

A Classification of Generalisation Operators Formalised in OCL

Theodor Foerster, Javier Morales & Jantien Stoter
International Institute for Geoinformation Science and Earth Observation (ITC), the
Netherlands
{foerster, morales, stoter}@itc.nl

Abstract. The lack of formalised generalisation operators has been considered to be one of the obstacles towards achieving fully automated generalisation. This paper proposes an approach based on the Object Constraint Language (OCL) to set up a formalised classification of generalisation operators. The steps presented in this paper are: a) setting up an object model representing the classification of operators, b) describing the operators using OCL and c) applying the formalised classification to generalisation algorithms by formalising the Douglas-Peucker algorithm.

Keywords: generalisation operators, formalisation, Object Constraint Language, automated generalisation.

1 INTRODUCTION

Automated generalisation of geodata is a well-known problem within research, which has been investigated thoroughly, as presented recently in the book of Mackaness et al. (2007). One of the main problems in the context of automated generalisation is the formalisation of the knowledge of the cartographer. In the early days of research on automated generalisation, scientists captured the knowledge of the cartographer by the concept of generalisation operators. Those generalisation operators were a means to describe specific manual transformations of objects on the map on an abstract level, such as simplifying a line. Establishing a classification of generalisation operators is promising, as it improves the communication of related knowledge between humans as well as machines. Several operator classifications have been proposed, but none of them was satisfying. The main reason for this is that the description of the classification could not be formalised.

Formalisation enables one to describe the operators unambiguously and it is becoming more urgent as generalisation functionality is being published through Web Services (Burghardt et al. 2005; Foerster & Stoter 2006; Regnaud 2007). It is required for the meaningful execution and orchestration of such Web Service-based generalisation functionality. In detail it allows one to identify and communicate with the necessary (remote) functionality through a loose coupled and highly heterogeneous environment such as the Web. The ability of communicating with

functionality, which also considers the (formalised) context, has been entitled as semantic interoperability (ISO 2005).

A first effort towards formalising generalisation operators has been achieved by presenting a classification (Foerster et al. 2007), which is based on established data models for geospatial data from the International Organization for Standardization (ISO) and the Open Geospatial Consortium (OGC). Those models were described using the Unified Modeling Language (UML) (Arlow & Neustadt 2005). However, a formal description of the specific operators has been missing so far. This paper adopts the classification of generalisation operators of Foerster et al. (2007) (textual description referring to UML-encoded aspects of the specific data model) and extends it by applying the Object Constraint Language (OCL) (Warmer & Kleppe, 2003). Describing those operators via OCL is promising especially in this context as it provides a seamless shift from the (already existing) UML models towards a formal description. In detail the reasons for applying OCL are the following:

- The data models (published by ISO and OGC) which serve as a foundation for the generalisation operator classification of Foerster et al. (2007) are described in UML.
- OCL is an extension to UML and provides thereby a seamless integration into the already settled classification.
- UML models enriched with OCL can be translated to other formal languages (e.g. OWL Ontology Language for the Web) for instance in the context of the semantic web (Timm & Gannod 2007).

Especially OCL's direct link to UML is a clear advantage over other formalisation mechanisms supported by functional programming languages such as Haskell (Hutton 2007).

A formalised classification contributes to the theory of automated generalisation processing. Furthermore, as generalisation is a representative application of geo-information modelling, this paper contributes to modelling of geospatial processes in general. Additionally it will support future software developments of generalisation functionality by the concept of Model Driven Architecture (MDA) and will help one to describe Web Service-based generalisation functionality in the future.

The remainder of the paper is structured as follows. In the next section the paper will present some related literature on generalisation operators and OCL. Section 3 will then describe the classification of operators of Foerster et al. (2007). Section 4 will present a UML model for the generalisation operators based on the introduced classification. Subsequently, Section 5 will illustrate based on the developed UML model the OCL-based description of the operators. Section 6 will demonstrate how generalisation functionality can be described based on the formalised

operators for the case of the Douglas-Peucker algorithm (Douglas & Peucker 1973). Extending the formalised classification by such an algorithm will prove a) that the proposed model of operators is valid and b) that OCL is an appropriate mechanism to model the relation between generalisation operators and their implementing algorithms especially on a semantic level. Finally, the last section will discuss some open issues and conclude the paper.

2 RELATED LITERATURE

Before introducing the existing classifications of operators this section describes the relation of the terms *generalisation operator* and *generalisation algorithm*.

The idea of generalisation operators evolved within the early generalisation research by extracting abstract descriptions of single actions of the cartographer during manual generalisation. Thus a generalisation operator describes atomic generalisation functionality on an abstract level. The functionality is atomic in the sense, that it only affects well-defined and isolated aspects of data in an undividable way. Nevertheless, this does not imply that the generalisation functionality is without any side effects.

