DATA INTEGRATION AND GENERALIZATION FOR SDI IN A GRID COMPUTING FRAMEWORK

R. Guercke, C. Brenner and M. Sester

Institute of Cartography and Geoinformatics, Leibniz Universität Hannover
Appelstraße 9a, 30167 Hannover, Germany.
E-mail: richard.guercke@ikg.uni-hannover.de

KEY WORDS: GIS, Generalization , Three-dimensional, Model, City, Multiresolution

ABSTRACT:

So far, most spatial data infrastructures (SDIs) provide services for the generation of fixed-scale maps only. In a project funded by the German Ministry of Education and Science, the potential of grid computing for the deployment of geographic data processing services beyond this standard case is investigated. One of the services to be developed in this context is the generalization of 3D building models. This problem is important for almost all applications that are concerned with city models (e.g. environmental analysis or disaster management) because data sets are often quite large in this context and their size has to be reduced without losing relevant information. In order to deal with the complexity of the task, it is divided into three steps: feature extraction, building of a generalization tree and processing of a specific generalization request. Grammars are used in the description of all these steps to ensure the high degree of flexibility that is needed for a generic generalization framework; the maximum distinguishable feature size is the control parameter that serves as a link between the different components and drives the generalization process. In the paper, an architecture is presented for a system that is based on these approaches. Because the system is going to be deployed as a service within a computing grid, parallelization is an important issue to make optimal use of the computing power offered by the grid.

1 INTRODUCTION

Because of the distributed nature of the underlying data, many applications in geographic data processing are very suitable for parallelization. Therefore, a grid computing-based approach promises significant gains in performance in the deployment of such applications as services in a spatial data infrastructure (SDI). Beyond their standard task of providing fixed-scale maps, SDIs can offer more complex processing services with the help of grid technology. To investigate the potential of grid computing in SDIs, the German Ministry of Education and Science funds the "GDI-Grid" project.

As a part of this project, a service for the generalization of 3D building models is implemented. This problem has drawn the attention of researchers in recent years because it is important in fields like visualization of city models, environmental analyses and disaster management where there are large data sets that have to be reduced in size before they can be processed. Two scenarios from the last two of these fields are covered in the project: the first one is concerned with noise, the second one with flooding simulation.

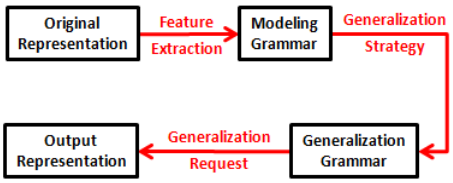

Figure 1: Workflow of the generalization service

In this paper, an architecture for this service is presented. The first section gives an introduction to the problem and an analysis of the difficulties that occurred in preceding approaches. In order to cope with these difficulties, the problem is split up into stages with intermediate representations as illustrated in figure 1:

- **Feature extraction:** A semantically enriched representation – an instance of the modeling grammar – is extracted from the input data set(s). This process is delegated to a separate task.
- In the second step, a **generalization strategy** is used to transform the model into a generative derivation tree. This tree is represented by a sequence of generalization (or rather refinement) rules from the generalization grammar.
- Finally, the desired output representation is generated by executing a **generalization request**. This request can specify, for example, individual resolutions for special features but it may also introduce local changes in the generalization strategy.

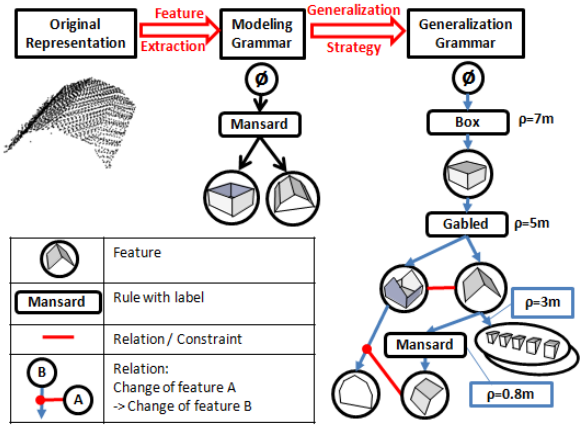

Figure 2: Illustration of the generalization workflow

Figure 2 illustrates the first steps: In the generalization strategy, the mansard roof is modeled as a gabled roof at more coarse resolutions. The relation between roof and body of the house makes sure that the split of the roof surfaces is propagated to the gable walls.

