TOWARDS SELF-GENERALIZING OBJECTS AND ON-THE-FLY MAP GENERALIZATION

Mamane Nouri SABO$^{1,2,3}$, Yvan BÉDARD$^{1,2,3}$, Bernard MOULIN$^{2,3}$, Eveline BERNIER$^{1,2,3}$

$^1$Canada NSERC Industrial Research Chair in Geospatial Databases for Decision-support
$^2$Centre for Research in Geomatics, Université Laval
$^3$Université Laval, Québec, Canada G1K 7P4
mamane-nouri.sabo.1@ulaval.ca yvan.bedard@scg.ulaval.ca bernard.moulin@ift.ulaval.ca
eveline.bernier@scg.ulaval.ca

Abstract:
Map generalization is a complex task which requires sometimes human intervention. In order to support such a process on-the-fly, we propose a generalization approach based on Self-Generalizing Objects (SGOs) which 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. Then, each created SGO is transformed into a software agent (SGO agent) in a multi-agent on-the-fly map generalization system. SGO agents are equipped with behaviours which enable them to coordinate the generalization process. In this paper, we present the concept of SGO and two prototypes that we developed in order to support this approach: a prototype for the SGOs’ creation and another for the on-the-fly map generalization (which uses the created SGOs).

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.
1. Introduction

In the past, map production was a cartographer’s responsibility because of the specific skills required to create maps. Users received the maps created in advance, generally produced in a series. However, the technological developments of the last decade have generated a new type of media for spatial data dissemination, the 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 the users’ needs.

This democratization of cartographic data has resulted in new users’ requirements for visualization tools and geographic data. Therefore, beyond a simple static visualization, users expect flexible querying of the data and more powerful map personalization tools. Most often, they want to define parameters such as map scale, symbology, map content, and to highlight individual cartographic features with a different color, a more detailed shape or a new symbol in order to get maps better adapted to their needs. Currently, the management of symbols and map content is rather well supported by GISs. However, these tools have limited capabilities 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 (it 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, it cannot be done in advance using existing specialized batch-oriented generalization modules. Similarly, Web mapping sites such as Google map are deprived of real map generalization capabilities since they rely on the selective display of one map amongst several predefined maps representing different scales. Zooming within the selected map provides no generalization capability, zooming at a scale outside of the range of the selected map brings another pre-built map. Furthermore, they are typically very limited regarding map customization since users cannot select or highlight individual objects, nor can they perform other key generalization features (e.g. object displacement, object elimination, changing symbols, enlarging symbols without enlarging the scale). However, considering today’s state-of-the-art, GIS and web mapping solutions represent a useful and necessary evolution, especially for on-line map users.

According to 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 on-line shopping (Young and Smith 2006). Not all users require a quasi-instantaneous response time, but certain applications do (ex. SOLAP). Who would object to the idea of seeing their maps generated on-the-fly and efficiently provide results better tailored to the displayed scale or better customized to their exact needs? The SGOs aim at providing a solution towards this ultimate objective. The ideal solution for helping users to create maps with a specific content and at a desired scale would be to have a single, large-scale database and to derive maps at
smaller scale according to users' needs using automatic cartographic generalization processes. However, in order to derive maps of a smaller scale according to the users' needs, automatic cartographic generalization processes are often deployed. In conventional cartography, map generalization is responsible for reducing map complexity through a scale reduction process. This process emphasises the essential, while suppressing the unimportant; it maintains logical and unambiguous relations between map objects, and preserves the aesthetic quality (Weibel and Dutton 1999).

During the last decades, several projects have dealt with cartographic generalization (Douglas and Peucker 1973; Mackaness 1994; Lamy and others 1999; Sarjakoski and Kilpeläinen 1999; Sester 2000; Weibel and Dutton 1998; etc.). 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 earlier period, generalization operations were typically applied to all occurrences of an object class, no matter what the object’s characteristics or context were. The expected power of the used algorithms was to include the necessary parameters for all possible geometries that may be encountered in any given context. Except in fully controlled and well-defined situations, this is an impossible objective. Map generalization is a complex process, requiring a coordinated usage 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 approach 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 (Baeriswyl, Demazeau, and Alves 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 for generalization to take place (Müller 1991). Unfortunately, cartographic knowledge is very difficult to formalize given that map generalization is a subjective and holistic process.