Generalisation operators have been identified as a key concept of abstraction in order to compare and classify different generalisation algorithms. An operator is thereby or can be implemented by different algorithms. A well-known example is the simplification operator, which defines the abstract functionality of thinning out specific aspects of a geometry. An algorithm, which actually implements this abstract functionality, is the Douglas-Peucker algorithm (Douglas & Peucker 1973). This example will also be used within the paper to show, how the relation between generalisation operators and algorithms can be modelled semantically (Section 6).

2.1 Review on Existing Operator Classifications

Classifications of operators are required because they provide an unambiguous definition of the functionality and provide thereby a common foundation when communicating between individuals and/or systems (i.e. interoperability). One of the most prominent classifications of processes within the domain of geo-information modelling is the 9-intersection model of Egenhofer & Franzosa (1991) and the map algebra of Tomlin (1990).

Classifications have always to be based on formal models, in order to become formal. While reviewing the existing classifications of generalisation operators, this was our main requirement and also finally the motivation to start our own classification as presented in Foerster et al. (2007). This section will give a short review on existing classifications and will demonstrate that none of those classifications have been based on

formal models and are therefore not able to be formalised. Table 1 summarises the classification of operators of previous researchers in the field of automated generalisation.

McMaster & Shea		Cecconi et al. (Agent)		Liu et al.
<i>spatial transformations</i>	Simplification	<unspecified>	Thematic selection	Simplification
	Amalgamation		Thematic aggregation	Merge
	Refinement	<i>individual objects</i>	Weeding	Amalgamation
	Displacement		Unrestricted simplification	Aggregation
	Smoothing		Enlargement	Classification
	Merging		Exaggeration	Selection
	Exaggeration		Fractalization	
	Aggregation		Smoothing	
	Collapse	Rectification		
	Enhancement	<i>individual or groups of objects</i>	Selection	
Symbolization	Elimination			
<i>attribute transformations</i>	Classification	Displacement		
		<i>groups of objects</i>	Amalgamation	
			Combine	
			Typification	

Table 1: Overview of operator classifications merged and simplified after McMaster & Shea (1992), Cecconi (2003), Liu et al. (2001); taken from Foerster et al. (2007).

McMaster & Shea (1992) introduced a first classification of generalisation operators, which consists of twelve operators and two categories. Their introduced categorization into spatial transformations and attribute transformations is trivial in the sense that it assigns the classification operator and symbolization operator as attribute transformations and the other operators as spatial transformations. The classification and the attached descriptions of the operators do not apply any formal model, thus the interpretation of the operators is ambiguous. Additionally the classification does not seem to be sufficient to reflect the current aims of data production (decoupled from cartographic issues), as symbolization is mentioned as a generalisation operator. In current research the visualization and the data are separated to reduce complexity for modelling, analysis and maintenance and to supply the data to other applications.

The Agent project (Lamy et al. 1999) focused on enhancing automated cartographic generalisation for map production. The aim of the project was to develop a hierarchy of communicating objects (so called agents), which try to solve cartographic conflicts on the level of single features and groups of features. Thus they subdivided the operators into these two groups. Nevertheless, the classification lacks also of a formal model, which allows one to define uniquely the operators and to separate them.

The classification of Liu et al. (2001) aims at an object-oriented framework for model generalisation operators but it mixes up the concepts

of constraints for cartographic generalisation and model generalisation¹. Additionally some important operators for geometry type transformation are not covered such as combine and collapse (which are covered by the other classifications). This classification does not provide a formal basis either, which finally results in a classification that cannot be formalised.

A similar overview of existing classifications has been published recently by Li (2007). Also in the recently published book by Li (2006) such a review of operators is covered. However Li's reviews are based on a geometry-oriented perspective, which might not be sufficient for current generalisation applications, because they are modelled following a feature-oriented approach.

2.2 Concepts to Formalise Geospatial Processes

The formalisation of geospatial processes is one of the main challenges in current geospatial research in order to enhance the automated and meaningful use of Geospatial Web Services. This challenge has been also identified as important for generalisation processing (Regnauld 2007; Foerster et al. 2007a). One of the key concepts to accomplish such automated and meaningful Web Service-based processing is the ontology design. An ontology allows one to model the semantics of Web Services and enables thereby their discovery and composition. Thus an ontology enhances the semantic interoperability of Web Services. Ontologies in the context of Web Services and geospatial processes have been investigated for instance by Lemmens (2006) and Lutz (2007).

It is important to note that the term ontology can be seen as an analogy to what in the generalisation context has been called classification. Regarding the different wording, we want to stick to the term classification, as this has been mostly used in the generalisation research community.

Due to the reasons pointed out in Section 1, this paper will apply OCL for describing the operator classification. Finally this OCL can be used to be applied to the concept of Web Services using the approach of Timm & Gannod (2007), who developed an MDA-based approach for semantic interoperable Web Services.