There are five possibilities in this approach that can be used to develop application-specific generalization schemes: The two intermediate representations and the three transformations shown in figure 1 can be customized to fit the user's demands. The intermediate representations and the transformations can be stored in parts or as a whole to be used in other contexts (reusability).

2 3D BUILDING MODELING AND GENERALIZATION

Generalization is about reducing the amount of data to be processed without losing relevant information. The problem, however, is that the relevance of a feature can vary significantly for different applications. Additionally, different features like walls and roofs require different strategies for the generalization of their shape.

So far, most proposals to to deal with the problem of 3D generalization like (Glander and Döllner, 2007) and (Rau et al., 2006) work well on special kinds of building models and limited ranges of resolution only. There are some more general approaches as (Thiemann, 2003) but so far, most of them implicitly introduce limitations to the features that can be modeled and some are quite demanding in computation time. Additionally, the introduction of application-specific generalization strategies is not supported in most of them.

Using grammars for the definition of the processes in all stages offers a maximum of flexibility: Only the most basic concepts – the rule interpreter, geometry, and parallelization modules – are realized in the core modules while the rest is described in grammars. Each of the grammars for the different stages has a layered structure: The service offers a set of standard rules and features and the user can add or replace rules and features to make adjustments for the application at hand. In more complex scenarios, there may be several different layers for domains, applications and special tasks. A basic set of rules can be found in (Sester and Klein, 1999).

The term **feature** refers to a semantic entity associated with a geometric shape in this paper. The most important semantic properties of a feature are its type and associated metadata objects. In the modeling and generalization grammars, the symbols are features. More details on this subject can be found in section 3.2.

In the generalization grammar, every rule is assigned a **generalization radius** ϱ. After the application of the rule, the geometry of the rule's left hand feature is supposed to be completely contained in a buffer of ϱ around the corresponding object in the real world and vice versa (Hausdorff distance): ϱ is the radius of uncertainty around the feature that remains after the application of the rule.

A representation with a maximum feature size of R can then be derived by applying all rules with $\varrho \geq R$. This concept is similar to the one introduced in (Brenner and Sester, 2005) for 2D generalization.

One of the most important reasons why the generalization of 3D building models is such a complex problem is that in order to get proper results, a semantically rich representation of the data is needed. Most models that are available at the moment are, however, not given in such a representation. Therefore, the problem of generalization implicitly contains the task of feature recognition which is generally considered to be a very hard problem. For this reason, that problem is covered in a separate service with an independent set of rules. Approaches to solve this problem with grammars can be found in (Ripperda and Brenner, 2006) (for facade reconstruction) and in (Dörschlag et al., 2007). Data sets in the CityGML format introduced in (Kolbe et al., 2005) offer semantic information but they have to be preprocessed as well to get a generative model for generalization.

3 A SEMANTICALLY ENHANCED SHAPE GRAMMAR

3.1 Geometry

The basis of the system is the underlying geometry model. In the implementation, this model is a polygon mesh that supports basic topological operations. In order to enhance its efficiency, topological properties except the identity of vertices are not tested automatically: The computation of topological relations like intersections has to be triggered explicitly. An additional advantage of this design is that it supports the definition of transactions: a valid state can be transformed into another valid state through a set of invalid ones.

Additionally, the definition of "invalid" states depends greatly on the application. Non-manifold meshes are, for example, not acceptable in applications that are concerned with solid shapes only (like in volume calculations) while they are perfectly valid in other scenarios like visualization or the calculation of occlusions in the planning of mobile antenna locations. A result of this observation is that the data structure must be tolerant with respect to geometric and topological properties.

A geometry model that supports all CSG modeling operations has to be able to compute all Boolean operators on the polygon mesh. Because the implementation of these operations is a tedious task (especially for non-manifold meshes), the first versions of the system will probably support only a limited range of operations – the calculation of intersections, for example, is left to the application.