Despite the significant progress that has been made over the past few years, automatic cartographic generalization remains a difficult task, still requiring human intervention (Sabo and others 2005). For applications such as Web mapping, that require extremely fast response times, today's alternative is to use a multiple representations database to simulate on-the-fly map generalization. In its simplest, but most common form, a multi-representations database (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 (although rarely implemented), Bédard, Bernier, and Devillers (2002) present the MRDB as the storage unit for various geometry, semantics, and graphics of the same object. Thus, according to the user's request, the most suitable representations are selected from the multi-representations database 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. This is usually based upon simple enlargement/reduction of the presented map. When the enlargement/reduction ratio reaches a large number (e.g. 5X),
the represented map is replaced by another map, which may differ significantly in the degree of its generalization. In such an approach, map scales are predefined and do not allow users to get the data corresponding 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 upon geometric patterns (shapes representative of several cartographic features), combined with sets of generalization algorithms (applicable to patterns and cartographic features) and spatial integrity constraints, all encapsulated in a single structure called an SGO (Self-Generalizing Object). This approach allows us to include additional human expertise in an efficient way at the level of individual cartographic features, which then leads to database enrichment that better supports automatic generalization. We begin this paper with an overview of Web mapping and on-the-fly generalization concepts. Then, we 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: first, the SGO’s Creation Engine, and secondly, the On-the-fly Map Generalization System. The latter is based upon a multi-agent approach and allows us to generate maps at arbitrary scales. Finally, we conclude by presenting the perspectives and the 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 the specific users’ requirements, in contrast to maps generated in mass quantities for the generic needs (e.g., topographic maps) of large groups. The user’s requirements may concern the map’s scale, its content, the objects to be highlighted or filtered, the symbols to be used, the time necessary for delivery, etc. This 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 having an adequate level of abstraction, a content and a graphic representation suited to their needs, while taking into account the specifications of each user’s visualization tools and hardware constraints (e.g., screen size and network bandwidth).

Traditional cartography is not unfamiliar with on-demand maps. Indeed, there are several types of maps with low printing volumes in traditional cartography; those which are used in extremely specialized fields are typically generated on demand. The main difference between traditional and today's, on-demand mapping is that traditional on-demand mapping tries to satisfy a small group of users from a specific field, whereas on-demand mapping from 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, there are many more constraints that must be taken into account.

The digital solutions that are currently available typically use a unique and homogenous level of abstraction for their maps. Changing the level of abstraction generally means
changing the map for another one created using a different scale. However, as manual cartography has already shown, the level of abstraction does not need to be uniform within a map. For example, in 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 using a higher level of abstraction. Indeed, except for topographic maps that try to find a certain balance in terms of levels of abstraction for all the different categories of map features, the other types of maps usually present heterogeneous levels of abstraction.

Recently, the number of Internet users exceeded the one billion mark (Internet World Stats 2006), which represents 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 helped with the development and democratization of geographic data, it has also brought with it new production challenges with respect to technological issues (e.g., data transfer rates and map supports), and user issues (e.g., heterogeneous users and users without basic cartographic knowledge). Consequently, it is impossible to totally know in advance the user’s context, or the kind of customization he or she will require. For example, in order to adapt the abstraction level, cartographic generalization is necessary. With an interactive and dynamic media such as the Internet, the concept of a "map on demand" has become synonymous to "map generated in real time," given 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). 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. Besides, in order to produce maps suited to a user’s requests, on-the-fly map generalization must also be flexible enough to take into account numerous factors.

