GENERALIZATION OF 3D CITY MODELS AS A SERVICE

R. Guercke, C. Brenner, M. Sester

Institute of Cartography and Geoinformatics, Leibniz University Hanover, Germany
Appelstr. 9A, 30167 Hannover
{richard.guercke, claus.brenner, monika.sester}@ikg.uni-hannover.de

KEY WORDS: Generalization, City Models, 3D, Service, Semantics, Modularization

ABSTRACT:

In recent years, 3D models have been created of many cities around the world. Most of these models are, however, only used for visualization in web frontends or navigation systems. With the rapid progress in the development of sensor systems and algorithms for the interpretation of their measurements, an increasing number of more and more detailed models is going to be available. Such detailed models can, however, not be used directly in many applications because the data sets are too large or because there is information in the model that cannot be handled by the application. The purpose of generalization is therefore to reduce the size and semantic complexity of a model to a level that can be handled by the application without losing relevant information. In order to implement such a highly application-dependent task as a generic service, it is divided into atomic modules. Generalization requires information about the objects and their context they are embedded in. Thus the features present in the data have to be revealed first, before generalization can be applied. Thus, especially with a stronger separation of generalization and feature extraction steps, such a modularization offers great potential for the reusability of components and flexibility in the integration of application-specific components.

1. INTRODUCTION

3D city models are used in a growing number of domains. While the purpose of these models was first mostly visualization, the scope of applications has widened, and city models form the basis for an increasing range of calculations for simulation and analysis.

In order to avoid having to start a complete new data acquisition for each new application, it is sensible to try to reuse models created in earlier projects. Especially with the more and more detailed models that are going to be created in the near future, a growing need is going to arise for the generalization of such models because many application scenarios – especially if complex computations are necessary – cannot handle the amount and semantic complexity of the data in the model. Many applications will, for example, have problems dealing with building models in which walls are modeled by interior and exterior surfaces or with decorations modeled in greater detail. These are not only problems of size but also of semantic complexity since even if the data set was small enough for the calculation, it would not work because of the presence of unexpected features in the model.

The purpose of generalization is therefore to reduce the size and semantic complexity of a model without losing information that is relevant for the application.

The concept of relevance being inherently a semantic one, semantic criteria play an important part in the problem of generalization. This dependence on detailed semantic information is the cause for the mixture between feature extraction (structure recognition) and generalization that can be identified in many generalization approaches. In this scope, the term *semantics* refers to the type and application-specific data fields associated with a feature.

One of the main challenges in the development of a generic service for generalization is that the concept of relevance and therefore the whole process of generalization strongly depends on the application. In order to meet this challenge, the our concept of the service is implemented in the form of a framework with different options for customization. In order to spare the users the trouble of having to develop their own models and generalization approaches for standard scenarios, default models and generalization procedures can be defined. In the course of the development of the framework, a basic model with basic generalization procedures is created. Additionally, patterns features (like arrays) and generic operators on these patterns (like typification) are investigated and implemented.

In a digital model, resolution is not a physical restriction. A very basic (but nonetheless quite powerful) option for customization is the assignment of non-uniform resolutions to different parts of the model based on semantic and spatial criteria. This includes the cases that parts of the model are supposed to be kept or left out. With the support of a frontend with basic GIS functionality, even users with little experience in generalization and programming can formulate queries like "Give all features within 500m of the river Elbe at a resolution of 1.5m, the bed of the river and all features in it (like bridges etc.) at 1m, and everything else at 2.5m". Additionally, application developers can provide prepared requests for recurring scenarios. In the easiest (default) case, the user chooses a uniform resolution for the whole model.

Within the framework, basic feature types with corresponding simplification methods are provided as default solutions. If nothing different is chosen by the user, the default methods are used with default parameters. This way, the users or application developers can focus on those aspects of the generalization process that are important for their purpose.