3.2 Features

The modeling strategy is similar to the *CGA shape* architecture introduced in (Müller et al., 2006). The symbols of the grammar are features (shapes in CGA). Beyond the information attached to shapes in CGA (geometry and a label), the features contain additional semantic information: Each feature is associated to a metadata object that stores application-specific properties and its generalization interval ϱ in the generalization grammar.

Additionally, features have types. These types are used in different ways: First, the type of metadata objects associated with a feature can be chosen according to its type (addresses are, for example, assigned to houses). Another useful effect of these types is that a class system with the well-known useful concepts of inheritance, composition etc. can be introduced.

This class system includes encapsulation: A feature's properties (especially its geometry) can only be manipulated through the methods offered by the feature. This way, every feature has control over all possible changes that may happen to it.

Figure 3 shows a simple rule and a part of the feature class hierarchy. It states that a BuildingPart feature consists of a body and a roof. The types Body and Roof are abstract; concrete types may be the Gabled roof or the general polygonal body. Note that a gabled roof on a general polygonal body causes an inconsistency because the gabled roof's initialization (constructor) needs the methods from the rectangular body that give access to the parallel side faces.

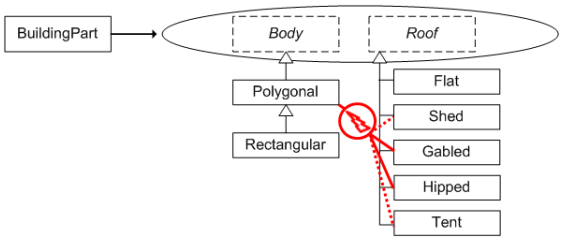

Figure 3: A rule template with inheritance and conflicts

As in (Müller et al., 2006), arbitrarily oriented bounding boxes are stored for the features. These bounding boxes define a local coordinate system: Two vectors in the base plane and the normal vector describe its rotation (and perhaps scaling factors); an offset vector defines the origin.

The bounds can be used to treat the feature hierarchy as a kind of R-tree: If a feature is modeled as a child of another feature, then its bounds are first calculated in its own system of orientation. The resulting bounding box is then added to the bounding box of its parent feature (in the parent's orientation). If the parent's bounds had to be extended in this process, the parent's parent has to be notified – up to the root feature if necessary. The advantage of this system is that – if the modeling rules are adequate – a natural and usually quite efficient search structure is given on the data.

Another interesting point is that this concept offers a possible way of guiding a feature extraction algorithm: One might try to experiment with quality measurements taken from the R-tree data structure – like minimal overlap or minimal additional space covered – when the decision has to be made how features should be grouped (for example if a chimney on a roof top should be treated as a child feature of one of the roof surfaces or as an independent feature on the roof).

In the first version of the service, the feature and rule types are going to be implemented as classes in the source code. Derivations can be added by subclassing from the feature and rule types defined in the default set. In later versions, a system with dynamic typification of rules and features may be added.

3.3 Rules

The rules define transitions from one configuration of the derivation to another. Such a configurationis represented by the tree of features and subfeatures derived up to the point at which a rule is executed.

All rules may have only one feature on their left-hand side. This restriction is introduced to control in which parts of the model the rule can have effects: A rule can only modify (and generate) child features of its left-hand feature (including the feature itself). A rule that affects more than one feature is usually associated with the first common ancestor of all features involved.

If a feature is linked to another one by a relation, a change to this feature can make changes to the related features necessary. The generalization strategy has to make sure that these side-effects do not lead to changes outside the rule's radius of influence in the generalization grammar.

Rules can be interpreted as executable leaves in the feature tree inserted as children of their left-hand features. When a rule is applied, it is removed from the tree and its right-hand side is executed. In the evaluation of the right-hand side, new features and rules can be added to the configuration.

The features store references to the rules that depend on them; in the generalization grammar, the generalization radius of the rules is stored together with the rule. This way, all active rules – the rules that can be executed in the current configuration – are available. In the generalization grammar, these references to the active rules are also collected in a priority queue ordered by the generalization radius.

If the generalization request states nothing else, the rules are simply executed in the order given by this priority queue until the required level of detail is reached. The request can be customized in a generalization request by skipping rules or by executing rules before their regular turn has come.