One of the main problems associated with on-the-fly map generalization is to determine whether or not the generated map is suited to the user’s needs. In conventional cartography, the validation of results, in accordance with the producer’s specifications is done a posteriori by the map producer. However, in the case of on-the-fly map generalization, data producers have limited control over the content of resulting maps and, consequently, over the quality of the generalized data. What’s more, most users of Web mapping applications have little or very limited knowledge about the map making process, and they cannot assess the quality of the resulting product (and thus, the quality of the map generalized on-the-fly). Consequently, on-demand map production tools in
general, and on-the-fly map generalization processes in particular, must generate '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 upon generalization algorithms

This approach is exclusively based upon specific generalization algorithms (e.g. Douglas-Peucker 1973). However, as previously mentioned, 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 still nevertheless necessary to find methods that can accelerate processes in order to support on-the-fly map generalization. This explains why certain authors propose methods based upon pre-computed attributes in order to accelerate the generalization process (e.g. Van Oosterom and Schenkelbaars 1995). Unfortunately, these approaches do not take into account the local spatial environment of the map's objects. Furthermore, due to its complexity, generalization cannot be carried out by simply applying generalization algorithms sequentially without taking into account the objects' spatial neighbourhood. Finally, existing federative generalization methods based upon artificial intelligence techniques (e.g., the multi-agent approach) are very complex and require long treatments and adjustments, which currently limit their use for on-the-fly map generalization.

2.2.2.2. Multi-representations approach

As previously mentioned, the ideal solution to allow users to get data at the desired level of abstraction would be a unique database at the most detailed level of cartography from which we could automatically create products at smaller scales. However, given the current state of automatic map generalization, this is still impossible. Therefore, some researchers use multi-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 from the manual or semi-automatic generalizations of the most detailed representations. Most of the time, this approach leads to a multi-scale database, including several map layers at different scales. Then, according to the user’s request, the appropriate scale is selected. Throughout recent years, much effort has been devoted to manage multi-representations (Martel 1999; Bédard 2002; Devogele 2002; Vangenot 1998; Timpf 1998, Sarjakoski 2007; Bernier and Bédard 2007). In terms of personalization, multi-representations are extremely limited because all scales need to be predefined. Other important problems related to multiple representations are the difficulty to create necessary map scales and to
structure such a database, data redundancy as well as the difficulty to propagate 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 where the system selects the representation that is closest to the user's desired map representation (based upon 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 because the smaller the difference between the initial map scale and the desired one, the easier the generalization process. However, to be truly efficient, 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 automatic generalization and multiple representations. It is in this context that we propose an approach combining generalization algorithms and geometric patterns, a type of multi-representations where data redundancy is kept to a minimum.

### 3. 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 of generalization based upon a fast generalization method (geometric patterns) and database enrichment, all encapsulated in an object called SGO (Self-Generalizing Object). This enrichment that exploits human expertise allows us to both generate explicit spatial information (e.g., the alignment of buildings), which is implicit in the initial database, and to introduce into the database, the generalization knowledge (e.g., procedural knowledge) that is difficult to formalize.

An SGO, or Self-Generalizing Object, can be defined as an object based upon fast generalization methods and database enrichment, in addition to being 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 (recurrent sets of generalization algorithms and their sequences) (see Section 3.2.2); 3) Spatial integrity constraints (see Section 3.2.3); behaviour pattern (in order to coordinate generalization process) (see Section 3.4.2).
Thus, during a database enrichment phase, the Operator can create (interactively and/or using automatic routine) an SGO for every map feature and enrich it by specifying its components (e.g., geometric patterns, process patterns, spatial integrity constraints and behaviour pattern). Once created and enriched, an SGO contains the necessary tools, data and knowledge that enable it 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 that, SGOs are equipped with behaviours which enable them to coordinate the generalization process and to interact with other SGOs. To solve a violated constraint, an SGO uses geometric patterns and/or process patterns as generalization methods. Although the SGO’s enrichment can be considered as additional work for cartographers, it is important to mention that this work is done only once, and created SGOs are possibly used thousands of times (i.e., every time map generalization occurs).

