A data model for multi-scale topographical data

J.E. Stoter[1], J.M. Morales[1], R.L.G. Lemmens[1], B.M. Meijers[2], P.J.M van Oosterom[2], C.W. Quak[2], H.T. Uitermark[3], and L. van den Brink[4]

[1] ITC, P.O. Box 6, 7500 AA Enschede, The Netherlands.
{stoter|morales|lemmens}@itc.nl

[2] Delft University of Technology, P.O. Box 5030, 2600 GA Delft, The Netherlands.
{b.m.meijers|c.w.quak|p.j.m.vanoosterom}@tudelft.nl

[3] Kadaster, P.O. Box 9046, 7300 GH Apeldoorn, The Netherlands.
harry.uitermark@kadaster.nl

[4] Dynasol BV, Buffelstraat 124a, 3064 AD Rotterdam, The Netherlands.
linda@dynasol.nl

ABSTRACT
The lack of fully automated generalisation forces National Mapping Agencies to maintain topographical data sets at different map scales. For consistency between map scales, but also for supporting (future) automated generalisation processes, information on similarities and differences of the separate data sets should be identified and formalised. This includes information on valid data content at the different scales ('scale state'), but as important is the semantics of multi-scale and generalisation aspects ('scale event'). As 'scale state' and 'scale event' are strongly related ('different sides of the same coin') it is important to integrate these in a single model. This paper presents a semantically-rich data model for an integrated topographical database, facilitating (semi-)automated generalisation. UML (including OCL) is used to formalise the model. The scope of the model is outlined and the model is presented based on an analysis of several alternatives for modelling multi-scale and generalisation aspects. The model is evaluated by instantiating the model and applying it to test data.

Keywords: multi-scale, spatial data modelling, generalisation, knowledge formalisation

1. Introduction

Integration of topographical data sets at different map scales is an important requirement for implementing automatic generation of (updates in) smaller scale data sets from (updates in) larger scale data sets. For implementation of this integration within a DBMS as well as of the implementation of (semi-)automated generalisation processes, a data model is needed which makes all required information and knowledge explicit in a formalised way. This comprises firstly information on data content covering all scales (as in information model design for a single data set): object classes, attributes, attribute values, constraints for valid data content and relationships between object classes within one map scale ('map scale state'). For supporting generalisation additional semantics on multi-scale aspects ('scale event') is required, such as how object classes and instances behave at map scale transitions and relationships between object classes and instances at different map scales. This paper presents a semantically-rich integrated data model for multi-scale topographical data sets, maintained by the Kadaster (Dutch Land Registry Office) facilitating (semi)automated generalisation between topographical data sets at different map scales. The designed multi-scale data model is called Information Model TOPography (IMTOP). The Unified Modelling Language (UML), including the Object Constraint Language (OCL), is used to formalise the model.

In Section 2 previous initiatives on multi-scale data modelling (2.1) as well as the three basic spatial data models (2.2) are presented. Section 3 defines the scope of IMTOP and presents the requirements for IMTOP. Section 4 describes the various steps that have been taken to design the multi-scale data model. The model is evaluated in Section 5 by applying the model to test cases. The paper ends with conclusions in Section 6.

2. Previous approaches for multi-scale and single data models

2.1 Data modelling approaches for multi-scale data

A multi-scale data model is a specific type of a multi-representation data model. The issue of multi-representation was introduced in a research program of the National Center for Geographic Information and Analysis (NCGIA 1989; Buttenfield and Delotto 1989). Since then many researchers have focused on this issue. The Multiple Representation Management

System (MRMS) of (Friis-Christensen and Jensen 2003) provides MR-methods such as 'checkConsistency' and 'restoreConsistency', as well as triggers to execute those methods in case of updates and insertions, modelled with UML and OCL and implemented on top of Oracle. The Modeling of Application Data with Spatio-temporal features (MADS) of (Parent et al. 2006) is based on stamps. One or several stamps can be assigned to object classes, relationships, attributes, values, etc. to indicate for which map scale the object class, etc. is relevant. 'Perceptory' (Bédard et al. 2004) is a plugin extending existing UML editors with spatio-temporal icons allowing modelling of multi-representation concepts in methods of the object classes. There is no independent description of multi-representation concepts, as in MADS. Jones et al. (1996) propose a conceptual model for a multi-representation database as a single database that is capable of storing spatial objects with multiple geometries. This approach does not take into account the complexity of the relationships that can exist in multi-representation (and multi-scale) data sets. The work of Devogele et al. (1996) models map scale transitions, but only between pairs of objects; it does not consider a complete topographical database. The work of Kilpelainen (1997) focuses on the link between object instances when there is an exact dependence among the object classes (e.g. building as complex polygon, building as simple polygon, building as point, building as part of a building area). What is new in the research presented in this paper is that the data model formalises all knowledge required for both integration and for automated generalisation of topographical data sets.

2.2 Three basic approaches for spatial data models

When looking at spatial modelling in the past, three main approaches can be distinguished: 1) geometry/topology-first approach, 2) object-first approach, and 3) a hybrid approach.Geometry is the main entrance for object classes in the geometry-first approach, often structured in a topological structure (e.g. a linear network, or a partition of space). Attributes are added to these geometries in order to classify the objects.

The object-first approach models the object classes first with added geometry attributes. Every object class can have its own set of thematic attributes which may vary for the different object classes. Every object class has its own geometric description independent of any other object. The model does not explicitly contain topological relationships, which are very important for generalisation; e.g. what are the neighbours of this instance

(candidates for aggregation)? is the network connectivity damaged when this road segment is removed? etc.