The introduction of the role of the application developer is necessary because neither can the developer of a generic framework foresee the needs of all conceivable applications, nor should all users be faced with the problem of having to implement the feature types and adaptors that may be needed to customize the framework to meet the requirements of their application.

Due to the modular concept of the framework, application developers can use the standard procedures where they are sufficient and "plug in" application-specific features and generalization methods where they are needed. The addition of new data fields to existing feature types does not require the definition of new feature types; if a new feature type with special properties is needed, it can be derived from existing ones through inheritance. Using the different options the framework offers for customization, the application developers can offer special algorithms for parts of the generalization process and complete generalization scenarios for special purposes.

In this paper, we report on concepts for such a generic service for the generalization of 3D city models. It also gives an overview of the status of the prototype for a generic generalization service that we develop as a part of the GDI grid project that is concerned with the application of grid technology to spatial data infrastructures as a part of the German grid computing initiative (Groeper et al., 2009).

2. RELATED WORK

The approach of using hierarchical models for 3D city model generalization has been introduced in (Lal, 2005) in his distinction between micro, meso and macro models for generalization. There are, however, only these three fixed levels in his hierarchy; it is therefore not possible to extend the model towards larger or more fine-grained structures. He also stresses the necessity of a stronger separation of the processes of feature extraction and generalization. The focus of his work is, however, on feature extraction and the specific generalization operation of aggregation.

H. Fan et al. (2009) introduce an approach to extract the exterior shells of building models that contain interior and exterior surfaces for walls and roofs – with the generalization step consisting of replacing the original geometry by the exterior shell. Additionally, different strategies for the generalization of (regular arrays of) windows are evaluated.

M. Kada (2007) uses the wall surfaces of a building complex to detect structural parts (cells) of an ensemble of building components. He introduces parametric primitives for roof forms. Using the different roof primitives, regular patterns of roofs can be detected in order to apply the generalization operator of typification. For the general structure of the building complexes, the selection operator is used: If a cell is too small to be retained after the generalization process, it is removed from the model. The generalization approach works on geometric models and consists to a great part of a feature extraction component. It is limited to building models that consist of wall and roof surfaces.

Thiemann and Sester (2004) also present an approach towards the generalization of 3D city models: The roof and wall planes in the model are used to derive a CSG representation of the building. The generalization step is a selection that is employed by removing those primitives from the representation that are too small for the given resolution.

Representing and constructing 3D city models using formal grammars has been proposed by (Wonka et al. 2003); similarly such grammars can also be used for the reconstruction process as showed by (Reznik and Mayer 2008).

There are different approaches towards making cartographic generalization processes available in a service-oriented framework. In the work of Neun et al. (2008), for example, the potential need for additional semantic information for different generalization services is stressed. The semantic enrichment has to be accomplished by feature extraction or pattern recognition methods (Heinzle and Anders 2007). This need is even greater in the 3D case because geometrically more complex problems may arise here that can be simplified significantly with appropriate semantic information.

3. THE DEFAULT FEATURE MODEL

We propose a default feature model with a hierarchical structure in which the parent-child relation represents a consists-of or containment relation. This structure is very similar to the one underlying the CityGML data model (see Kolbe et al., 2005). The geometry of the features is, however, represented implicitly through the feature type and geometric parameters.

As in the CityGML model, different layers for different semantic contexts like buildings, transportation, and vegetation are going to be present in the default model. The focus of the first prototype is, however, on settlement objects.

Due to its hierarchical organization, the model can easily be extended by more detailed and more general structures.

Figure 1 shows the visualization of a building model created for the prototype of the generalization service. Figure 2 gives an overview of the corresponding structure of the model.

