Automated processing of state civil maps of middle scales

Karel Staněk [a,*], Petr Šilhák [a]

[a] *Masaryk university Brno, Czech Republic, karst@geogr.muni.cz, 408102@mail.muni.cz*

* Corresponding author

Abstract: We want to provide information about the early stage of the automatization of civil topographic state maps in our presentation. Basic concepts and used procedures will be presented. Part of the presentation is database structure and implementation experiences. At the beginning is brief information about middle-scale topographic maps.

Keywords: cartographic generalization, rule based system, PostGIS, ArcGIS Pro, topographic base map

1. Introduction

On March 1st, a partial automatization project of the generalization map content, text placement, and map frame design started. The project's aim is not full automatization but to significantly decrease the number of cases solved manually by operators (to shrink three year period of the state map actualization nowadays to 2 years in the future). This implementation project follows previous stages. First, one evaluated the possibility of automated generalization (2015-2017) for base topographic maps 1:10000, and the second one in the year 2019 focused on the implementation plan and selection of implementation team for this actual project. The map assembly line's supposed enhancement is not only national map agency motivation for this project. The primary motivation is the continuously increasing difference between ZABAGED (Czech state database of fundamental geographic data) and base topographic map 1:10000. In the past, ZABAGED was derived from base map 1:10000, but in the last 20 years is the content of the ZABAGED enriched by features from cadastral maps, orthophotos, and lidar measurement. Therefore map 1:10000 is not just a symbolization of the ZABAGED, and generalization became a significant part of the assembly process. Finally, this project is the technological change of the map assembly line - nowadays, our national mapping agency uses ESRI ArcGis Desktop custom environment with Oracle database system. Our project is reflecting a technological shift in both GIS and DBMS tools. User interface and graphic generalization are implemented into ArcGIS Pro, and application logic with model generalization routines are implemented into PostGIS.

1.1 Middle-scales state maps

State map work (civil one - more than 50years ago were military and civil national map agencies separated, and still they are. Nevertheless, both map works use ZABAGED these days) encompasses middle map scales 1:10000,25000,50000 and 100000. In contemporary practice, maps in scales 1:10000 and 1:25000 are quite similar (annotations are strongly reduced on 1:25000, and some map features missing). Maps 1:50000 and 1:100000 are with a little exaggeration derived through enlargement or scaling-down compiled source map in scale 1:75000. Accidentally is a contemporary situation similar to old imperial map work in scales 1:25000 and 1:75000. Moreover, here we got another motivation of national map agency in the project - a clear definition of a separate map works to fit a particular scale better.

Contemporary practice is a mostly manual derivation of the map 1:10000 from ZABAGED snapshot (automated processing is limited to very light vertices weeding, ArcGIS provided building simplification and elimination of small objects). Map 1:25000 is compiled similar way; some results of the map 1:10000 are transferred. The base for scales 1:50000 and 1:100000 was manually drawn 20 years ago from ZABAGED. That time was ZABAGED conform with scale 1:10000. Nowadays, just running a manual update by comparison with topical ZABAGED snapshot, which has much higher shape granularity. Above mentioned geometrical base of maps, 1:50000 and 1:100000 is under the name DATA50, available for free to download.

2. Project description

There are many conceptual approaches to the implementation of automated cartographic generalization. In our proposal, several concepts are synthesized, among significant resources belongs from theory (Brassel and Weibel, 1988) and from pragmatic (Lecordix and Lemarié, 2007),(Schürer, 2008).

Our approach is based on rule-oriented processing, which encompasses the following principles:

- We try to define a deterministic system that covers a significant part of collision situations by massive rules.
- Similar rules support situations uncovered by identifiable rules.
- Too complicated, less frequent, and dissimilar situations are left for human operators.
- Rules are compiled based on map features in a collision, topometric relationships surrounding the collision, and finally, the relationship of colliding objects to the identified structures.
- Set of constraints identifying collisions.
- Collisions are processed

  - from the weakest map feature to the most stable ones
  - from simple collision situation to complicated ones
  - from structure members to unstructured objects.

Generalization processing is divided into

- Structure recognition and description (derived from source ZABAGED data). Structures can be either a single object, group of objects with various orders or zones.
- Object surrounding situation topometric description (used for rule identification and topological check)
- model generalization (uniform operations solving imperceptibility and auto-coalescence)
- Graphical generalization (symbology overlapping, solved are only cases where are symbols overlapped), both generalization processings are complemented by topological checks.