The third approach, the hybrid approach, treats geometry and the object class equally. It combines the strengths of both approaches: the thematic attributes are specifically designed for every object class, but the model also enables shared geometry and use of embedded structures. The spatial domain is a full partition and the result is described using tables for nodes, edges, and faces (and solids in 3D). The instances are modelled in the same way as in the object-first approach with the exception that objects do not have their own independent geometry attributes, but refer to primitives in the geometry/topology part of the model (node, edge, face,…). This is the approach as described in the 'formal data structure' (FDS) theory of Molenaar (1989) and quite recently implemented in products such as 1Spatial's (formerly LaserScan) Radius Topology, and Oracle's spatial topology (first introduced in version 10g). It cannot be claimed that one model is 'better' than another model. This depends on the application context and use. If one specifies a number of important characteristic of the application domain and typical use, then it is possible to state which approach is preferred (Stoter et al. 2007).

3. Scope of IMTOP

3.1 Previous initiatives on multi-scale models as input for IMTOP

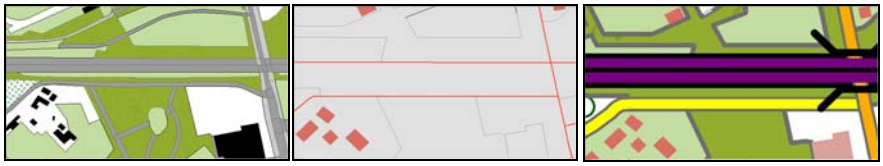
What the Information Model TOPography (IMTOP) adds to past research is that the model specifically focuses on data sets that cover the same reality using a similar set of object classes. It models how classes, as well as their instances, change at map scale transitions. In that sense IMTOP does not model different (=multi) representations of real world objects. Instead IMTOP models one collection of object classes together with semantically-rich information on scale transitions. Another specific aspect of IMTOP is that complete topographical coverage at every scale needs to be modelled (there are no gaps), which is more inclusive than defining inheritance relationships between object classes as in most multi-representation approaches. An object cannot be eliminated without being merged with another object because of the topological structure at every scale. For example, in TOP10NL (1:10k base map of The Netherlands) faces of the topological structure consist of road polygons, water polygons and land use. In TOP50NL (the 1:50k map), and smaller scales (TOP100NL,

TOP250NL, and TOP500NL), road polygons are collapsed, and therefore faces in topological structure are formed by water polygons and land use.

For IMTOP the hybrid approach (Section 2.2) was identified as optimal approach. Some criteria justify the object-first approach with functionality also supported in the hybrid approach, but not in the geometry-first approach: bridge over water should be allowed; administrative area can overlap topographical objects; multi-geometry of objects should be possible, e.g. both center lines and polygons for roads. On the other hand topological structure as available in the geometry-first and hybrid-approach, but not in the object-first approach is needed for automated generalisation. Therefore IMTOP is based on the hybrid approach and topological primitives are used to model geometries. The model will adhere to ISO and OGC standards as much as possible.

3.2 Integration of landscape model and cartographic model

For every map scale, Kadaster supplies two products that should be covered by IMTOP: a data set for GIS analyses and a digital map, which is a cartographic version of the data set. It is not trivial to answer the question whether a so-called Digital Landscape Model that does not take into account any symbolisation (DLM) as well as a so-called Digital Cartographic Model (DCM) should be available (and modelled) at every scale. Current TOPxxvector data sets as supplied by Kadaster integrate DLM and DCM aspects: the geometries in the vector data sets take already into account the way they will appear on the map. For example, a motorway in TOP50vector will be portrayed with a line-symbol of width 1.5 mm, which is 75 meter in reality. To avoid overlap of the motorway symbol with other instances such as buildings, instances are displaced and simplified in current TOPxxvector products. Creating the map is so to speak a simple button push, which adds symbology to the geometries (see Figure 1).



TOP10NL TOP50vector TOP50MAP

Fig. 1. TOP10NL, TOP50vector and TOP50MAP in current production process of Kadaster

From a theoretical point of view it seems straightforward to distinguish between database and cartographic representation, since inaccuracies because of symbolisation are avoided in the database. However, for IMTOP it was decided to integrate the landscape and cartographic model. The database product is therefore a vector representation of the map. This approach is also applied by for example KMS, Denmark (West-Nielsen and Meyer 2007). There are several arguments which favour this approach:

- Generalisation leads to loosing accuracies, whether this is for the database or the map. The inaccuracies of current vector products are no problem for GIS analyses at small scales. If more accurate data is needed, one can use TOP10NL where symbolisation does not yield major graphical conflicts and therefore does not yield inaccuracies.
- It was tried for IMTOP to separate between DLM and DCM, in contrast to current practice. This showed that for many transitions it is not easy to identify where they fit, e.g. elimination of small buildings.
- A multi-scale topographical database requires keeping data models as well as databases at all scales consistent. Separation between model and cartographic representations requires twice as many data models and databases to keep consistent.

3.3 Basic requirements for IMTOP

Main objective of IMTOP is supporting semi-automated generalisation of medium to small map scales. The criteria that follow from this requirement comprise the possibilities:

1. To have a model describing topographical data in the scale range 1:10k to 1:1,000k at specific scales (e.g. 1:10k, 1:50k).
2. To extract a UML class diagram for a specific map scale.
3. To produce a GML application schema (.xsd) for a specific map scale from the UML diagrams generated in step 2.
4. To produce a GML application schema (.xsd) for multi-scale topographical data.
5. To generate the DBMS structure for a specific map scale from every schema generated in step 3.
6. To generate the DBMS structure for multi-scale topographical data, including scale transition information for instances and object classes.
7. To populate the multi-scale database and link instances at different map scales.