Because rules in the generalization grammar can store references to the rules in the modeling grammar from which they were created, the generalization request can even introduce changes to the generalization strategy by replacing rules according to the original rules from the model.

Rules can also add new rules to the current configuration. This is especially useful in the generalization process: It offers the possibility of procedural generalization.

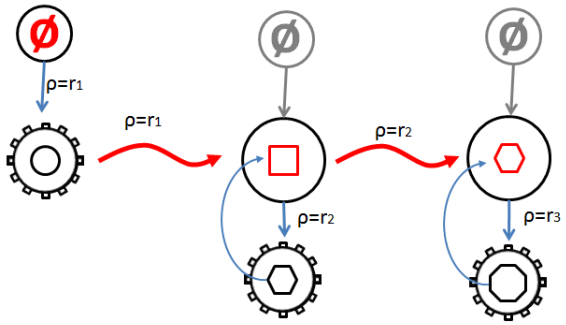

Figure 4: Procedural model of a circle

Figure 4 shows an example: the modeling of a circle for a polygon-based client through regular inner (or outer) polygons. Whenever the maximum tolerable error drops below the distance between circle and polygon, the shape of the current feature is replaced by a new polygon with more sides and a new rule is added that prepares the next step if the resolution drops even further. At a resolution of r_1, the generalization system detects that a circle has to be modeled. As only polygons are returned, the feature is generated with a rectangle as its associated geometry. Additionally, a rule is created that at a resolution of r_2, the rectangle has to be replaced by a hexagon and so on. In the drawing, rules are marked by the gear symbol while features are represented as circles.

3.4 The Modeling and Generalization Grammars

There is an infinite number of ways in which a given feature may be generalized; the choice of the most appropriate approach depends on the application. The generalization strategy defines the transformation of a semantically annotated model into a generalization tree.

The modeling language is defined in such a way that as much semantic information as possible is present in the model and that there are sufficient degrees of freedom to define custom generalization strategies.

Consider, for example, a mansard roof. In the modeling grammar, such a roof may be introduced directly as an addition to the

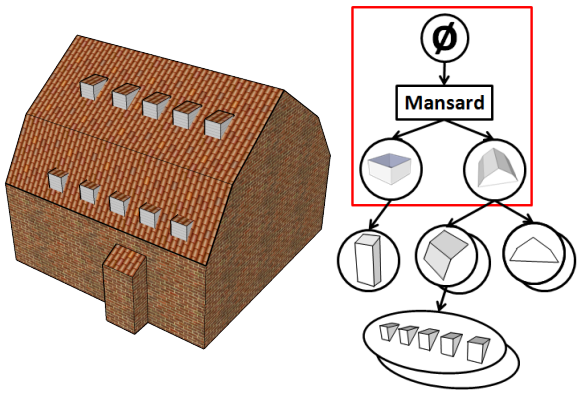

Figure 5: Modeling tree for a mansard-roof house

building's torso as illustrated in figure 5. A generalization strategy could, for example, transform this first derivation (indicated by the red box in the modeling tree) into a generalization tree like the one shown in figure 6.

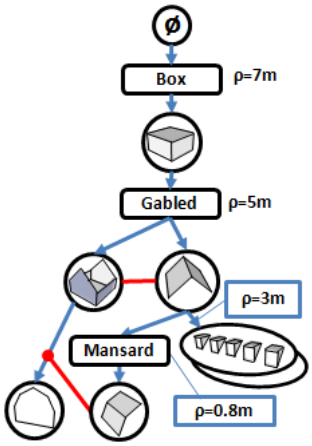

Figure 6: Part of a modeling tree transformed for generalization

The dormers are added in a further step, for example at a radius of 3m. As figure 6 shows, the mansard roof is modeled as a gabled roof at that resolution, so the dormers' base surfaces will have to be adjusted to fit on the roof's side surface. When the roof surface is split, the dormers will assume their original shape.

In order to allow such generalization strategies, it must be possible to change the basic parameters of a feature and sometimes even its type in the generalization process: The lower roof surface of a more pronounced mansard roof may, for example, be modeled as a wall at lower resolutions. In this case, the dormers on this surface would first appear as windows.