2.3 Introduction to the Object Constraint Language

The Object Constraint Language² (OCL) has been designed by the Object Management Group (OMG³) to add concepts to specify the semantics of UML models more detailed. The final aim was to enhance the Model Driven Architecture (MDA) and to have a seamless integration of models and software code (i.e. software code is generated through models). The

¹ Definition of constraints in the context of generalisation research please refer to Beard (1991).

² OCL is currently specified as version 2.0 (OMG 2006).

³ OMG website: www.omg.org.

level of this seamless integration has also been introduced as Modelling Maturity Levels (Warmer & Kleppe, 2003). The gap between different levels of modelling and the actual software code has been identified as one of the main problems in software engineering and data model design. The objective of OCL is to avoid typing programming code. The model is specified via UML and OCL and is used to generate programming code automatically and platform-independently. So in practice this means that a Java Enterprise application, but also a .Net Server application can be generated based on the same model. Additionally, OCL can be used to check and validate existing models and thereby to verify the implementation (Beckert et al. 2007). OCL has become a part of UML and gained already some interest in industry as well as research (Garcia & Moeller, 2007).

OCL is a declarative and strongly typed language to query and describe additional semantics of object-oriented models. It can be attached to different types of UML diagrams such as class diagrams and state diagrams. It allows one to specify invariant states of objects and attributes. Methods can be defined via pre and post conditions or via a body expression. Additionally OCL defines a comprehensive mechanism of object querying, which is basically realized by different set operations (indicated with \rightarrow).

As generalisation operators are meant to be abstract, we will specify the generalisation operators by the means of pre conditions (which are valid before executing the method) and post conditions (which are valid after execution). The implementations will then inherit the specified conditions according to Liskov's Substitution Principle and extend the specification by describing the body, as it will be demonstrated in Section 6.

3 GENERALISATION OPERATOR CLASSIFICATION

The basic idea behind the classification of Foerster et al. (2007) is to separate operators according to their intended purpose (i.e. producing data vs. producing maps). This scheme thereby follows the generalisation approach of Gruenreich (1992), also see Figure 1. This approach distinguishes between model generalisation and cartographic generalisation thus separates generalisation operators into two groups:

- model generalisation operators
- cartographic generalisation operators.

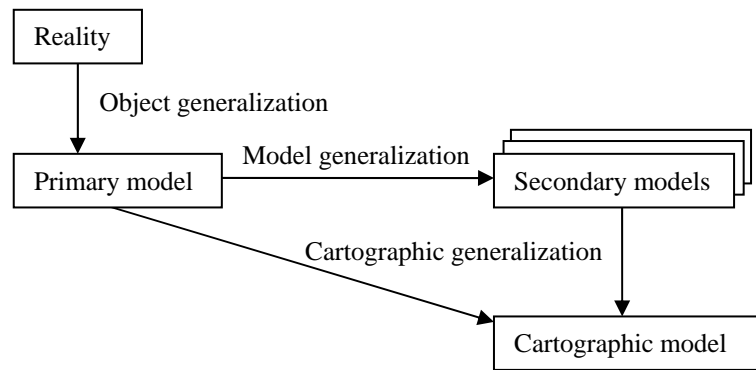


Figure 1: Generalisation approach of Gruenreich (1992).

To define the different operators unambiguously, Foerster et al. (2007) described each group of operators by means of a specific data model. The *ISO 19109 model* (ISO 2003) serves as a basis to formalise the model generalisation operators and the *OGC Go-1 Application Objects model* (OCG 2005) allows one to formalise the cartographic generalisation operators. These models enable one to formalise the abstract functionality encapsulated in each of the generalisation operators by describing their impact (Figure 2).

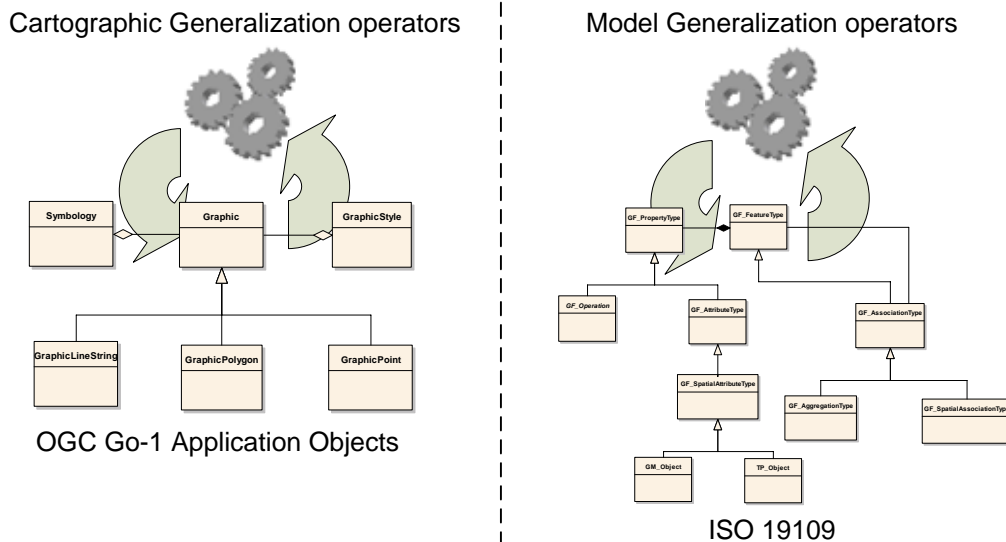


Figure 2: Each group of operators is defined upon a specific data model.