8. To use all information (including supporting structures, map scale and transition information) directly in the generalisation process, with minor human intervention.

The model was tested on these requirements to see to what extent IMTOP is suitable for multi-scale data modelling taking generalisation aspects into account. Results are reported in Section 5.

4. A data model for multi-scale topographical data

IMTOP is a result of several steps:

1. Designing data models at separate map scales in UML including refined semantics expressed in OCL.
2. Integration of data models at different scales to model scale transitions of object classes that are apply to a complete set of an object class, e.g. conversion of geometry types (collapse or combine), or reclassification, for example ‘streets’ and ‘local roads’ are eliminated at scale 1:250k, and smaller.
3. Extend the model with semantics on transitions that apply only to specific instances.

These steps are described in respectively Sections 4.1, 4.2 and 4.3.

4.1 Modelling object classes, attributes, and relationships at separate scales

An analysis of current product specifications for TOPxxvector products (Kadaster, 2002) showed which object classes, attributes, attribute values, and geometry types (which could be multiple geometries) should be modelled in IMTOP. Separate UML diagrams were designed per map scale to cover this information. For generalisation, the diagrams at the separate scales should also express how the specific data as generalisation output should look like. This information is currently available in generalisation specifications, software code or even in human minds; only at the human knowledge level, since the information is meant for cartographers to be used in interactive generalisation processes. From current generalisation specifications (Kadaster, 2006) generalisation-related constraints within and between different object classes were deduced, and added to the diagrams in OCL. In a case study it was evaluated to what extent this information can serve the automated generalisation process (see Section 5).

The constraints that were defined in this process are called *invariants* in OCL terminology and express a valid state of the data set at a certain map scale. This is the 'scale state' aspect of the model. Based on these constraints, a data set at a particular level of detail can be validated (e.g. all buildings have a minimum size at scale 1:50k).

Specifying valid contents of the model is the known role of constraints in data modelling. Constraints are used more and more to include additional semantics in data models (Louwsma et al., 2006; Oosterom, 2006). In IMTOP, constraints addressing semantics on transition processes were added in a next phase: this is the 'scale event' aspect of the model. For that phase, constraints are expressed as part of the change from one level of detail to the next, i.e. as *pre conditions* and *post conditions* of the transition process, for respectively selecting appropriate large scale input (i.e. instances) and for checking resulting smaller scale output which might trigger another event. The result of the scale transition can be stored via associations between larger and smaller scale object classes and instances. Therefore, post conditions of scale transitions are related to the invariants (constraints at the resulting scale). See Section 4.3 for pre and post condition constraints.

In this section, constraints of the 'invariant' type are presented, thus defining which data content is allowed, only addressing aspects within one scale. The conceptual modelling of these constraints using OCL is illustrated in Figure 2, which depicts an excerpt of the model for the classes 'Building' and 'Road'.

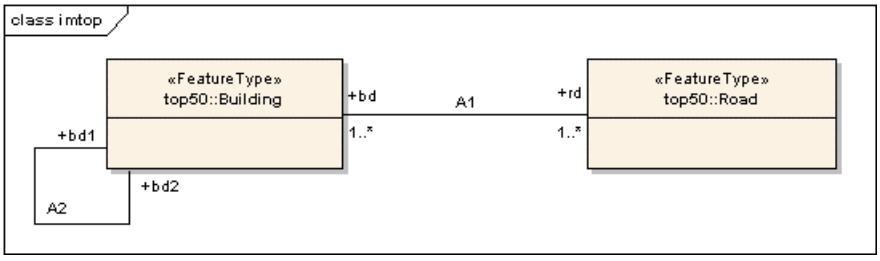


Fig. 2. UML class diagram of object classes 'Building' and 'Road'

We distinguish four types of spatial relationships constraints related to the partial model in figure 2:

- Type 1: constraints on a single instance from a single class.
- Type 2: constraints between two instances belonging to the same class, e.g. between two buildings.
- Type 3: constraints between two instances belonging to different classes, e.g. between building and road.

- Type 4: constraints on a group of instances, e.g. group of buildings.

In the UML class diagrams, constraints of types 2, 3 and 4 are modelled as constraints which navigate through associations. In the example of Figure 2 an association A1 is added between Building class and Road class for TOP50NL, with the following constraint in OCL (of Type 3) defining that roads and buildings must be disjoint:

```
context top50::Building
inv:
  -- Building and Road should be disjoint
  Disjoint(self.Geometry,rd.Geometry)
```

Note that the predicate 'Disjoint' can be evaluated because of the topological structure available in the hybrid-approach. A constraint of Type 2 is defined through association A2, identifying a minimum distance between two symbolised buildings:

```
context top50::Building
inv:
  -- Minimum distance between buildings is 0.2 mm in
the map
  Distance(self.Geometry,bd2.Geometry)>=0.2
```

An example of a Type 4 constraint is the constraint that building instances on area of land use type 'other' should never exceed 10% of the area coverage. The action to be taken if objects do not adhere to the constraint (e.g. 'displace') are not modelled in this part of the model, as this is an issue of the transition process between the different map scales (see Section 4.3). Functions such as 'Disjoint', and 'Distance' as used in these examples are assumed to have standardised implementations in software. Standards from ISO and the Open GeoSpatial Consortium (OGC) are used as much as possible.

As stated before, these example invariant constraints will also be expressed as post conditions of scale transitions. Figure 3 shows how constraints appear in the UML modelling software used in this research (Enterprise Architect, 2008).