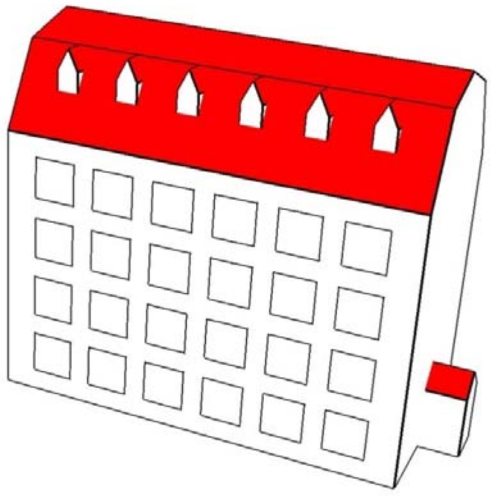

Figure 1: A Building Model

For the generalization process, it is important to distinguish between the necessary parts and optional additions of a feature. The necessary parts of a building are, for example, its body and roof: If one of them is removed in the course of the generalization, the building is usually not valid anymore.

This notation was chosen to remind of the UML notation for the aggregation and composition relations because it has a similar meaning. It does, however, work in the opposite direction: In the UML definition, an aggregation relation is a composition if the parts do not make sense without the whole while in the model the whole is not valid without the essential parts. The rhomboids are used to illustrate the distinction between necessary and additional child features. The necessary parts are modeled as attributes of the class (e.g. body and roof for a building) and labeled with the names of these attributes. As the optional additions can, in principle, be any kind of feature, they are stored in a simple list.

In the representation of the structure, the necessary parts are marked by filled, the optional additions by empty rhomboids. The template features of an abstract group feature like a regular array are, in fact, necessary parts that can appear more than once in different positions defined by the pattern.

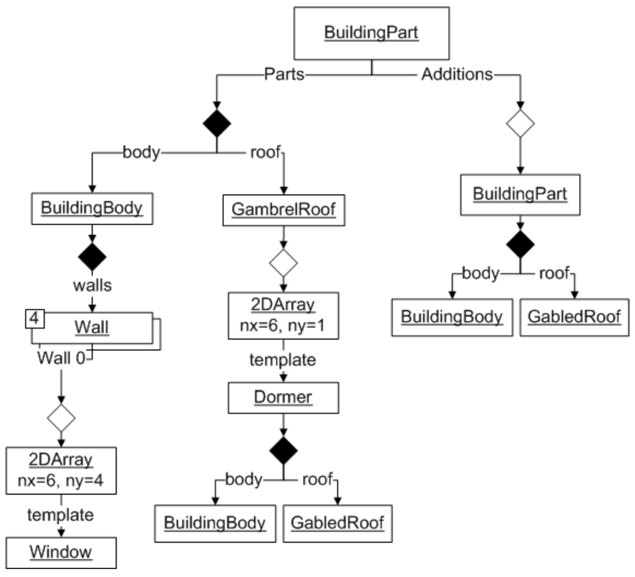

Figure 2: Structure of the model shown in Figure 1

For some applications, not all of these parts may be necessary. In such a case, special kinds of necessary parts may be removed from the list of parts for a feature. This can, however, make adjustments to simplification procedures for the feature necessary. For this reason, the set of required parts for a feature type remains fixed in the default model. Such strongly application-specific relaxations of very general constraints have to be evaluated in the post-processing step when the output data is prepared for the application.

4. BASIC WORKFLOW IN A GENERALIZATION SCENARIO

In a hierarchical model, patterns of features can be modeled as special features. For this reason, pattern recognition is treated as a special kind of feature recognition in this context.
With the generalization of 3D city models, a workflow like the one shown in Figure 3 can be identified in most approaches — often with different instances of the workflow for different situations or patterns in one generalization algorithm. This happens because in most models that are available at the moment, the level of semantic information needed for generalization is not represented explicitly. Even if highly structured models like CityGML are used, it is not necessarily explicitly modeled, that e.g. neighboring windows form a row or even a grid of windows.