To determine the group (model or cartographic) that an operator belongs to, the following criteria are used: if the operator had an impact upon a complete dataset, i.e. set of features from the ISO 19109 model (model generalisation); or only on a single cartographic feature or on a group of cartographic features from the OGC Go-1 Application Objects model (cartographic generalisation). Classifying operators in this way goes inline with the common view (Weibel & Dutton 1999) i.e.: generalizing a map is based on cartographic conflicts between specific feature instances due to

cartographic constraints; and generalizing a dataset is based on formal rules applied to the whole dataset, which ensures that later data analysis produces reliable results.

Table 2 presents a list of operators, extracted from literature (presented in Section 2) and organized based on the introduced criteria.

<i>Model generalisation</i>	Class Selection
	Reclassification
	Collapse
	Combine
	Simplification
	Amalgamation
<i>Cartographic Generalisation</i>	Enhancement
	Displacement
	Elimination
	Typification
	Enlargement
	Amalgamation

Table 2: Operator affiliation to the Gruenreich model (based on Foerster et al. (2007)).

By now this generalisation operator classification has been only described in plain text, by referring to parts of the data models of ISO and OGC. The formal description of the operators will be introduced in the following section using OCL.

It is important to note, that the applied classification of Foerster et al. (2007) is only one possible way to formalise generalisation operators. However, we claim, that the link to established models of ISO and OGC provides main advantages to carry out a formalisation, which is general and can be easily applied to existing generalisation applications (which already use these data models of OGC and ISO).

4 AN OBJECT ORIENTED MODEL FOR GENERALISATION OPERATORS

The foundation of the formalised operators is an object oriented model described in a UML class diagram reflecting the generalisation operator classification as it has been described before. Thus to represent the basic separation of model and cartographic generalisation operators, the model specifies two interfaces *ModelGeneralisationOperator* and *CartographicGeneralisationOperator* (Figure 3). Interfaces define the methods (i.e. the syntax), which have to be implemented by the subclasses. In an appropriate design, all classes implementing the same interface share the same semantics regarding the inherited methods. Both interfaces have the method *process()*. Depending on the impact of the operator, the method

consumes a parameter (Figure 3) representing a set of features (ISO 19109 model) or a set of cartographic features (OGC Go-1 Application objects model).

The actual operators (Table 2) have been designed as separate classes. This enables algorithms to inherit from a specific operator. Each operator implements one of the interfaces regarding its affiliation to model or cartographic generalisation. A special case of inheritance is Amalgamation, which implements both interfaces, as it might apply to both model and cartographic generalisation.