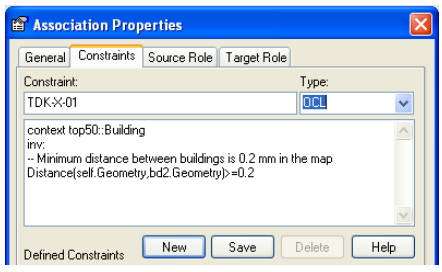


Fig. 3. Appearance of OCL constraints in the UML modelling software

4.2 Integration of data models at different map scales

The integration of the separate data models should make explicit what happens to object classes at scale transitions. For IMTOP these scale transitions are identified from product and generalisation specifications (Kadaster, 2002; Kadaster 2006). An analysis of the data models as generated in step 1, showed that going to smaller scales does not only lead to reducing information, but sometimes adding information. An example is the attribute value 'roundabout' for TypeOfInfrastructure, which is not registered at map scale 1:10k (since roads are represented by polygons) but is needed at map scale 1:50k, where roads are collapsed and roundabouts are represented by points. There are basically three alternatives for the integration which are discussed below.

Constraints for scale dependent attributes and attribute values

The appearance of the object class at a specific map scale is defined by constraints to allow or disallow attributes and attribute values at specific scales. For example 'if 1:50k then geometry type of 'secondary roads' is line'. The disadvantage of this approach is that the model will not be easy to read as a lot of OCL expressions have to be inspected and these OCL constraints are used both for modelling valid data content as well as for modelling scale dependent information. It is also not easy to automatically derive a model per scale.

Inheritance and derived attributes for scale dependent information

For every class that occurs in topography an abstract superclass is modelled containing attributes that are valid at the starting scale (TOP10NL in our case). A subclass is modelled as specialisation for TOP10NL, whereas all similar object classes at the other scales are modelled as a derived class from the previous scale. An example for the 'Road' object class is shown in Figure 4 for TOP10NL, TOP50NL and TOP100NL. Road classes at scales 1:50k and 1:100K contain derived attributes (indicated with '/'). The derivation rules can be modelled in OCL (for example: 'derive: derivedFromTOP50NL.typePavement' for typePavement in TOP100NL). Apart from derived attributes, the classes contain two other types of attributes: a) attributes that are introduced at this scale (e.g. 'exit' for Roads in TOP50NL) and b) attributes that disappear at this specific scale, indicated with multiplicity of 0 (e.g. 'geometrySurface' for Roads in TOP50NL and TOP100NL and 'exit' in TOP100NL). The inheritance approach is only used for object classes, and not for enumeration since inheritance is only

appropriate for object classes according to ISO 19103. Advantages of this approach are that the model is easy to read and it is easy to get back to a model per map scale by just showing the relevant classes for that map scale only.

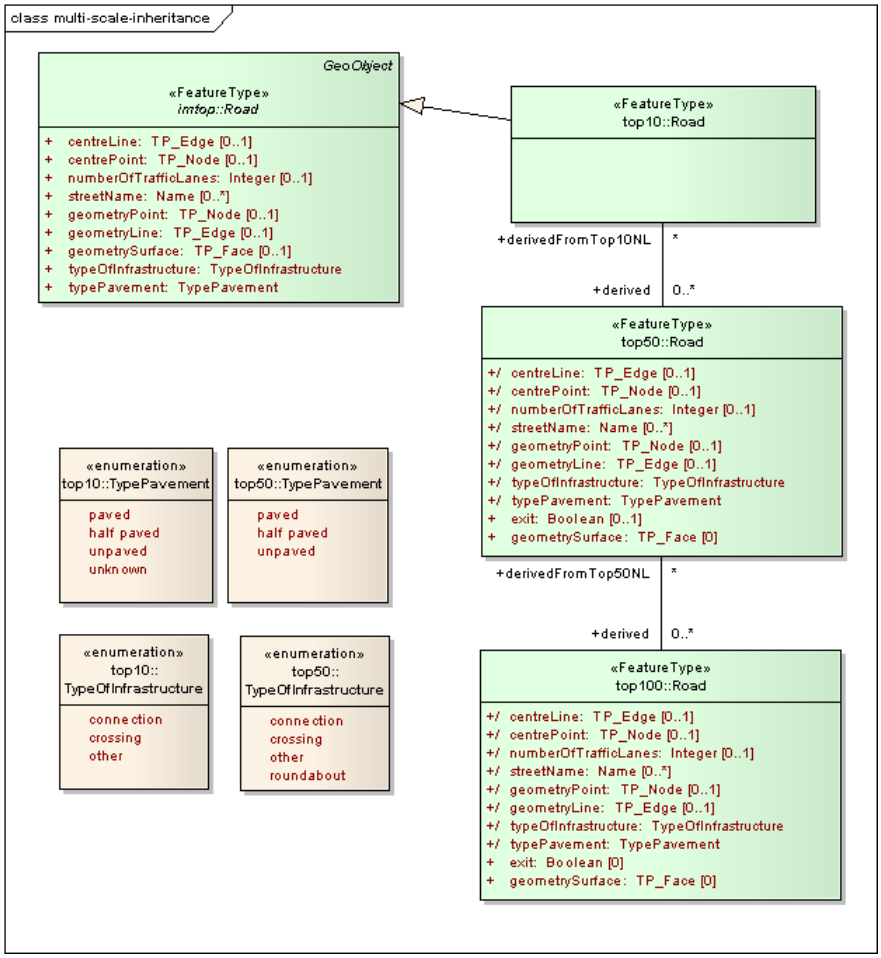


Fig. 4. Using inheritance, derived attributes and multiplicity on attributes to define what is valid on every scale

Stereotypes for multi-scale semantics

In the third approach the underlying meta-model of UML was extended with multi-scale aspects. UML stereotypes and tagged values that can be

used to model additional semantics in UML are applied to extend the UML model. The stereotype <<MultiScale>> is used to indicate that the given class will have different representations at different map scales (similar to stamps in MADS, see Section 2.1). Attributes and attribute values of MultiScale objects that are labelled with the <<MultiScale>> stereotype, get a 'minScale' and 'maxScale' tag to indicate on which map scales they are valid. In the example of Figure 5 all spatial attributes are stereotyped with the <<MultiScale>> stereotype and the correct minScale and maxScale tags are added. Disadvantage is that the tags are not visible in the class diagram itself, as can be seen in Figure 6. In addition, the model is very compact and it is therefore not easy to read: every object class (e.g. 'Road') is modelled as a single object class for all map scales and multi-scale aspects are only visible when analysing the specific attributes and attribute values. Finally it is not easy to automatically generate separate UML models from this model.