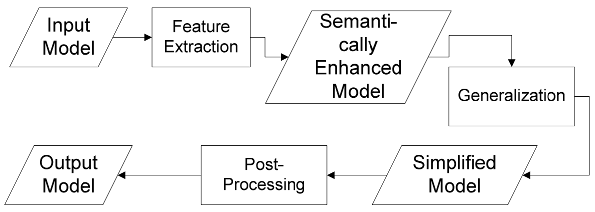

Figure 3: Basic Generalization Workflow

Unfortunately, many of the approaches introduced so far are limited to models with more or less specific properties because the feature extraction (or structure recognition) and generalization are bundled into a single process. Especially the implicit combination of feature extraction and generalization components is a problem because feature extraction is a research topic in its own right.
Splitting up the different approaches into atomic units can significantly increase the reusability of the different components. Especially a general availability of feature extraction techniques or of models enriched with more detailed structural information is going to advance the progress in the field of generalization considerably because it will help developers to use more rich object descriptions as basis for their developments of generalization operations.
Especially in the restructuring component of the generalization process described in section 5, it can be helpful or even necessary to apply feature extraction algorithms to parts of the model during the main generalization step. Instead of being an argument against the separation and modularization of the different steps, this is rather a further motivation for the modularization because existing atomic modules from earlier steps or other contexts can be used very flexibly in the generalization process.
The post-processing module encapsulates the adjustments and abstractions that are necessary to transform the output of the generalization into a model that can be handled by the application. Such a module can, of course, only be implemented with knowledge of the application. This will usually be done by an expert in the application domain – possibly in cooperation with developers of generalization modules. Once such a module is available, different users from the domain can use it to get access to all sources of information for which there are feature extraction procedures that can identify the relevant structures for the application
For a flooding scenario as reported by (Kurzbach and Pasche, 2009), for example, upright surfaces like walls cannot be handled directly. This means that the walls either have to be modeled as sloped surfaces or the houses have to be replaced by special elements representing holes in the terrain model.

5. A PROCESS MODEL FOR THE GENERALIZATION STEP

Figure 4 gives an overview of the general structure of the generalization process. Basically, it consists of a depth-first traversal of the feature hierarchy. In the course of this traversal, the generalized model is assembled.
Within this traversal, feature extraction steps may be necessary, so the basic workflow described in section 4 can appear in several instances in the course of the generalization of a model. Additionally, the process can be embedded in such a workflow as the generalization component.

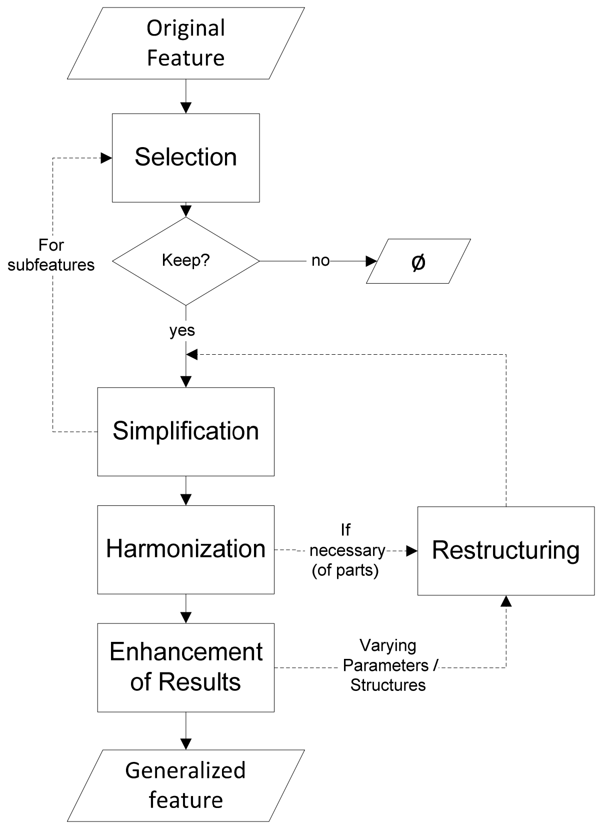

Figure 4: Process Model for Generalization

For all sub-processes in this process model, different modules can be defined to perform the respective task. The choice of the module to be used for an individual feature is made according to the specifications in the generalization request. In most cases, the standard modules provided by the framework should be sufficient – possibly with some adjustments of parameters.
The basic idea behind the introduction of standard features and procedures is to relieve the user of having to be concerned with issues they are not interested in. The user has only to specify the cases in which exceptions from the standard components occur. This way, the users can focus on the parts of the process that are relevant for their special application.