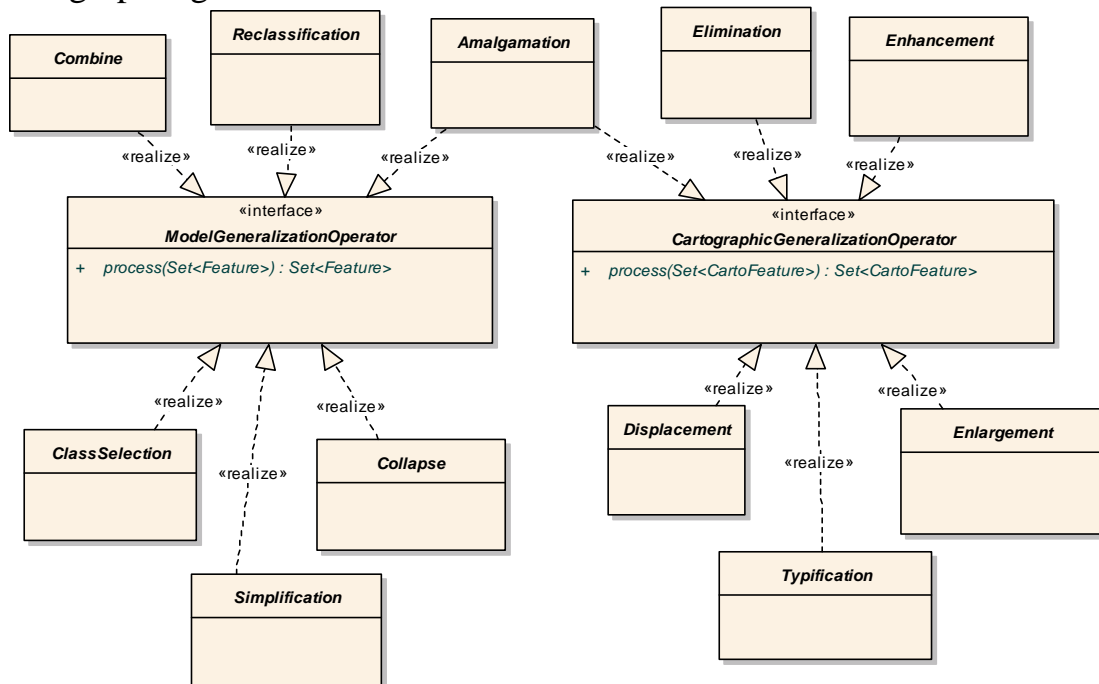


Figure 3: UML model of the generalisation operators.

It is important to note, that the modelled operators do not reflect any specific parameters apart from the parameters representing the set. Parameters are dependent on the specific algorithm and its implementation. Therefore parameters have to be modelled at a lower abstraction level. An example that illustrates how to handle algorithm-specific parameters is presented in Section 6.

Each operator is modelled as an abstract class, which ensures that the operator cannot be instantiated and thereby has to be implemented by a subclass (i.e. the generalisation algorithm). However, the operators are not modelled as interfaces because the interfaces do not allow one to specify any functionality. The abstract functionality of the operators is specified via pre and post conditions (Section 5.1 & 5.2), because the operators do not describe any explicit functionality. The implementing generalisation algorithms are then specified via the body OCL expression to represent the functionality (Section 6).

5 FORMALISING GENERALISATION OPERATORS IN OCL

Based on the model of operators depicted in Figure 3 the following two subsections illustrate the operator formalisation approach by describing a subset of operators of both types (model and cartographic) textually and then formalising them in OCL. Details about the syntax of OCL can be found in Warmer & Kleppe (2003). The syntax of the OCL statements presented here has been validated using ArgoUML⁴ (an Open Source modelling tool). This syntax checker uses the OCL parser from the Dresden OCL toolkit⁵.

5.1 Model Generalisation Operators

This section illustrates based on examples of the Collapse and the Simplification operator, how OCL can be used to specify model generalisation operators more thoroughly.

Collapse

This operator decreases the dimensionality of the geometry of a feature, i.e., it collapses a geometry from polygon to line or point or from line to point. One of the most dominant applications for this operator is the collapse of road polygon geometries to road line geometries representing the road's center line (Figure 4).

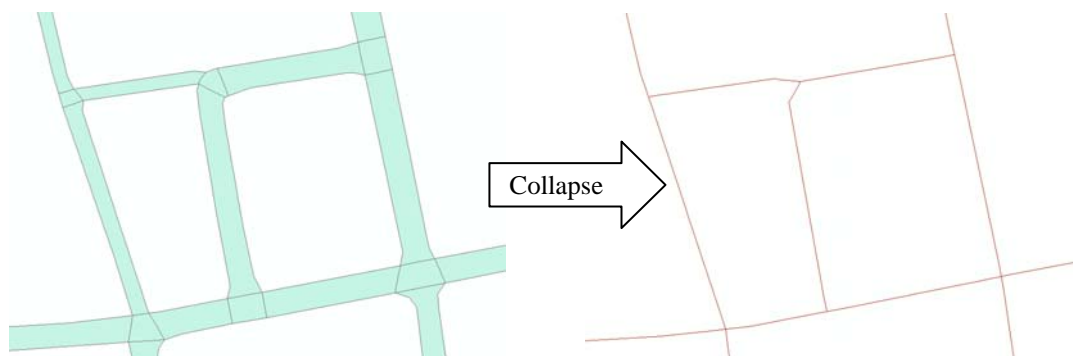


Figure 4: An example of collapsing a set of polygon features to line features.

We specify the functionality of the operator by defining a pre and a post condition. The pre condition ensures that the features passed to the operator contain a collapsible geometry, that is a line or polygon geometry. The post condition defines that the resulting geometry must have a lower dimensionality with respect to the initial geometry.

```
Context Collapse::process(fc: Set<Feature>) : Set<Feature>
pre: fc->forAll(Feature f | geometry.dimension>=1)
post: result->forAll(Feature f | f.geometry.dimension <
    fc.getByFid(f.fid).geometry.dimension)
```

⁴ ArgoUML website: <http://www.argouml.org/>

⁵ Dresden OCL toolkit website: <http://dresden-ocl.sourceforge.net/>

Simplification

This operator is mainly used to reduce the amount of data. It deletes specific vertices (point members) of a geometry, but it does not change or add any coordinates (Figure 5).