The difficulty in automating the generalization process, particularly the difficulty in formalizing generalization knowledge, has led some researchers to propose the generalization approach, allowing the division of tasks between humans and computers (Weibel 1991; Weibel and Buttenfield 1988). This division of tasks is achieved by using interactive generalization tools. By using these tools, the cartographer can: select the objects or groups of objects to be generalized; determine the generalization operations, algorithms, and parameters to be applied; assess the quality of the generalized data; and finally, detect the conflicts and eliminate them. Unfortunately, even today, the cartographer’s work is partially lost because typically, only the generalized map is stored in the database, but not the tasks that he or she performed (e.g. the chosen sequence of generalization algorithms). Sometimes it is even necessary to redo all the tasks in order to generate a map of the same territory, but only using another scale. The philosophy underlying the SGO approach is to either recover or reproduce the work necessary for map generalization, and to store it. Therefore, the SGO can provide means to recover the work completed by the experts during the generalization process. Figure 1 demonstrates the philosophy underlying the SGO’s concept.

Figure 1. The SGO philosophy.

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, several phenomena or 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 picture). Thus, the geometric pattern (Figure 2, right picture) is a geometrical construction of a shape pattern (i.e., a geometrical construction of a recurring shape). The created geometric patterns are then stored in a geometric patterns database.
Figure 2. Detected patterns of shape on a map (left picture) and their corresponding geometric patterns (right picture).

The idea is to associate a geometric pattern with each cartographic feature (when it is possible). When a geometric pattern is associated with a cartographic feature, parameters like the feature’s orientation, position, and size are defined and stored. Then, during the generalization process, instead of generalizing this feature using the 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 pattern is adjusted using simple operations (rotation, positioning, and scaling) and stored parameters (e.g., position, size, and orientation). This solution can be taught of as a combination of a lookup table (stored geometric patterns) and a little 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.

Figure 3. Use of a geometric pattern in a cartographic generalization process: A - Initial map; B – Initial feature replaced by a geometric pattern; C - Pattern's orientation; D - Pattern's scaling; E – Result.

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 several connected primitives (e.g., juxtaposed rectangles). This flexibility facilitates the generalization of the geometric pattern, which in turn allows us to generate a pattern’s simplified form. When generalizing a geometric pattern, the operation is under control thanks to 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. These operations are: elimination of a primitive, combination of primitives, and changing the size of a primitive. A UML class model of our data structure for a geometric pattern is illustrated in Figure 4. Figure 5 shows some of the geometric pattern’s generalization operations. Finally, Figure 6 shows us a geometric pattern, along with its simplified forms.

Figure 4. UML class diagram for the geometric pattern’s data structure.

Figure 5. Geometric pattern’s generalization operations.

Figure 6. A geometric pattern and its different simplified forms obtained by iterative generalization (primitive's elimination) (Sabo, Bédard, and Bernier 2005).
The concept of geometric pattern is not limited to buildings. It may be also applied to other classes of objects (e.g., road cloverleaves, cul-de-sacs, and silos). For instance, a highly sinuous road segment, such as those found in the Alps, can be generalized using 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 dataset from Quebec City have shown that one geometric pattern and its simplified forms allow replacing 2844 individual buildings or 365 groups of buildings appearing at several different scales. In addition, adjusting a geometric pattern is more than 20 times faster than simplification operations used in traditional automatic generalization. For more information about geometric patterns, see Sabo, Bédard, and Bernier 2005 and Sabo 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 having 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 only appear on at small scale (e.g., at a very small scale, almost all buildings are represented by a rectangle). 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 would require the application of the same group of algorithms, and probably in the same sequence. Accordingly, the idea is to create, for each group of objects with similar characteristics, a set of generalization algorithms. For example, we will have different sets of algorithms for buildings having orthogonal angles, depending upon whether they are aligned or scattered. Contrary to the algorithms for 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 they 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. As with geometric patterns, process patterns are also based upon the principle of recurrence. They typically apply to groups of similar objects, and contrary to the more traditional algorithmic-based approach, they do not have to be applied to the complete map layer or to an entire object class.