Model generalization is applied to previous model generalization as scales follow; just model generalization of the map 1:50000 is applied on the results of the graphical generalization of the map 1:25000.

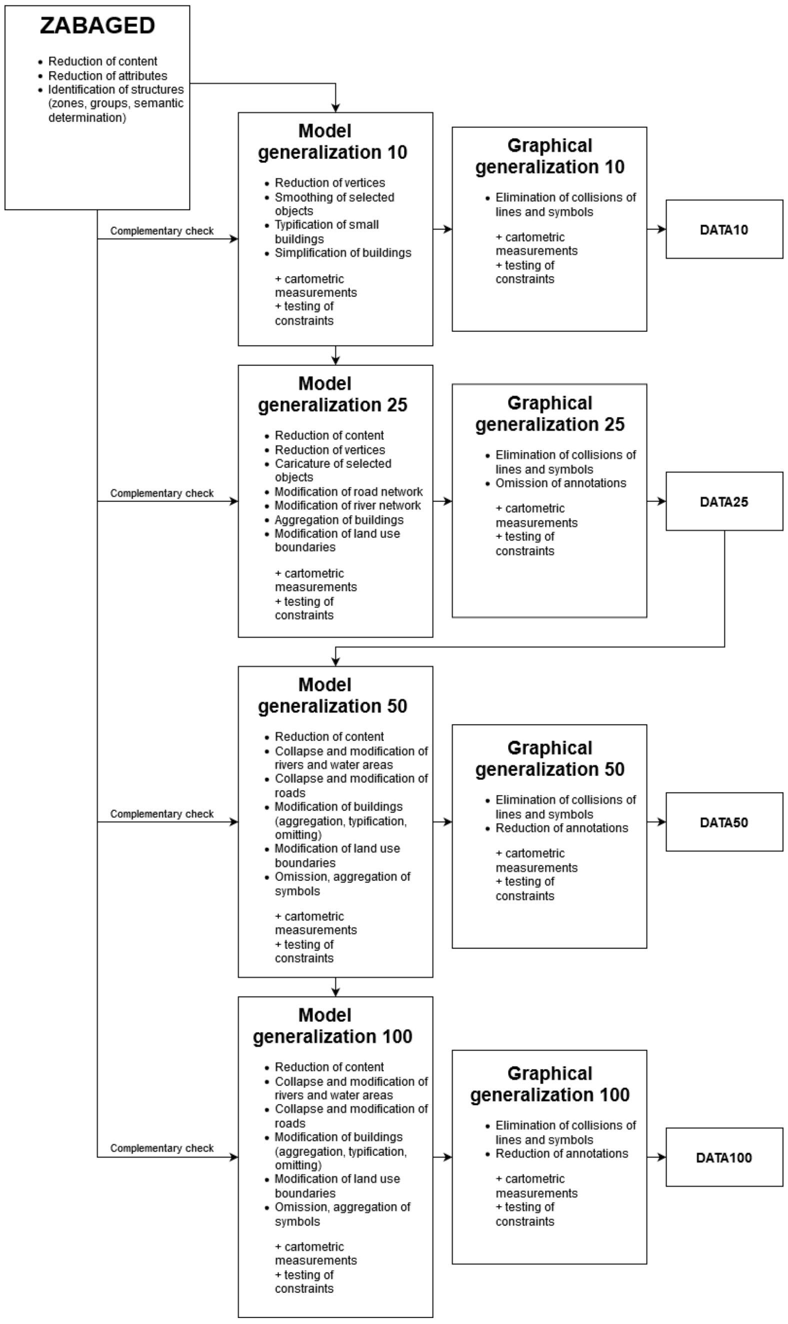
ZABAGED
Reduction of content
Reduction of attributes
Identification of structures (zones, groups, semantic determination)
Complementary check
Model generalization 10
Reduction of vertices
Smoothing of selected objects
Typification of small buildings
Simplification of buildings
+ cartometric measurements
+ testing of constraints
Graphical generalization 10
Elimination of collisions of lines and symbols
+ cartometric measurements
+ testing of constraints
DATA10
Complementary check
Model generalization 25
Reduction of content
Reduction of vertices
Caricature of selected objects
Modification of road network
Modification of river network
Aggregation of buildings
Modification of land use boundaries
+ cartometric measurements
+ testing of constraints
Graphical generalization 25
Elimination of collisions of lines and symbols
Omission of annotations
+ cartometric measurements
+ testing of constraints
DATA25
Complementary check
Model generalization 50
Reduction of content
Collapse and modification of rivers and water areas
Collapse and modification of roads
Modification of buildings (aggregation, typification, omitting)
Modification of land use boundaries
Omission, aggregation of symbols
+ cartometric measurements
+ testing of constraints
Graphical generalization 50
Elimination of collisions of lines and symbols
Reduction of annotations
+ cartometric measurements
+ testing of constraints
DATA50
Complementary check
Model generalization 100
Reduction of content
Collapse and modification of rivers and water areas
Collapse and modification of roads
Modification of buildings (aggregation, typification, omitting)
Modification of land use boundaries
Omission, aggregation of symbols
+ cartometric measurements
+ testing of constraints
Graphical generalization 100
Elimination of collisions of lines and symbols
Reduction of annotations
+ cartometric measurements
+ testing of constraints
DATA100