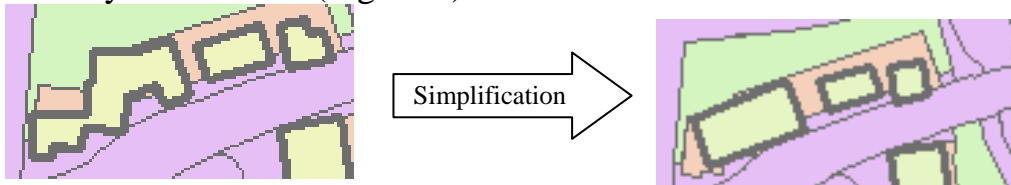


Figure 5: An Example of simplifying polygonal features.

This example also uses a pre and a post condition. The pre condition specifies that all features consist of a geometry, which is at least of type linestring. Additionally, coordinates should not be changed or added to the geometry. Therefore we define a `changeAllowed` attribute to the coordinate (`changeAllowed=false`) and an `addAllowed` attribute for the geometry (`addAllowed=false`). The post conditions define that the number of coordinates has to be decreased. The OCL statements below also refer to *Geometry* and *Coordinate*, both are taken from the ISO 19109 model.

```
Context Simplification::process(fc: Set<Feature>) : Set<Feature>
  pre: fc->forAll(Feature f|f.geometry.dimension>=1)
      fc->forAll(Feature f|f.addAllowed = false)
      fc->forAll(Feature f|f.geometry->forAll(Coordinate c|
        c.changeAllowed = false))
  post: result->forAll(Feature f | f.geometry.nGeometry()<
      fc.getByFid(f.fid).geometry.nGeometry())

Context Coordinate
  def: changeAllowed: Boolean = false

Context Geometry
  def: addAllowed: Boolean = false
```

5.2 Cartographic Generalisation Operators

This section illustrates based on the example of the Displacement operator, how OCL can be used to specify cartographic operators more thoroughly.

Displacement

This operator moves the complete cartographic object by applying the same vector to each part of the object. The final result is an object with a changed location but still preserving the original shape, also in absolute terms. The example illustrates, that the polygons are displaced from their origin to be more visible on the target map (Figure 6). Note that, the deformation of designated parts of a feature, as displacing parts of a line is addressed by the Enhancement operator. To specify the Displacement operator we define a post condition that determines a displacement vector for an arbitrary coordinate pair in the result geometry. This displacement

vector is then compared for all coordinate pairs and should remain constant.

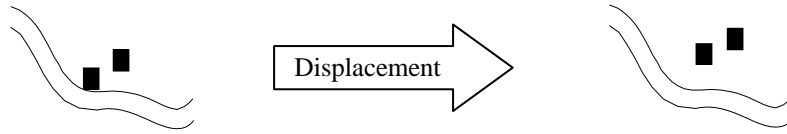


Figure 6: An example of displacing a polygonal feature.

```
Context Displacement::process(fc :Set<CartoFeature>) :
Set<CartoFeature>
  post: let vec: TupleType(x: Real, y:Real)=TupleType{x:
  result->first.geometry.x - fc.getByFidId(result-
  >first.fid).geometry.x), y: result->first.geometry.y -
  fc.getByFidId(result->first.fid)result-
  >first.geometry.y}

  result->forall(Feature f | f.geometry.x =
  result.getByFid(f.fid).geometry.x = vec.x and f.geometry.y =
  result.getByFid(f.fid).geometry.y = vec.y)
```

6 FORMALISING DOUGLAS-PEUCKER ALGORITHM IN OCL

In order to demonstrate, how the OCL definitions (Section 5.1 & 5.2) can be used to specify the implementing algorithms, this section will introduce an OCL description of the Douglas-Peucker algorithm (Douglas & Peucker 1973). Following the idea of generalisation operators and generalisation algorithms, the Douglas-Peucker algorithm by being a Simplification algorithm inherits from Simplification (Figure 7).

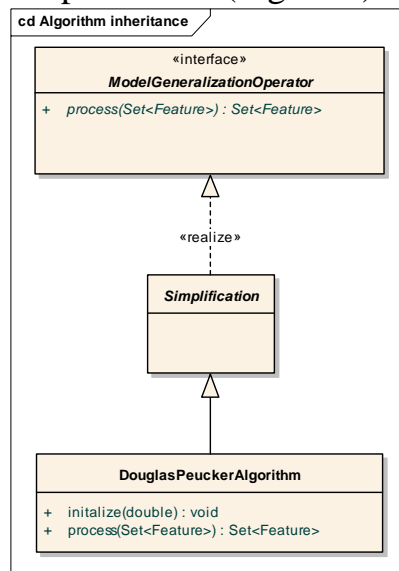


Figure 7: UML model of Douglas-Peucker Algorithm inheriting the Simplification operator.

According to the principle of inheritance, all the pre and post conditions defined for Simplification are valid for any subclass. This particular algorithm is configured using a parameter, which drives the selection of

points for the simplification. For a description of the algorithm see Douglas & Peucker (1973).