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 where the results are sometimes dubious. Therefore, the choice can be made during a database
enrichment process in order to continue benefiting 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 individually choose 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 inter-dependence of many of their operations, as it is the case for 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, they need to be integrated 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: the proximity calculation and conflict detection, calculation of the displacement vector, the displacement itself, as well as the evaluation of the displacement’s consequences. By breaking down the displacement into several simple operations, we can determine such things as 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 or several generalization operations at once. In the case 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 Multi-methods. In contrast to Single-method process patterns, those having Multi-methods make use of several algorithms for the same generalization operation. For example, certain simplification algorithms, such as Douglas-Peucker, are not suited for great 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 or the other algorithm can be used. These distinctions make use of algorithms that are 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 Group Process Pattern. But if the process pattern is applied
to a single feature, it is an *Individual Process Pattern*. The UML class model of Figure 7 shows the various types of process patterns.

**Figure 7. UML class model of the process pattern.**

### 3.2.3. Spatial integrity constraints

According to João (1998), geometric data models for generalization are only effective if they can record spatial relationships between features. For several databases, these spatial relationships can be deduced through simple spatial queries. Unfortunately, certain important relationships are not explicit in spatial databases (e.g., building alignment). Not only are existing methods not able to adequately detect all the necessary spatial relationships, but this detection may be very complex and time-consuming sometimes, and usually requires the intervention of a human operator 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 generated during the database enrichment process.

Detection of spatial relationships is a necessary condition, but is insufficient for a generalization process. In order to coordinate generalization processes, these spatial relationships must be expressed 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 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 cartographic generalization can be defined and expressed in the form of spatial integrity constraints. The quantity and types of constraints to be considered depends upon the characteristics of the cartographic features. This task is only done 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 category is called *"Individual Constraints,"* which 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. Secondly, we have the *"Group Constraints"* that are applied to groups of map features (e.g., proximity constraints and alignment constraints). Contrary to an *Individual Constraint*, a *Group Constraint* applies to the relationships existing between several SGOs. We differentiate two types of *Group Constraints*:

- **Binary Constraint** connects two SGOs (e.g., proximity constraint). During the generalization process, the 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, the elimination of an object that is connected to another object by a proximity constraint involves the deactivation of this constraint. However, for certain constraints, the elimination of one object does not necessarily mean the disappearance of the constraint, but rather its transformation. For instance, if several buildings are aligned along a rectilinear road, and at a specific Level of Detail, the road disappears, what will happen to this constraint? First of all, this 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 being only one, the disappearance of the road should have automatically 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 only generate the disappearance of the proximity constraint. Thus, to ensure proper generalization, constraints must be sufficiently flexible to adapt to new situations. Therefore, the breakdown of complex constraints into several simple constraints is necessary. Thus, complex constraints 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 only contain 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 can only contain process patterns. Finally, some map features can have an SGO that contains both geometric patterns and process patterns.

According to 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. A Complex SGO can be 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 faced with an elimination of map features. During the generalization process, SGOs are selected or eliminated according to their importance. What’s more, a complex SGO can be a member of another complex SGO; there is no limit on number of SGO imbrications. Figure 8 shows the data structure to manage SGOs.

---

**Figure 8.** Simplified UML class diagram of the data structure to manage SGO.
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 the dynamic aspect of an SGO (i.e., coordination of the generalization process). We will present 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 into account the environment, 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 where interaction can be both to and from the Complex SGO.

- **Horizontal or Transversal Relationship**: a relationship between SGOs having the same hierarchical level (but it is not a parent-child relationship). It is the kind of relationship that links SGO members belonging to the same Complex SGO. It can connect two Simple SGOs, two Complex SGOs, or a Complex and a Simple SGO. The only conditions are that the two SGOs are not 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 its 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 pattern

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 behaviour patterns is to coordinate generalization operations. They know which generalization method (geometric pattern vs. generalization algorithm) to use during a generalization process. The behaviour pattern also allows 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 previously defined in the process pattern, the behaviour pattern allows the SGO to change this predefined sequence if necessary.