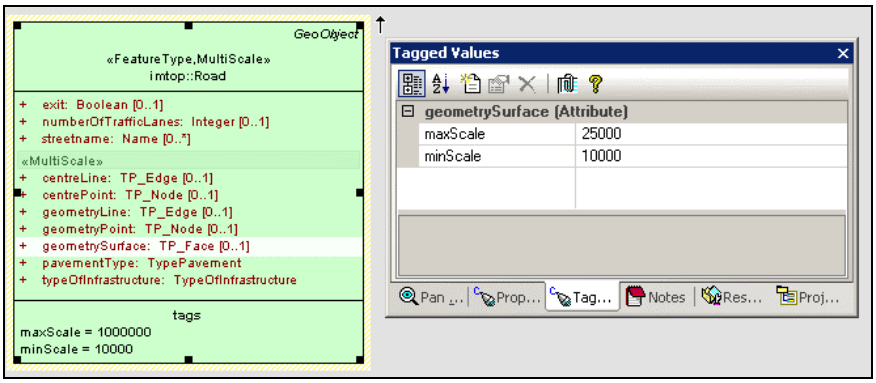


Fig. 5. Example of using a MultiScale stereotype to model map scale dependent information. The tags are only available in the GUI of Enterprise Architect

Based on the considerations outlined above the second approach was selected for IMTOP. Constraints are used to address valid data content whereas scale-related information is modelled with inheritance, which makes the model transparent. It is also possible to extract a data model for a specific map scale as will be seen from the tests in Section 5.

4.3 Modelling map scale transitions

With consistent sets of object classes throughout the various map scales, the model is completed with transition relationships, which further formalise knowledge on the generalisation process. Transition relationships rep-

resent associations between instances and object classes at different map scales. Semantically, these relationships represent transition paths for specific instances from one map scale to another. This as additional information to the information on transitions applied to all instances of an object class (see Section 4.2). To capture the transitions that can differ per instance, the model is extended with transition models for volatile transition classes and processes. Figure 6 shows a simplified example of a transition model for the transition path to generate *TOP50NL built-up area* instances from *TOP10NL* instances.

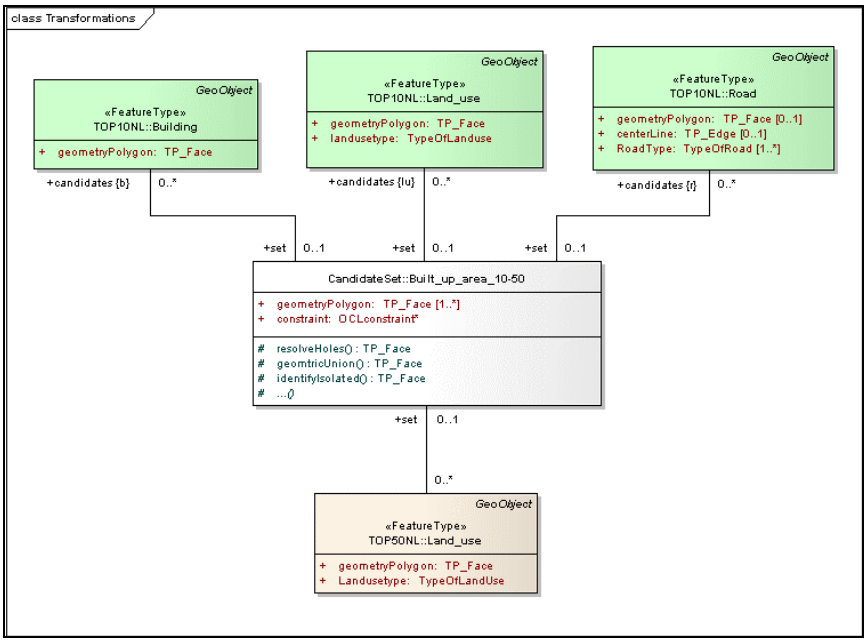


Fig. 6. Model of transitions for generating built-up area in TOP50NL from TOP10NL buildings.

A transition process specifies firstly the selection of instances based on *pre conditions* and secondly actions that should be applied to those instances at map scale transition based on *post conditions* (which can be similar as the pre conditions used for selecting instances). Central concept in the process is the so-called “Candidate Set” class which is a container for instance sets derived from source classes that are potential members of a target class. The class has an attribute called *constraint*. This attribute is used to define pre conditions in the form of OCL constraints to identify instances from an object class at a source map scale that should go to the Candidate Set class. At scale transitions the constraints are evaluated to populate the Candidate