The basic tasks of a generalization strategy are the setting of the resolution parameters and the calculation of the bounding boxes for the features.

4 RELATIONS AND SPECIAL FEATURES

4.1 Relations and Constraints

Relations are used to model situations where properties of different objects depend on each other. The features whose relations are modeled will be called the **clients** of that relation. The most common relations are metadata-based relations, patterns, and geometric relations like intersections or alignments.

Some relations can be associated with special features. The intersection of two (or more) features, for example, can be modeled by a special feature that is associated with the intersection of the geometry of its client features.

Constraints are conditions that have to be satisfied in a valid instance of a model. Many of them can be modeled by relations. The The most basic topological constraint states that features have to fit without holes (continuously). This means that parts of the features have to be identical like the sides of the walls in figure 7. This identification of feature parts is indicated by the "=" sign in the constraint link.

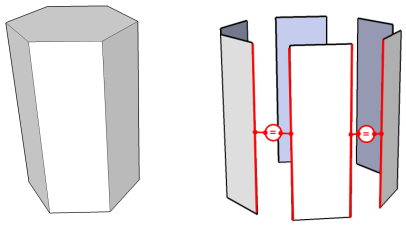

Figure 7: Constraints in a polygonal arrangement of walls

Such a constraint also occurs in figure 6: The roof has to fit on the body of the house without holes between them. When the gabled roof is split to form the mansard roof at a resolution of 0.8 meters in the example, the relation requests a deformation of the gable wall surfaces that makes sure that the roof still fits.

Another group of constraints is concerned with **alignments**: Parts of a feature may be defined to be aligned with directions defined in other features. These constraints occur especially in the context of intersections (see chapter 4.3). The basic alignments are **parallel** and **perpendicular** alignment; for the parallel alignment, a specific, minimal or maximal distance may be required.

Many constraints can be realized implicitly (for example in the constructors of features) but in the generalization process, features may change their shape or even their type. In such cases, the relational features can serve as safeguards and moderators.

4.2 Patterns and Collections

One of the most common relations between features is their arrangement in a **pattern**, for example along a line or in a matrix. In such a case, the features can be accumulated in **collection** features.

The building in figure 8 shows a case of nested collections: The linear array of five dormers is present three times on the building's roof.

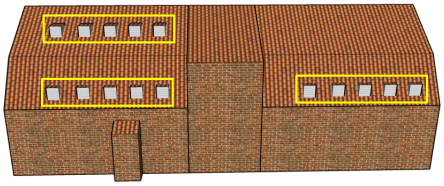

Figure 8: Repetition of dormers

Using collections is the precondition for the generalization techniques of typification – in which n objects of the same type are represented by $m < n$ objects of the same type – and aggregation where n objects of the same type are merged into one.

The example shows another important point about collection features: If the linear sets of five dormers in the yellow boxes are modeled as five different dormers each, then they will appear much later in a generalization process because the set of dormers has a width of several meters while the maximum extend of an individual dormer is probably not much more than one meter in any direction.

The extraction of collection features from unordered data is a complex problem but necessary to get satisfying generalizations, especially in visualization. In (Heinzle and Anders, 2007), for example, an approach is presented to find patterns in road networks.

4.3 Intersections

The computation of intersections of polygon meshes is a complex geometric problem. Therefore, they are only computed on demand. In the modeling process, there are two general cases where intersections are important: For consistency checks and to localize features that "live" on an intersection of other features (like a turret at the corner of two adjacent walls).

For consistency checks, intersections can be modeled as simple geometric constraints that the geometry of a given feature should not be intersected by another feature.

Another case in which intersections are important is the situation that a feature "resides" on the intersection of other features. In this case, the intersection has to be known in order to place the new feature correctly.

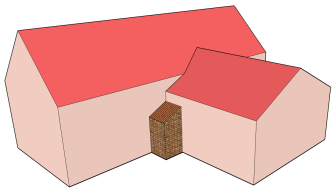

Figure 9: Feature defined on an intersection

Figure 9 illustrates this situation: The bay is aligned to the corner formed by the intersection of the walls in the two building sections.