5.1 Selection

Starting with the root feature of the input model, the first step is a selection operation. If the feature does not qualify to be retained in the output model, nothing is returned. Besides a selection based on semantic criteria, the most common criterion for this decision is to test if the diameter of the feature's bounding box is greater than the target resolution assigned to the feature. Features can be labeled to be retained without having to pass the selection step.

5.2 Simplification

The term "simplification" is used in a more general sense than usual: It refers to any generalization operation that changes to shape of the feature. This can be either a feature(type)-specific operation like the replacement of gabled roof by a flat one or an abstraction of a cartographic generalization operator.
The simplification process is the point where individual generalization procedures for the rearrangement of components of the feature can be plugged in. In this step, the parts and parameters of the original feature can be rearranged or the feature can be replaced by a less complex one – a gabled roof may, for example, be replaced by a flat one. In order to deal with the sub-features of the affected child features, it is necessary to keep track of the mappings from the old to the new sub-features.
Many of the well-known standard generalization operators can be interpreted as simplification operators for higher-level features. The typification operator, for example, can be seen as a simplification method for features representing regular patterns or clusters of features: It takes a cluster of features and returns a cluster consisting of less (mostly also less complex) features. For this reason, it can be seen as a simplification procedure for a cluster of features where the "cluster of features" is an abstract type of feature. One subclass of this "cluster of features" class is, for example, the FeatureArray class that models features that are arranged in a regular grid.
In the standard case, the first step is to start the generalization process for the necessary parts of the current feature. The generalized versions of the parts are then joined to form the generalized feature. In order to use this standard procedure for a feature, it has to provide a get_parts() method that returns the necessary parts of the feature. For the standard simplification, there has to be restore_from_parts() method to reassemble the feature from its generalized parts.
After that, the different optional additions of the feature are subjected to the generalization process – including the selection step. If additions are removed (in case they do not pass the selection test), the selection step can be repeated for the current feature at this step: The bounding box could have been enlarged significantly by small features like high (but thin) chimneys or antennae on the roof of a house.

5.3 Harmonization

In the harmonization component, conflicts arising due to the independent simplification of the sub-features are resolved. Especially the conflicts between additions and the feature itself or its necessary parts are an issue here.
A simple case are, for example, dormers on a roof. If a gabled roof is simplified to a flat one, the dormers would either disappear below the roof or sit on it like small turrets. In order to avoid such a situation, either the dormers have to be removed or the roof simplification step has to be undone. The second option would require a new start of the simplification step with the simplification to a flat roof disabled. Dormers on steep roof surfaces (for example, the lower surfaces in mansard or gambrel roofs) can also be replaced by windows.
In more complex cases – especially in the context of the displacement or emphasis –, more sophisticated techniques are going to be necessary. One possible way to solve such conflicts is to introduce priorities for the different features and to restart the generalization of the feature of lower priority with the areas occupied by the higher priority features marked as forbidden areas that must not be intersected by any part of the generalized feature. One of the principal challenges in the development of the framework is the identification of generic classes of conflicts and the development of generic strategies for their resolution.

5.4 Restructuring

The restructuring component handles requests for changes to be applied for a new generalization cycle. This includes the handling of blocks for certain generalization steps or the definition of forbidden areas.

The principal function of this component is, however, to make adjustments of the input model possible. These changes can be used to resolve conflicts or to increase the quality of the result.