Set class. Each instance in the Candidate Set class is a collection of objects. Each candidate set is determined by its own constraints. Consequently new constraints for other sets can easily be added afterwards. This is why pre condition constraints are defined as attributes of the Candidate Set class and not on the class itself. Post conditions are defined on the Candidate Set class to specify how the instances in this class should be treated in the generalisation process. As was indicated in Section 4.2, these post conditions are similar to the invariant constraints defined for the separate scales. The transition models of IMTOP cover in the first place the currently available generalisation specifications (see also Section 4.1). In a case study it was evaluated to what extent this information specified in IMTOP can serve the automated generalisation process (see Section 5). In the future the transition models can be extended with new, machine-based knowledge on generalisation. The steps used for the transition in Figure 6 are:

1. Define all candidate instances from TOP10NL on their pre conditions:
   a. instances from *land use* class with *land use type* value 'built-up area', in OCL:

```
cs1    context cs:built_up_area_10-50
       cs.candidates→select( lu | lu.landusetype =
       'built-up area')
```

   b. instances from the *land use* class with *land use type* value 'other' that contain buildings that cover more than 10% of the area of the land use object, in OCL:

```
cs2    context cs:built_up_area_10-50
       cs.candidates→select( lu | lu.landusetype =
       'other'
       and b→exists( b | b.geometryPolygon iscon-
       tainedby(lu.geometryPolygon)
       and area(b.geometryPolygon) >
       area(lu.geometryPolygon)*0.1))
```

   c. instances from the *road* class that touch any of the instances in cs1 or cs2 (as a consequence of the collapsing of road polygons to road centrelines), in OCL:

```
cs3    context cs:built_up_area_10-50
       cs.candidates→select( r | r.geometryPolygon
       touches(..cs1_constraint..)
       or rd.geometryPolygon touches(..cs2_constraint..))
```

   d. instances of the *land use* class of any *land use type* located in potential *built-up area* and smaller than *x* m^2, in OCL:

```
cs4    context cs:built_up_area_10-50
```

```
cs.candidates→select( lu | lu.geometryPolygon
iscontainedby(cs1.geometryPolygon or
cs2.geometryPolygon)
and area(b.geometryPolygon) < x)
```

2. Populate the candidate class with the indentified instances.
3. Trigger generalisation processes by post conditions defined on the Candidate Set class for TOP50NL built-up area. Note that post conditions a and d are related to pre conditions b (*cs2*) respectively d (*cs4*):
   a. Instances of land use type 'other' can be covered by buildings by at most 10%
   b. Holes are not allowed
   c. Topologically adjacent instances should have no boundary between them
   d. Small instances of *land use* are not allowed
4. Generate a new set of geometries by applying operations on the *candidate object* class based on the post conditions in step 3; assign instances to land use class in TOP50NL and update attribute values
5. Recycle instances that were part of the candidate object class but are not transformed into a TOP50NL land use class of type '*built-up area*' after having finished the process.

UML state diagrams can be added to model the generalisation process in more detail (see Figure 7).

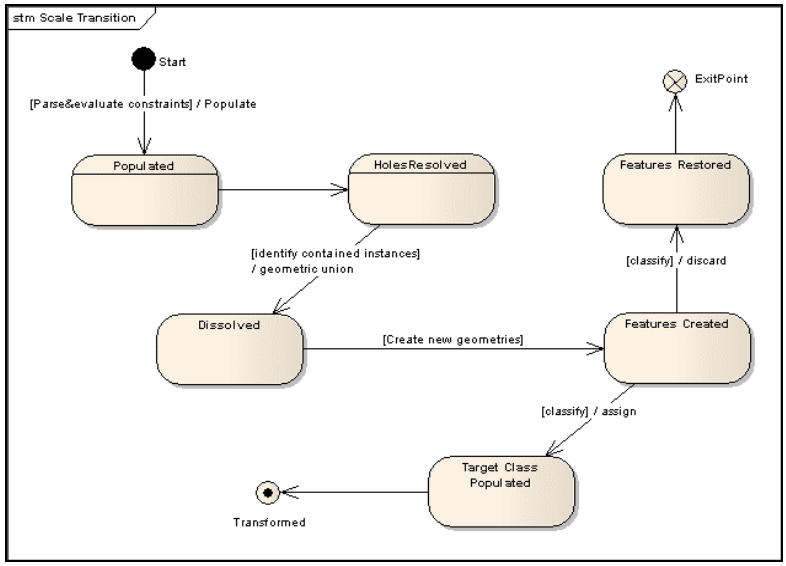


Fig. 7. State diagram for the generation of built-up areas

5. Results of IMTOP with respect to the requirements

Tests were carried out that instantiated the model in order to evaluate IMTOP on the requirements as defined in Section 3.3. From these tests the following conclusions can be drawn. From the model as selected in Section 4 (see Figure 4), it was easy to extract a UML model for a specific scale, since the separated scales could be modelled as separated packages in the model. In addition it was possible to extract an XSD file per map scale from the UML model, as can be seen from the XML fragment in Figure 8. An XSD file for a specific map scale refers to the XSD file for the general model. This reference is generated automatically and is triggered by the specialisation relation between classes in specific scale models and the general model. An XSD file for all map scales could not be easily generated, but is easy to build (once) by hand as its only content are references to all specific map scale XSD files and the general XSD file. The ISO primitives such as TP_edge which are used in the models are not automatically translated to the corresponding GML types. A simple type mapping was implemented for GML types using an Extensible Stylesheet Transformations (XSLT) stylesheet, and applied to the generated XSD files.

```
<?xml version="1.0" ?>
- <xs:schema targetNamespace="http://www.kadaster.nl/schemas/top10nl" xmlns:xs="http://www.w3.c
    xmlns:top10nl="http://www.kadaster.nl/schemas/top10nl" xmlns:imtop="http://www.kadaster.nl/sc
    <xs:import namespace="http://www.kadaster.nl/schemas/top50nl" schemaLocation="top50nl.xsd" />
    <xs:import namespace="http://www.kadaster.nl/schemas/imtop" />
    <xs:element name="Road" type="top10nl:Road" />
  - <xs:complexType name="Road">
    - <xs:complexContent>
      - <xs:extension base="imtop:Road">
        - <xs:sequence>
            <xs:element name="derived" type="top50nl:/Road" minOccurs="0" maxOccurs="unbounded" />
          </xs:sequence>
        </xs:extension>
      </xs:complexContent>
    </xs:complexType>
  - <xs:simpleType name="TypeOfInfrastructure">
    - <xs:restriction base="xs:string">
        <xs:enumeration value="connection" />
        <xs:enumeration value="crossing" />
        <xs:enumeration value="other" />
      </xs:restriction>
    </xs:simpleType>
  - <xs:simpleType name="TypePavement">
    - <xs:restriction base="xs:string">
        <xs:enumeration value="paved" />
        <xs:enumeration value="half paved" />
        <xs:enumeration value="unpaved" />
        <xs:enumeration value="unknown" />
      </xs:restriction>
    </xs:simpleType>
  </xs:schema>
```