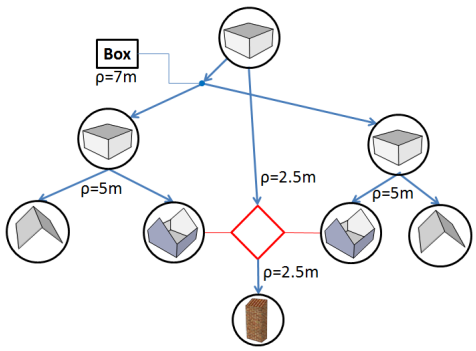

Figure 10: Model of a feature defined on an intersection

Figure 10 shows a derivation tree for a generalization of the situation in figure 9: The intersection is ignored (this is, of course, only possible if no structures inside the building are relevant and the inner surfaces do not disturb the calculations) until its geometry (symbolized by the red rhombus in the feature tree) is needed to place the bay at a generalization radius of 2.5 meters. If one of the walls is shifted during the generalization process, the bay has to be moved along with it.

If there are no inconsistencies in the model that arise from intersecting features, these intersections do not have to be modeled explicitly. This reduces the complexity of the modeling process.

In the generalization process, however, intersections and other side-effects of displacements and simplifications of features are a constant nuisance because they can lead to inconsistencies that are difficult to detect: In figure 11, features A and B intersect because of the deformation of feature A in the generalization.

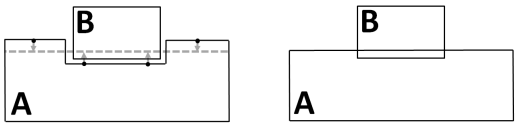

Figure 11: Intersection during generalization

In order to deal with these problems, intersections of objects have to be established if the generalization process involves the deformation or displacement of a feature. The bounding boxes help to reduce the number of features that may intersect with the currently modified feature.

It is up to the generalization strategy how it deals with these intersections. Some possible solutions are to accept the intersection without any changes, to merge A and B into one feature, or to shift A or B or both in such a way that there is no intersection.

4.4 Parallelization

The issue of parallelization is addressed by the introduction of special split rules. These rules are connected to methods in the source code that trigger a tiling of the geometry model and dispatch the execution of all rules applied to features within a tile to a separate process for each tile.

In the splitting process, the generalization interval for each rule is a great help: It defines the (3D) area in which the different tiles can overlap. If needed, special moderator features can be defined in the area where the tiles overlap.

These features should be defined in such a way that as few references from the intersection features to their clients are needed in the generalization process. This can be done by copying all rules that affect the moderator feature to all processes that work with it.

5 FEATURE DETECTION

Feature detection is a challenging problem in its own right. The focus of the implementation of the service will therefore be on the generalization step itself and on the development of models that are convenient for user-defined generalization.

The feature detection process can cover the whole range of semantic expressiveness: from raw laser scanner data via purely geometric shape representations up to highly expressive CityGML models.

The modeling grammar defines the valid representations for the input of the generalization service. For this reason, the feature extraction may consist of a transformation from one semantically annotated representation to another.

6 SYSTEM ARCHITECTURE

Figure 12 shows a diagram of the components of the system:

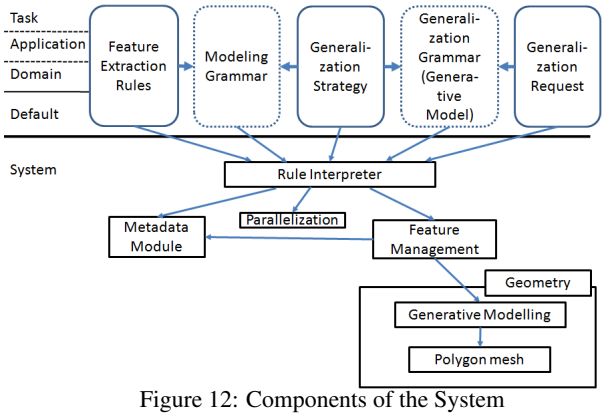

Figure 12: Components of the System

The central part of the system is the rule interpreter. The complexity and flexibility of the rule interpreter define how much effort will be needed for the introduction of custom feature types and rules.