Figure 5 shows, for example, a façade and two possible structural decompositions of it. Even though its structure is rather 2(.5)-dimensional, the principles shown in this example can obviously be applied to three-dimensional situations as well.

In the left interpretation, the façade is modeled as a symmetric arrangement of windows with the (slightly protruding) central part being treated separately. In the right one, the whole façade is interpreted as a set of different windows arranged in a regular grid. The basic concepts of the façade model and the picture are taken from Ripperda & Brenner (2009).

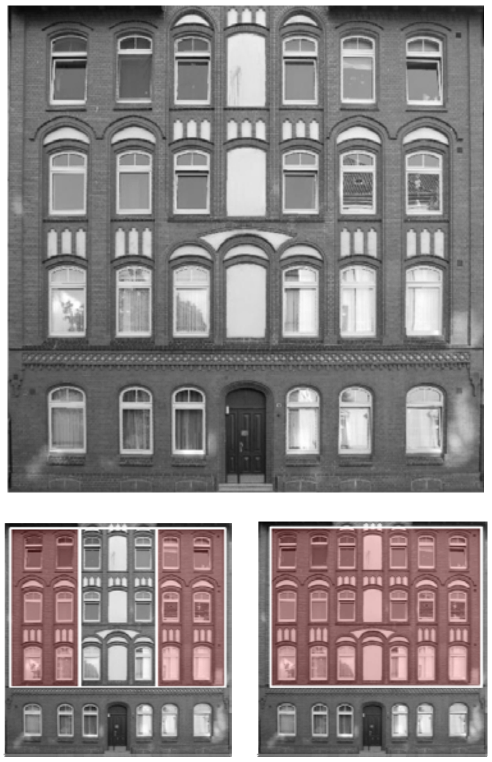
Figure 5: Two structural decompositions of a façade

Depending on the required resolution, both views are justified: If more detail is required, the first option is often more suitable for the generalization of the whole structure because the problems arising from having to model and deal with the great number of exceptions necessary in the second model would offset advantage of the simpler structure. If, however, a less detailed model is requested, the differences of the individual features may not be relevant anymore, and the second option will probably yield better results.

In the restructuring component, such different options can be generated, for example, by the application of structure recognition algorithms or through predefined possible transitions.

The harmonization component controls the process of testing different structures (generated by the restructuring component) in order to get more suitable generalized models.

5.5 Example

The upper part of the façade in Fig. 5 is assumed to be modeled according to the left of the smaller pictures: It consists of a central part and two symmetric sides – of which only one is modeled explicitly.

In the generalization of this part of the facade, the first step consists of testing if this component is supposed to be kept for the given resolution. If this answer to this question is negative, then the space occupied by this facade could made available for other features (not very likely in this context) or left blank to be filled by the default background pattern.

If the feature is decided to be kept, the facade elements covering the central and the side parts are analyzed (sent to an appropriate generalization service).

In the example, both center and sides could be modeled as arrays of windows with slightly different features in the rows (different ornaments), so a typification service for inhomogeneous array data could be used to get a generalized version of the parts. In this first run, both center and sides are considered essential parts of the façade, so we pass them to the typification service with the selection process of the typification service disabled. This service would then pass the individual window templates to an appropriate generalization service which may, for example, simplify or remove ornaments. After that, the typification service will rearrange the distribution of the (simplified) window templates according to the chosen resolution. The result of this process is that there are two new arrays of windows with – possibly fewer – less complex window templates (with less complex ornamentations, crossbar structures etc.) and possibly less instances of these templates (for example, only one column for the sides).

After having received the resulting features for the central and side parts, the generalization service for the whole structure can invoke a service (in the reconstruction component) that tests if the two features can be merged. In the case of the two arrays, such a test can evaluate if the arrays have sufficiently similar template features. At coarse resolutions, the template features (windows) will be similar because the ornaments which make up for most of the differences between the windows will be more similar of completely removed. In this case, it may be decided that the center and the sides can be treated as one array of windows. This one array can then be fed to the typification service for inhomogeneous feature arrays. The result could, for example, be an array of 3 rows with four windows each. If the difference between the sides and the central part are too big for the given resolution, the generalized versions of the center and the sides can simply be kept.