Figure 1. Process diagram with particular steps included for specific scale.

2.1 Knowledge acquisition

Knowledge acquisition is made by a combination of several methods. The first input was a list of most frequent collision situations provided by operators from the national mapping agency (NMA). That source was followed by a set of interviews with operators related to the map compilations. NMA provided us old manuals for map compilations. We are running extensive carto-metric measurements over ZABAGED and DATA50 (geometry of maps 1:50000). We are also trying to identify generalization procedures on middle-scale maps in our area in the 20th century. Last but not least is the analysis of the publications related to the automatization of the map compilation of other European NMAs. Surprisingly identification of constraints is more complicated than rules. The same trouble is acquisition structures from operator interviews.

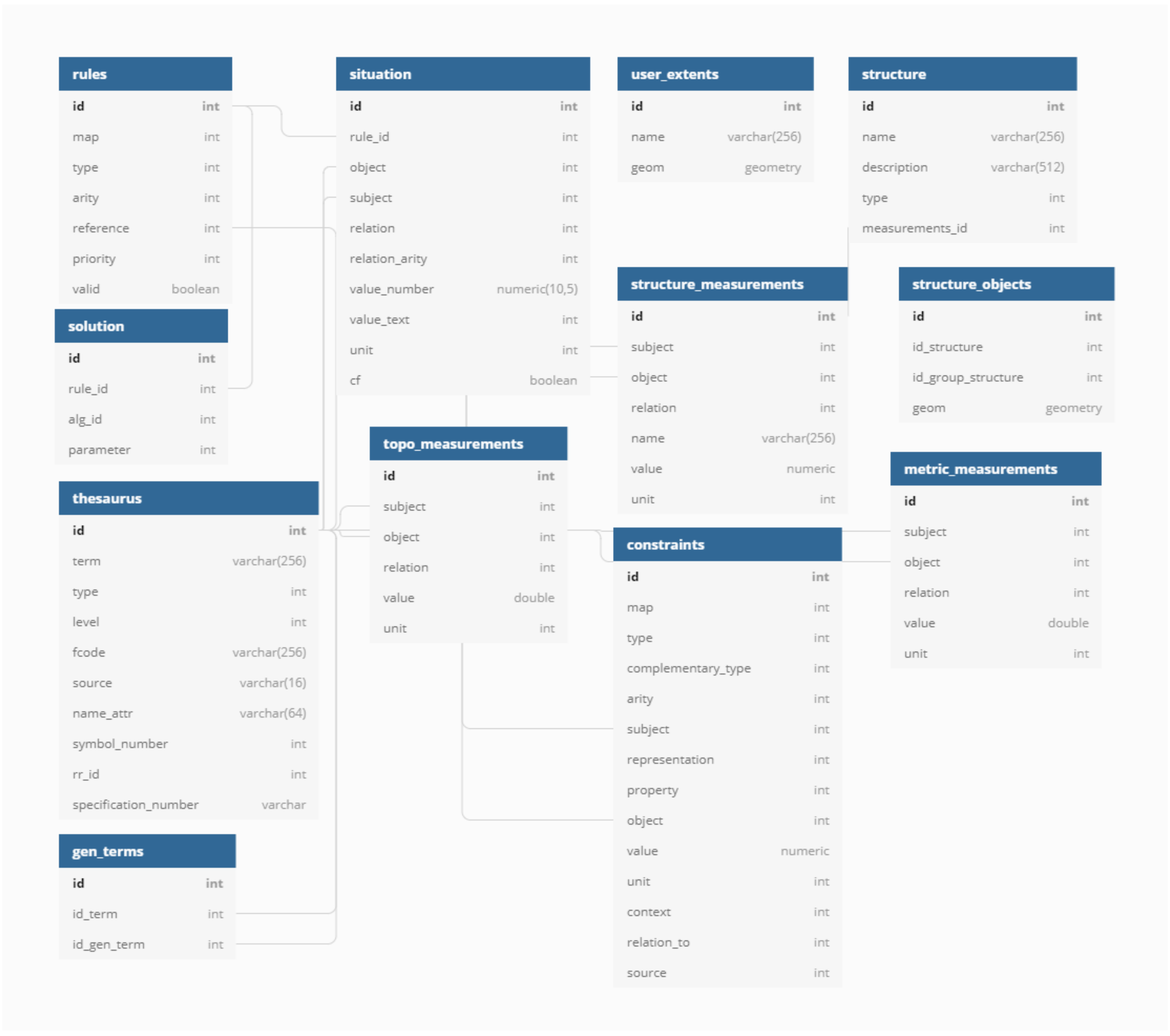

Figure 2. Database model of generalization implemented in PostgreSQL. Many tables have been omitted from the image (enumerations, logging tables, symbology tables etc.).

2.2 Application design

As was already mentioned, implementation of the generalization procedures running in ArcGIS Pro and PostGIS environments. Nevertheless, we do not use any generalization function implemented in ArcGIS Pro or PostGIS - we use geometric engines and in-house routines. Structure recognition and model generalization is implemented in C language, user interface, and graphic generalization in C#. We also do not use ArcGIS Pro database connector, but in-house developed one.

In the tender part of the project and this early stage of the regular part of the project, we are dealing with terrain steps (slopes, trenches, embankments, cuts), representing the most exhausting work for operators at the map 1:10000 (40-60% time concerning map sheet configuration). Slopes are the weakest and most frequent map feature with productive interaction with communications and water shores. Besides generalization, slopes need harmonization of the visualization to be included in the processing.