The OCL specification below has one important aspect. As the Douglas-Peucker algorithm has a recursive behavior, the algorithm calls itself several times. To cater for this, *process()* has a call to an atomic function called *simplifyGeom()*. This does not affect the conditions defined in the superclass. The pre and post conditions will still be met, as they are defined for *process()*.

```
Context DouglasPeuckerAlgorithm::init(dpTolerance: double):
Set<Feature>
  inv: dpTolerance > 0
      self.dpTolerance = dpTolerance

Context DouglasPeuckerAlgorithm::process(fc: Set<Feature>)
  body: result = fc->forall(Feature f|
      f.setGeom(Geometry(simplifyGeom(f.geometry)))

Context DouglasPeuckerAlgorithm
  def: simplifyGeom(geom:Geometry) Set(coords)
  body:
    if(geom.size >2)
      let cand : Geometry = geom.coordinates-
      >sortBy(c.distance(Line(geom.first,geom.last)))->at(1)
      if(cand.distance(Line(geom.first,geom.last))>
      dpTolerance
        let coords: Geometry
        add(cand)

        coords->prepend(simplifyGeom(geom(first,cand)))
        coords->append(simplifyGeom(geom(cand,last)))
        result = coords
      endif
    endif
```

7 CONCLUSION

This paper presents an approach to formalise a generalisation operator classification such as that of Foerster et al. (2007). As the proposed classification was based on UML models of ISO and OGC, this paper describes the formalisation based on the UML-related concept of OCL. This paper first proposes a UML model describing the generalisation operators (Figure 3; Section 4), which directly refers to the models of ISO and OGC. The UML model is extensible, as any new operator could be attached to the classification.

Subsequently, this UML model for operators is formalised by means of OCL to capture specific operator semantics (Section 5.1 & 5.2). Based on the described semantics (encoded as pre and post conditions), we also present how specific generalisation algorithms can be formalised with OCL (Section 6), based on the specification of an operator. This last step shows

three things: First our approach of using OCL is valid. Secondly it proves that the proposed UML model of operators supports the formalisation of a generalisation algorithm. Finally it proves that the classification of Foerster et al. (2007) provides a valid foundation for modelling generalisation functionality on a semantic level.

After specifying the Douglas-Peucker algorithm with OCL it would be interesting to verify if the description actually meets the inherited conditions of the Simplification operator, as specified in Section 5.1. This would require a tool, which could actually do reasoning between both OCL descriptions. Unfortunately such functionality is currently not yet available. Following the principle of the Model Driven Architecture, it would be possible to directly generate code from the OCL-enriched UML model. Again, tools available today do not support this kind of task. Nevertheless, enriching UML models with OCL is a promising approach, as it actually allows one to capture those semantics, which can not be directly expressed in UML models as such.

Additionally, the OCL statements in combination with the UML models can be used in the context of MDA to extract semantic descriptions for Web Services (Timm & Gonnot 2007). Based on the approach as described in Timm & Gonnot, it is possible to extract semantic descriptions for Web Services by using the presented OCL statements and thereby to enable semantic interoperability of generalisation functionality in the future. Semantic interoperability will allow one to reason on generalisation functionality and overall to perform automated and meaningful generalisation processing on the Web (Regnauld 2007).

Currently the OCL descriptions of the operators do not carry out any cross dependencies such as, that a certain operator is relying on another. Formalising such aspects would enhance the automated orchestration of generalisation operators and finally lead to a more comprehensive model. The modelling of such dependencies is an aspect that we will incorporate in future research. From the application of cartographic generalisation linking the formalised operators to cartographic constraints, as for instance formalised in Burghardt et al. (2007), would lead to a holistic formalisation of the cartographic generalisation process. From the application of model generalisation it would be interesting to link the formalised operators to data models of for instance topographic data (Stoter et al. 2008), which are also formalised in UML and OCL. This could finally lead to a MDA-approach for multi-scale data models.

ACKNOWLEDGEMENTS

The presented research is part of Theodor Foerster's PhD thesis work, which is sponsored by the RGI research program⁶. The authors want to thank the anonymous GI-days reviewers for their constructive remarks.