The SGO behaviour pattern is determined by the type of the SGO (e.g. Simple or Complex), the type of the cartographic feature associated with the SGO (e.g. building feature) and the type of constraints which are 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 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 behaviours patterns. The first behaviour pattern which enables the SGO to solve its personal constraints and the second which allows it to respect the constraint 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 upon multi-agent technology, using the previously created SGOs as input. Figure 9 shows the simplified architecture of the developed prototypes.

Figure 9. A simplified view of the architecture including both developed systems: the SGO Creation Engine and the On-the-fly Map Generalization System.
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 (JTS 2006). The JTS library is a Java API of 2D spatial predicates and functions using an explicit precision model and robust geometric algorithms. JTS implements the Simple Features Specification for SQL, published by the Open GIS Consortium. For the visualization of geographic data, the prototype uses the GeoTools API (GeoTools 2006), which is 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 the Ministry of National Defence (Canada). These data are on a scale of 1:1000 and cover a part of Quebec City. Two classes of objects, namely "buildings" and "roads", were used for the SGO creation. These data cover 814,537 m² and contain 50 buildings (average building area 408.83 m²) distributed in 6 building blocks and 52 street segments and 27 crossroads forming 21 building blocks (so, 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.

4.1.1. Creation of geometric and process pattern databases

During this phase, the geometric and process patterns are created in order to 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 the geometric pattern’s primitives (i.e., number of juxtaposed rectangles) is used. Based on this number, specified by the Operator, the system creates the geometric pattern. After that, this geometric pattern is 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’s database. The result is represented in a hierarchical tree of simplified shapes (derived from the pattern, see Figure 10-D), where we can choose the desired geometry having 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 which generalization operations are to be included from a list box. 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 selected operation, a list of available algorithms is presented. From this list, algorithms which 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. Importation of a map’s features and the creation of 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 a Shapefile format. For each imported feature, a corresponding SGO is created. A Simple SGO (an SGO of a single cartographic feature) is created for each building feature. What’s more, this automatically created simple SGO is not enriched (i.e., without a geometric pattern, a process pattern, or a spatial integrity constraint). The imported road-network features are polygonal and each street is made 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. Enrichment of 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 created complex SGO, a spatial integrity constraint is defined, that allows us to preserve the created group during the generalization process. Currently, three Group constraints (i.e., applied to a group of map features) are implemented: the proximity constraint (i.e., between two buildings, or between a building and a street), the alignment constraint, and the block constraint (i.e., allows buildings to remain within the same block). However, created building 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 & 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., 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 by its geometric pattern. This phase also allows the association of 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.

Contrary to the enrichment of building SGOs, the enrichment of road-network SGOs is based upon 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 recreate the road network. Based upon 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). What’s more, the automatically allotted parameters can be changed interactively if necessary.

Figure 10 illustrates the interface of the developed SGO’s Creation Prototype. 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 which will be described in the next section. The UML sequence diagram of the enrichment of database is illustrated on the figure 11. The creation of the SGOs used to test this prototype took about an hour. This processing time obviously depends on the complexity of the map’s objects, the ability of the human operator and his familiarity with the tool. In addition, many operations carried out during the interactive SGO creation can be automated and certain are already in the course of automation. Once this automation is done, the operator’s intervention can be reduced to a minimum (e.g. validation of the result).

Figure 10. User interface of the SGO creation prototype. 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.

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

4.2. On-the-fly Map Generalization System

As with the previous one, this prototype is developed in Java. Its visualization tool is based upon the GeoTools library. The multi-agent module for this prototype is developed using the JADE environment (JADE 2006). JADE (Java Agent Development Framework) is a software framework implemented in Java. It simplifies the implementation of multi-agent systems through a middleware implementing the FIPA (Foundation of Intelligent Physical Agents) specifications (FIFA 2006). This prototype supports on-the-fly production of maps at arbitrary scales. As an input, the prototype uses the SGOs that were 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’s 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 the specification, the system checks the compatibility of each value and emits a visual signal (i.e., color change) in the case of an incompatibility. The multi-agent module is launched using this administrative 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 the 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., this agent intercepts the user’s requests and extracts the needed parameters, such as the required map’s 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**, and the User Interface that includes the navigation tools. The **Drawing Agent** is responsible for drawing the generalized data.