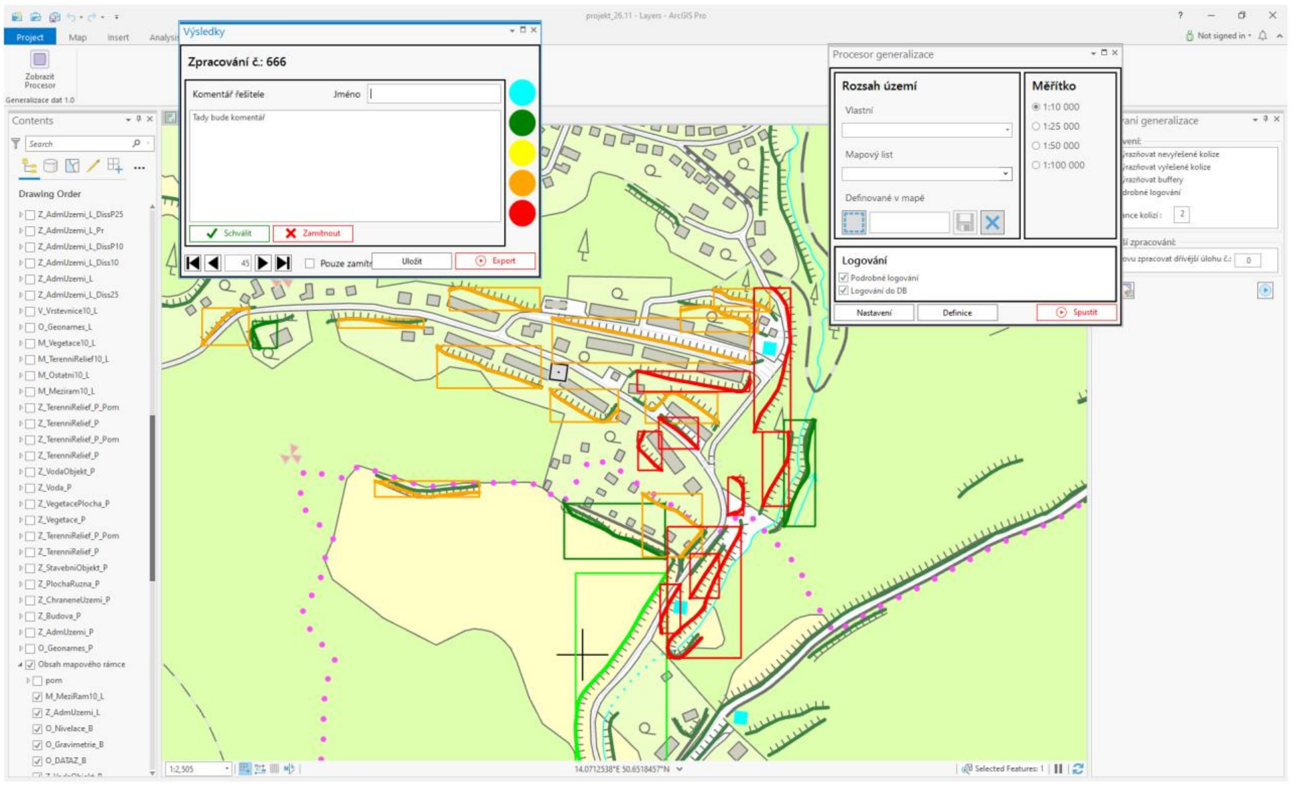

Figure 3. Generalization processor frontend. Results of graphical generalization of slopes.

3. Implementation into PostgreSQL database system

Our decision to use the PostgreSQL database system for analysis and model generalization, which will also serve as the main data store, was based on the assumption that we do not want to send large amounts of data between parts of the generalization systems during the computations. However, it was clear in advance that the implementation of some algorithms in PL/pgSQL will be complicated. Firstly, PL/pgSQL is an interpreted language that is more of a superstructure over the classic SQL language and therefore does not provide enough performance for compute-intensive operations. Sometimes code implemented in PL / pgSQL can be up to 100 times slower than similar code implemented in C. Secondly, the implementation of some algorithms would be very time consuming using PL/pgSQL syntax.

An alternative was the possibility of other branches of procedural languages implemented to PostgreSQL, such as PL/Java, PL/Python or PL/Javascript. At first, this option seemed relatively feasible but the projects seemed relatively unmaintained, and these languages also used the database connectors of each language to communicate with the database, so this approach would not work for us, because the data would not be processed directly in the database.

Finally, we decided to use functions written in C. Since PostgreSQL is written in C, there is relatively good support for this way of writing functions. In terms of performance, this is practically the best option that PostgreSQL provides. Moreover, due to the existence of the PostGIS library (which is also written in C), it is possible to get a huge number of examples written by experts.

```
PG_FUNCTION_INFO_V1(vectgen);
Datum vectgen(PG_FUNCTION_ARGS) {
    GSERIALIZED *geom = PG_GETARG_GSERIALIZED_P_COPY(0);
    int type = GSERIALIZED_GET_TYPE(geom);

    GSERIALIZED *result;
    double tolerance = PG_GETARG_FLOAT8(1);

    if ( type != LINETYPE) PG_RETURN_POINTER(geom);

