. Cambridge, MA: Harvard University Press.

Nickerson, B.G., and H. Freeman. 1986. "Development of a Rule-Based System for Automatic Map Generalization." In *Proceedings of the 2nd International Symposium on Spatial Data Handling, Seattle, WA.* 537–56.

Rainsford, D., and W.A. Mackaness. 2002. "Template Matching in Support of Generalization of Rural Buildings." In *Proceedings of Symposium on Geospatial Theory, Processing and Applications, Ottawa, Canada*, 9–12 July, 2002. Available at www.isprs.org/commission4/proceedings02/pdfpapers/137.pdf.

Ruas, A. 1999. "Modèles de généralisation de données géographiques à base de contraintes et d'autonomie." PhD diss., Université de Marne-la-Vallée, France.

Sabo, M.N., Y. Bédard, and E. Bernier. 2005. "Methodology for Developing a Database of Geometric Patterns to Better Support On-the-Fly Map Generalization." Paper read at the International Cartographic Conference, 9–16 July, A Coruña, Spain.

Sabo, M.N., A. Cardenas, Y. Bédard, and E. Bernier. 2005. "Introduction du concept de patrons géométriques et application aux bâtiments afin de faciliter leur généralisation cartographique à la volée." *Geomatica*. 59: 295–311.

Sarjakoski, T. 2007. "Conceptual Models of Generalization and Multiple Representation." In *Generalization of Geographic Information: Cartographic Modelling and Application*, ed. W. Mackaness, A. Ruas, and L.T. Sarjakoski. Kidlington, UK: Elsevier. 11–36.

Sarjakoski, T., and T. Kilpelainen. 1999. "Holistic Cartographic Generalization by Least Squares Adjustment for Large Data Sets." In *Proceedings of the 19th International Cartographic Conference, Ottawa, Canada.* 1091–98.

Sester, M. 2000. "Generalization Based on Least Squares Adjustment." *International Archives of Photogrammetry and Remote Sensing*. 33B: 931–38.

Timpf, S. 1998. "Hierarchical Structures in Map Series." PhDdiss., Technical University of Vienna.

Vangenot, C. 1998. "Représentation multi-résolution, concepts pour la description de bases de données avec multi-représentation." *Revue internationale de géomatique*. 8: 121–47.

Van Oosterom, P., and V. Schenkelaars. 1995. "The Development of an Interactive Multi-scale GIS." *International Journal of Geographical Information Systems*. 9: 489–507.

Vivid Solutions. 2006. JTS Topology Suite. Available at http://www.vividsolutions.com/jts/jtshome.htm.

Weibel, R. 1991. "Amplified Intelligence and Rule-Based Systems." In *Map Generalization: Making Rules for Knowledge Representation*, ed. B.P. Buttenfield, R. McMaster, and H. Freeman. Harlow, UK: Longman Scientific. 172–86.

Weibel, R., E. Bernier, Y. Bédard, and A. Cecconi. 2002. "La généralisation à la volée." In *Généralisation et représentation multiple*, ed. A. Ruas. Paris: Hermes. 319–35.

Weibel, R., and B.P. Buttenfield. 1988. "Map Design for Geographic Information Systems." In *Proceedings of GIS/LIS, 30 November–2 December, San Antonio, Texas*. 350–59.

———. 1998. "Constraints-Based Automated Map Generalization." In *Proceedings of the 8th International Symposium on Spatial Data Handling, Vancouver, Canada*. 214–24.

Weibel, R., and G. Dutton. 1999. "Generalising Spatial Data and Dealing with Multiple Representations." In *Generalising Spatial Data and Dealing with Multiple Representations*, ed. P.A. Longley, M.F. Goodchild, D.J. Maguire, and D.W. Rhind. Chichester, UK: Wiley. 125–55.

Young, J., and S. Smith. 2006. "Akamai and Jupiter Research Identify '4 seconds' as the New Threshold of Acceptability for Retail Web Page Response Times." Press release, Akamai Industries. Available at http://www.akamai.com/html/about/press/releases/2006/press_110606.html.