A need for harmonization can arise, for example, if the space of the side panel is not sufficient to hold even a single window. In such a case, it would have to be enhanced. This requires that the central part is supposed to give up some of its width to the sides which may lead to new problems. In our example, this case is not so critical because the center and the sides could probably be merged in such a situation.

In the enhancement step, different models for the same real-world object may be generated and compared. In the example, this could involve merging the upper and lower part of the façade (storing the differences) and testing how such changes affect the result of the generalization. The different results could then either be presented to an operator for the final decision or selected according to an application-specific evaluation function.

6. CONCLUSIONS AND OUTLOOK

In this paper, we presented the concept for a process model for the generalization of hierarchical feature models. Due to the modular structure of the generalization process, different

feature models with associated simplification procedure can be integrated in the generalization framework.
To validate our approach, a prototype of the generalization service is developed. In its current version, the selection and simplification modules have been implemented for a simple feature hierarchy.
In order to show the practicability of the approach, the next step is to model more complex composite buildings and façade structures and to investigate the problems arising in the generalization of these features.
Especially the harmonization and enhancement of results for such complex structures are crucial points.
After the default model and generalization procedures are implemented, the next step is to develop customizations for the application scenarios in the context of flooding and noise simulation that are associated with the project.

7. ACKNOWLEDGEMENT

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

8. REFERENCES

H. Fan, L. Meng, M. Jahnke, 2009. Generalization of 3D Buildings Modelled by CityGML, Advances in GIScience: Proceedings of 12th AGILE Conference on GIScience, Lecture Notes in Geoinformation and Cartography, p. 387-405, Springer, Berlin, 2009.

R. Groeper, C. Kunz, C. Grimm, 2009. Connecting OGC Web Services and the Grid using Globus Toolkit 4 and OGSA-DAI, Proceedings of the 10th IEEE / ACM International Conference on Grid Computing (Grid2009)

M. Kada, 2007. 3D Building Generalisation by Roof Simplification and Typification. In: Proceedings of the 23th International Cartographic Conference, Moscow, Russian Federation.

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

S. Kurzbach and E. Pasche, 2009 A 3D Terrain Discretization Grid Service For Hydrodynamic Modelling, 8th International Conference on Hydroinformatics 2009, Concepción, Chile

J. Lal, 2005. Recognition of 3D Settlement Structure for Generalization. PhD thesis, Technische Universität München, 2005.

M Neun, D. Burghardt, R. Weibel, 2008. Web service approaches for providing enriched data structures to generalisation operators. International Journal of Geographical Information Science, 22(1-2): 133-165.

N. Ripperda and C. Brenner, 2009. Application of a Formal Grammar to Facade Reconstruction in Semiautomatic and Automatic Environments. Proceedings of 12th AGILE Conference on GIScience, Hannover, Germany, 2009.

F. Thiemann and M. Sester, 2004. Segmentation of Buildings for 3D-Generalisation. Proceedings of the ICA Workshop on generalisation and multiple representation, Leicester, UK.

Heinzle, F., and K. H. Anders. 2007. Characterising Space via Pattern Recognition Techniques: Identifying Patterns in Road Networks. In *Generalization of geographic information: cartographic modelling and applications*, eds. W. Mackaness, A. Ruas and T. Sarjakoski, 233-253. Oxford: Elsevier.

Reznik, S., and H. Mayer. 2008. Implicit Shape Models, Self-Diagnosis, and Model Selection for 3D Facade Interpretation. *Photogrammetrie - Fernerkundung - Geoinformation* 3:187-196.

Wonka, P., M. Wimmer, F. Sillion, and W. Ribarsky. 2003. Instant Architecture. *ACM Transaction on Graphics* 22 (3):669--677.