REFERENCES

- Arlow, J. & Neustadt, I. (2005): UML 2 and the Unified Process: Practical Object-Oriented Analysis and Design (2nd Edition) *Addison-Wesley Professional*, 624 p.
- Beard, K. M. (1991): Map Generalization: Making Rules for Knowledge Representation Constraints on rule formation. *Buttenfield, B. & McMaster, R. (ed.): Map Generalization: Making Decisions for Knowledge Representation, Longman*, 121-135.
- Beckert, B.; Hähnle, R. & Schmitt, P. H. (ed.) (2007): Verification of Object-Oriented Software: The KeY Approach. *Springer-Verlag*.
- Burghardt, D.; Neun, M. & Weibel, R. (2005): Generalization Services on the Web - A Classification and an Initial Prototype Implementation. *Auto-Carto 2005*.
- Burghardt, D.; Schmid, S. & Stoter, J. E. (2007): Investigations on cartographic constraint formalisation. *Workshop of the ICA Commission on Generalisation and Multiple Representation*.
- Cecconi, A. (2003): Integration of Cartographic Generalization and Multi-Scale Databases for Enhanced Web Mapping. PhD thesis, *University of Zurich*.
- Douglas, D. H. & Peucker, T. K. (1973): Algorithms for the reduction of the number of points required to represent a digitized line or its caricature. *The Canadian Cartographer*, 10, 112-122.
- Egenhofer, M. J. & Franzosa, R. (1991): Point-set topological spatial relations. *International Journal of Geographic Information Systems*, 5, 161-174.
- Foerster, T. & Stoter, J. (2006): Establishing an OGC Web Processing Service for generalization processes. *ICA workshop on Generalization and Multiple Representation*.
- Foerster, T.; Stoter, J. & Kobben, B. (2007): Towards a formal classification of Generalization operators. *Proceedings of the International Cartographic Conference 2007*.
- Foerster, T.; Stoter, J. & Lemmens, R. (2007a): Towards automatic web-based generalization processing: a case study. *10th ICA Workshop on Generalisation and Multi Representation*.
- Garcia, M. & Möller, R. (2007): Certification of Transformations Algorithms in Model-Driven Software Development. *Bleek, W.; Räsch, J. & Züllighoven, H. (ed.), Software Engineering 2007, 105, 107-118*.
- Hutton, G. (2007): *Programming in Haskell*. Cambridge University Press.
- Lamy, S.; Ruas, A.; Demazeu, Y.; Jackson, M.; Mackaness, W. & Weibel, R. (1999): The Application of Agents in Automated Map Generalization. *Proceedings of the 19th International Cartographic Conference*.
- Lemmens, R. L. G. (2006): Semantic interoperability of distributed geo-services. *PhD thesis, Delft University of Technology*.
- Li, Z. (2006): Algorithmic foundation of multi - scale spatial representation. *CRC*.

⁶ RGI stands for *Ruimte for Geoinformatie* (in English: space for geoinformation); project website: www.rgi.nl.

- Li, Z. (2007): Essential operations and algorithms for geometric transformations in digital map generalization. *Proceedings of the International Cartographic Conference 2007*.
- Liu, Y.; Molenaar, M. & Tinghua, A. (2001): Frameworks for Generalization Constraints and Operations Based on Object-Oriented Data Structure in Database Generalization. Du, D.; Li, H.; Wang, J.; Zhang, Q.; Jiang, J.; Liao, K.; Wu, H. & Zheng, J. (ed.), *Proceedings of the 20th International Cartographic Conference*, 3, 2000-2012.
- Lutz, M. (2007): Ontology-based Descriptions for Semantic discovery and composition of Geoprocessing Services. *GeoInformatica*, 11, 1-36.
- ISO (2003): Geographic information - Rules for application schema, ISO 19109.
- ISO (2005): Geographic information – Services. *International Organization for Standardization*.
- Mackaness, W. A.; Ruas, A. & Sarjakoski, L. (ed.) (2007): Generalisation of geographic information: cartographic modelling and applications. *Elsevier*.
- McMaster, R. B. & Shea, K. S. (1992): Generalization in Digital Cartography. *American Association of Geographers*.
- OGC (2005): GO-1 Application Objects. Open Geospatial Consortium, Inc.
- OMG (2006): Object Constraint Language. *Object Modeling Group*.
- Regnaud, N. (2007): Evolving from automating existing map production systems to producing maps on demand automatically. *10th ICA Workshop on Generalisation and Multiple Representation*.
- Stoter, J. E.; Morales, J.; Lemmens, R. L. G.; Meijers, M.; van Oosterom, P.; Quak, W. & Uitermark, H. (2008 to be published): A Data model for Multi-Scale topographical data. *Proceedings of SDH 2008, LNCS, Springer*.
- Timm, J. T. E. & Gannod, G. C. (2007): Specifying Semantic Web Service Compositions using UML and OCL. *IEEE International Conference on Web Services (ICWS 2007)*, 521-8.
- Tomlin, C. D. (1990): Geographic Information Systems and Cartographic Modeling. *Prentice Hall College Div*.
- Warner, J. & Kleppe, A. (2003): The Object Constraint Language. *Addison Wesley*, 206 pp.
- Weibel, R. & Dutton, G. (1999): Generalising spatial data and dealing with multiple representations. Longley, P.; Goodchild, M.; Maguire, D. & Rhind, D. (ed.), *Geographic Information Systems - Principles and Technical Issues*, John Wiley & Sons, 1, 125-155.