Fig. 8. XML fragment showing part of XSD file for TOP10NL, generated from IMTOP

The UML model was exported into DDL (Data Definition Language) scripts that can generate the DBMS structure for the storage of the topographic data at the different scales. A relational DBMS with support for spatial types (PostGIS) was used. This experiment showed that the object-oriented hierarchy of a UML model is not easily mapped on flat DBMS tables as also concluded in (Sparks 2001). Therefore some database design choices were made in order to convert the UML model into a DBMS model. The following conversion rules were defined to automatically generate the DBMS model:

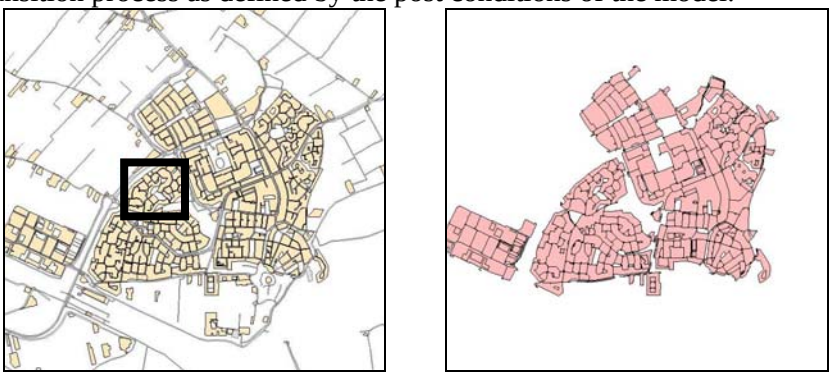
- Each non-abstract UML class (i.e. a class of which instances exist) is mapped to a DBMS table. The table contains columns for all the attributes of that class plus all the attributes of all superclasses.
- Each UML enumeration is mapped to a separate DBMS table and the use of an enumeration as an attribute is implemented as a foreign key to that table.
- References to topology items are implemented as foreign keys to a topological subsystem in the DBMS
- UML Associations are mapped to foreign key relations. ‘Many to many’ association need an intermediate table for the storage of the association.
- Multi-valued attributes are mapped to association tables.
- UML packages are mapped to DBMS table spaces.
- OCL constraints are mapped to Structured Query Language (SQL) views.

The conversion from a UML model to a DBMS structure (via DDL scripts) can be automated via a transformation language as shown in (Hespanha et al. 2008). The resulting DBMS model fits with the other requirements of IMTOP as defined in Section 3.3.

Also experiments were carried out to test the information on scale transitions specified in IMTOP on real data. For these experiments the TOP10NL part of the database was populated with TOP10NL data. According to IMTOP, instances from the source scale were selected first. After this selection, a procedure was applied to change the selected instances (either discard them, or adapt them to the target scale) based on invariants and post conditions. In this process views were generated on the original TOP10NL data tables using the constraints. For example the constraint specifying that instances of the class ‘buildings’, with attribute 'type of building' equal to 'glasshouse' should be removed if their area on the target map will be smaller than 0.36 map mm2 can be easily specified (with real world coordinates) in SQL. The view definition for the objects that should be removed based on this constraint could be as follows:

```
SELECT b.ident, b.typeofbuil, b.heightclas, b.id, b.geom
FROM building_area b
WHERE b.typeofbuil::text = 'glasshouse'::text AND
area2d(b.geom) <= 900::double precision;
```

Human interaction was used to express the constraints in SQL. An example to show how the transition model can be applied on a real case, is shown in Figure 9. In Figure 9a candidate instances are identified for built-up area in TOP50NL (see Figure 6) based on the input object classes and the constraints specified in the transition model. In Figure 9b all the polygons of land use type '*built-up area*' are shown that were formed after the transition process as defined by the post conditions of the model.



a. Candidate instances b. Transition result (Zoom-in)

Fig. 9. Snapshots of the model-based generation of built-up areas

The tests on the scale transitions were also used to validate if strictly applying currently available generalisation specifications specified in constraints, without adding any human interpretation, results in expected output. From the experiments it can be concluded that it is not always straightforward to apply the generalisation-related constraints. On the one hand the experiments showed that cartographer's interpretation is sometimes difficult to formalise. On the other hand they showed that the constraints need to be enriched for machine-based interpretation. An example of a constraint that is not sufficiently formal is the constraint that specifies that roads that are dead ends, classified as 'other', having main traffic use 'mixed traffic' and with a width of '2–4 meter' should be removed, except when the instance is longer than 100 meters and leads to a building. Two things that are not straightforward in this constraint is: how to evaluate 'a dead end' and how to evaluate 'leads to a building'. Connectivity is involved in both ambiguities and could be solved by using topological primitives and expressing the constraints with operations using this topological information.