    LWGEOM *in;
    in = LWGEOM_FROM_GSERIALIZED(geom);

    LWLINE *g = (LWLINE *)(in);
    POINTARRAY *pa = g->points;
    uint32_t in_npoints = pa->npoints;

```

Figure 4. Example of code C function implemented into PostgreSQL.

Implementing functions written in C into PostgreSQL is relatively easy if you do not need some third-party libraries. This was a little complication for us because we need to use the libraries contained in the PostGIS. For the initial test of the implementation, it was necessary to compile or use the compiled LWGEOM and PGCOMMON libraries (both are parts of the PostGIS). The first library provides the basic geometric types and operations used in PostGIS. The latter provides a bridge for the use of LWGEOM in PostGIS. After including these libraries, it is relatively easy to write very powerful compute-intensive functions in C language and use them as plugins in PostgreSQL. Example of how the structure of

PostgreSQL plugin written in C looks can be found in the PostgreSQL documentation.

The VectGen algorithm code sample can also be seen below. Input parameters are sent to the function via macros. This bridges the problem of data type mismatch between SQL and C. Type checking of input parameters takes place in the corresponding SQL function, which is one level higher than the C function.

4. Conclusion

The project is in the beginning. We focus on slope processing on larger-scale maps as a demonstration topic for application design. Parallel, we start water streams scale processing.

5. References

Brassel, K.E. and Weibel, R. (1988). A review and conceptual framework of automated map generalization, International Journal of Geographical Information Systems, 2:3, 229-244, 1988, DOI: 10.1080/02693798808927898.

Lecordix, F. and Lemarié, C. (2007). Managing Generalisation Updates in IGN Map Production. 10.1016/B978-008045374-3/50017-X.

Schürer, D. (2008). Das AdV-Projekt ATKIS-Generalisierung — Digitale Landschaftsmodelle und Karten aus dem Basis-DLM, KN - Journal of Cartography and Geographic Information, 58, 191-199, 2008, DOI: 10.1007/BF03543988.