- The **Map Generalization Sub-Module** is in charge of the map generalization. This sub-module contains several SGO agents which communicate with each other in order to generalize the selected data. Generalized data are sent to the **System Agent** that transfers it to the **Drawing Agent**. Figure 12 illustrates the interface of the developed on-the-fly multi-agent map generalization prototype.

Contrary to the approaches using a set of predefined scales (e.g. in google map), in our approach, each map object or group of map objects is generalized 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. required minimum distance between map objects in order to avoid map symbol superposition). According to the characteristics of each object and computed generalization parameters, the system determines how this object can be generalized (using geometric pattern or generalization algorithms, which algorithms and parameters to use). All this 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 parameter to specify, given that all the parameters were specified by the system administrator. The system allows the quasi-instantaneous generalization of data. For example, the average time necessary to generalize a hundred map features is about 300 milliseconds. In terms of response time, this is compatible with the requirement for on-the-fly map generalization. The prototype was tested using a Pentium 4 CPU 2.8 GHz with 512 megabytes of RAM (Random Access Memory) under Microsoft Windows XP family edition and Service Pack 2. The used number of buildings is representative of the number of buildings seen on a computer display at large scale.

**Figure 12.** Interface of the administration (A) and map generation modules of the on-the-fly, multi-agent map generalization prototype, and different map scales generated on the fly according to the map extend selected by the user (C, D).
5. Conclusion and discussion

In this paper, we presented the SGO concept which integrates geometric patterns, process patterns, spatial integrity constraints, and behaviour patterns. The existence of two generalization methods, namely the use of geometric patterns and generalization algorithms in order to generalize cartographic features or geometric patterns provide a good flexibility of the generalization process. This approach is also based upon a process of enrichment that makes explicit, information which 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, two prototypes were developed. The first is needed to create an SGO database, while the second, using multi-agent technology, allows the creation of on-the-fly maps at arbitrary scales.

In addition to facilitating the generalization process, SGOs also speed up the whole process and allow on-the-fly map generalization. This significant reduction in processing time is due on 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 that are needed in order to adjust and simplify the geometric pattern.

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

In order to personalize a map required by a user during a generalization process, several factors must be taken into account (e.g. the map scale, the 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 in 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 allowed to roads forming the route requested by the user for travelling. This allows for example to avoid that a road disappears suddenly and a user’s destination becomes inaccessible. In the current prototype, only the mechanism which allows capturing the users’ needs (like the scale of visualization and the width of the used symbols) has been implemented. But, in the future, we consider using ontologies to model the user's needs. For example, for each type of map, ontology can be created. Thus, this ontology will allow taking into account the specific constraints related to the
creation of each type of map (e.g. for a touristic 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 themselves their needs using a form transformed automatically into ontology.

Thanks to the geometric patterns, the use of SGOs can facilitate data transfer through the Internet network. Indeed, in a client/server architecture, geometric patterns can be stored on the client side (as the use of fonts in a word processing application). Thus, instead of transferring a generalized cartographic feature from the server to the client side, only parameters (feature’s position, size, orientation) allowing its replacement by its geometric pattern can be transferred. Using these parameters, the geometric pattern can be adjusted on the client side. The reduction of the traffic flow through the network, especially for on-line cartography applications, is of great importance given that the data transfer rate is known as being one of the main limitations for such applications (Bertolotto 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 re-used several times, leading to significant overall savings in time. 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, Rainsford and Mackaness (2002) proposed a method for the automatic recognition of alphanumeric templates 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 to store it in an SGO database by coupling the developed system with an existing generalization system.

6. Acknowledgements

Anonymized for review.

7. References


sur l'Intelligence Artificielle Distribuée et les Systèmes Multi-agents, AFCET&AFIA, Chambéry. 163-176.


Wonnacott L. 2000. The speed of business: If your pages are slow, your customers will go. Available at http://www.infoworld.com/articles/op/xml/00/09/11/000911opsavvy.html.