6. Conclusions

This paper presented a data model (called IMTOP) for an integrated topographical database containing rich semantics on generalisation in order to support automated generalisation. After analysing several modelling approaches in UML, a modelling approach was selected that covers both data content and scale transitions that apply to complete sets of object classes in a transparent way. Transition models are added to IMTOP to model transitions that apply to specific instances. This paper showed that OCL constraints can be used to model valid data content at separate scales as well as pre- and post conditions to trigger generalisation processes.
The results of instantiating the model to evaluate it against the model requirements show that it is possible to formalise information on the integration and generalisation with UML and OCL and to use it to generate a multi-scale database. The formalisation is meant to express generalisation information in an unambiguous way. The experiments show that more work is needed to improve the formalisation of generalisation-related information, specifically the formalisation of cartographer's interpretation that is currently used in interactive processes.
Besides supporting the workflow of the Kadaster in the generalisation process, IMTOP may also be the basis for users of multi-scale topography. This has important added value compared to independent scales. Future research will further focus on the possibilities of using IMTOP for a Model Driven Architecture approach to support the integration of databases at several scales, as well as automated generalisation. This requires a high level of formalisation. It is also future ambition to extend IMTOP with larger scale base data (1:1k). Finally, vario-scale models and progressive transfer will be investigated, which is outside the scope op the IMTOP project, but within the research of the involved authors (Oosterom 2005).

References

Bédard Y, Larrivée S, Proulx M-J, and Nadeau M (2004) Modelling geospatial databases with plug-ins for visual languages: a pragmatic approach and the impacts of 16 years of research and experimentations on Perceptory. LNCS 3289. Springer, Berlin, pp 17–30.

Buttenfield BP, and Delotto JS (1989) Multiple representations. National Center for Geographic Information and Analysis (NCGIA). Scientific Report for the Specialist Meeting, Technical paper 89–3, 87p.

Devogele T, Trevisan J, and Raynal L (1996) Building a multi-scale database with scale transition relationships. In: International Symposium on Spatial Data Handling, pp 337–351.

Enterprise Architect (2008) http://www.sparxsystems.com.au/, accessed on 22–Jan–2008.

Friis-Christensen A and Jensen CS (2003) Object-relational management of multiply represented geographic entities. In: Proceedings of the Fifteenth International Conference on Scientific and Statistical Database Management. Cambridge, MA, USA, July 9–11, pp 183–192.

Hespanha J, van Bennekom-Minnema J, van Oosterom PJM, and Lemmen C (2008) The MDA approach applied to the Land Administration Domain Model with focus on constrains specified OCL. Proceedings of FIG Working Week, 14–19 June, Stockholm, Sweden

Jones CB, Kidner DB, Luo LQ, Bundy GL, Ware JM (1996) Database design for a multi-scale spatial information system. In: IJGIS, vol 10, 8, pp 901–920.

Kadaster (2002) Specificaties TOPxxvector (*Specifications TOPxxvector*). Topografische Dienst, Emmen, The Netherlands.

Kadaster (2006) Generalisatievoorschriften TOP50vector (*Generalisation regulations TOP50vector*) Topografische Dienst, Emmen, The Netherlands.

Kilpelainen T (1997) Multiple representation and generalisation of geo-databases for topographic maps. PhD thesis, Finnish Geodetic Institute.

Lemmen CHJ, and van Oosterom PJM (2006) Version 1.0 of the FIG Core Cadastral Domain Model. XXIII International FIG congress, October, Munich, 18p.

Louwsma JS, Zlatanova S, Lammeren R, and van Oosterom PJM (2006) Specifying and implementing constraints in GIS – with examples from a geo-virtual reality system. In: GeoInformatica, vol 10, 4, pp 531–550.

Molenaar M (1989) Single valued vector maps: a concept in Geographic Information Systems. Geo-Informationssysteme, vol 2, 1, pp 18–26.

NCGIA, 1989, The research plan of the National Center for Geographic Information and Analysis. In: Int. J. Geographical Information Systems, vol 3, 2, pp 117–136.

Oosterom PJM van (2005) Variable-scale topological data structures suitable for progres-sive data transfer: the GAPface tree and GAP-edge forest. In: Cartography and Geographic Information Science, vol 32, 4, pp 331–346.

Oosterom PJM van (2006) Constraints in spatial data models, in a dynamic context. In: Drummond J, Billen R, João E, and Forrest D (eds). Dynamic and Mobile GIS: Investigating Changes in Space and Time, pp 104–137.

Oosterom PJM van, de Vries M, and Meijers M (2006) Vario-scale data server in a web service context. Workshop ICA Commission on Map Generalisation and Multiple Representation, June, Vancouver, 14p.

Parent C, Spaccapietra S, and Zimányi E (2006) Conceptual modelling for traditional and spatio-temporal applications. The MADS approach. ISBN: 3–540–30153–4.

Sparks G (2001) Database modelling in UML. Sparx Systems whitepaper.

Stoter JE, Quak CW, van Oosterom PJM, Meijers BM, Lemmens RLG, and Uitermark HT (2007) Considerations for the design of a semantic data model for a multi-representation topographical database. In: H. Kremers (ed), Lecture notes in information sciences. Berlin: CODATA, pp 53–71.

West-Nielsen P, and Meyer M (2007) Automated generalisation in a map production environment – the KMS experience. In: Mackaness WA, Ruas A, and Sarjakoski LT (eds). Generalisation of geographic information: cartographic modelling and applications. Amsterdam: Elsevier, pp 301–313.