The rule interpreter needs access to the whole underlying data model. This model consists of a geometry engine and modules for the handling of metadata and parallelization.

The feature management engine is responsible for the administration of the current configuration and controls access to the associated metadata objects.

The geometry module is needed to execute the parts of the rules that deal with the manipulation of geometry: The generative modeling component offers utility features like special shapes and higher-level manipulation operators that are not provided by the basic polygon model itself.

The sets of transformation rules responsible for **feature extraction**, specification of **generalization strategies** and the execution of **generalization requests** define the transitions between the original representation, the intermediate formats and the final output.

The **modeling grammar** describes a semantically enhanced representation of the features; the generalization strategy prepares the model for generalization by producing a derivation tree – an instance of the **generalization grammar**. A generalization request specifies how the final output is generated as the product of a traversal of this tree.

7 CONCLUSIONS AND FUTURE WORK

In this paper, a strategy has been sketched for the development of a system for the generalization of 3D building models. To avoid the difficulties of earlier approaches, the problem is split into the stages of feature extraction, application of a generalization strategy and execution of a generalization request.

A feature grammar with built-in resolution management and aspects of the R-tree data structure is introduced as a basis for the intermediate semantically enhanced representations of the data.

This paper gives an overview of the aspects that are going to be included in the generalization service that is developed in the course of the GDI grid project. So far, there is a model for the architecture of the service as well as the first prototype of the geometry module.

In the next step, a framework for the implementation of the grammars is going to be built. Using this framework, a set of default rules for modeling and generalization are developed.

ACKNOWLEDGMENTS

This work was funded by the German Ministry of Education and Science (BMBF) in the context of the GDI Grid project.

REFERENCES

Brenner, C. and Sester, M., 2005. Continuous generalization for small mobile displays. In: Agouris and Croitoru (eds), Next Generation Geospatial Information, ISPRS Book Series, Taylor & Francis Group, London, pp. 33–41.

Dörschlag, D., Gröger, G. and Plümer, L., 2007. Semantically enhanced prototypes for building reconstruction. In: U. Stilla, H. Mayer, F. Rottensteiner, C. Heipke and S. Hinz (eds), International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences, Vol. 36, Part 3/W49A.

Glander, T. and Döllner, J., 2007. Cell-based generalization of 3D building groups with outlier management. In: GIS '07: Proceedings of the 15th annual ACM international symposium on Advances in geographic information systems, ACM, New York, NY, USA, pp. 1–4.

Heinzle, F. and Anders, K.-H., 2007. Characterising space via pattern recognition techniques: Identifying patterns in road networks. In: W. Mackaness, A. Ruas and T. Sarjakoski (eds), Generalisation of Geographic Information: Cartographic Modelling and Applications, International Cartographic Association, Elsevier Ltd., pp. 233–253.

Kolbe, T. H., Gröger, G. and Plümer, L., 2005. CityGML: Interoperable access to 3D city models. In: First International Symposium on Geo-Information for Disaster Management GI4DM.

Müller, P., Wonka, P., Haegler, S., Ulmer, A. and Gool, L. V., 2006. Procedural modeling of buildings. In: SIGGRAPH '06: ACM SIGGRAPH 2006 Papers, ACM, New York, NY, USA, pp. 614–623.

Rau, J., Chen, L., Tsai, F., Hsiao, K. and Hsu, W., 2006. LoD generation for 3D polyhedral building models. In: Advances in Image and Video Technology, Springer Verlag, pp. 44–53.

Ripperda, N. and Brenner, C., 2006. Reconstruction of façade structures using a formal grammar and RjMCMC. In: K. Franke, K.-R. Müller, B. Nickolay and R. Schäfer (eds), Pattern Recognition, Proceedings of the 28th DAGM Symposium, Berlin, pp. 750–759.

Sester, M. and Klein, A., 1999. Rule based generalization of buildings for 3D-visualization. In: Proceedings of the 19th International Cartographic Conference of the ICA.

Thiemann, F., 2003. 3D-Gebäude-Generalisierung. In: Visualisierung und Erschließung von Geodaten, Kartographische Schriften, Vol. 7, pp. 185–192.